Titre: Title:	Une étude des réseaux de neurones artificiels pour la classification rapide d'impulsions radars
Auteur: Author:	Éric Granger
Date:	2002
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Granger, É. (2002). Une étude des réseaux de neurones artificiels pour la classification rapide d'impulsions radars [Ph.D. thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7054/

UNIVERSITÉ DE MONTRÉAL

UNE ÉTUDE DES RÉSEAUX DE NEURONES ARTIFICIELS POUR LA CLASSIFICATION RAPIDE D'IMPULSIONS RADARS

Éric GRANGER
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR (Ph.D.)
GÉNIE ÉLECTRIQUE
janvier 2002

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Cette thèse intitulée:

UNE ÉTUDE DES RÉSEAUX DE NEURONES ARTIFICIELS POUR LA CLASSIFICATION RAPIDE D'IMPULSIONS RADARS

présentée par: Éric GRANGER

en vue de l'obtention du diplôme de: Philosophiæ Doctor (Ph.D.)

a été dûment acceptée par le jury d'examen constitué de:

M. BRAULT, Jean-Jules, Ph.D., président

M. SABOURIN, Robert, Ph.D., membre

M. SAVARIA, Yvon, Ph.D., membre et directeur de recherche

M. SCIORTINO, John, Ph.D., membre externe

Remerciements

L'auteur tient d'abord à remercier son directeur de recherche, le Dr. Yvon Savaria, pour l'avoir supervisé durant ses études graduées. Ses remerciements vont ensuite au Dr. Pierre Lavoie du Centre de Recherches pour la Défense Ottawa (CRDO). L'un comme l'autre ont contribué, à plusieurs niveaux, aux travaux de cette thèse, et se sont montrés disponibles tout au long du processus. L'auteur est très reconnaissant pour leurs commentaires constructifs, pour leurs encouragements, et pour avoir cru en ses capacités.

L'auteur désire ensuite exprimer sa gratitude envers le CRDO pour le support matériel et financier. L'auteur tient aussi à remercier le Dr. Stephen Grossberg, du Department of Cognitive and Neural Systems (CNS), à l'université de Boston, pour lui avoir permis d'effectuer un stage dans son laboratoire. Des remerciements vont ensuite au personel du CNS, en particulier le Dr. Stephen Grossberg et le Dr. Mark Allen Rubin, pour l'avoir guidé dans sa recherche portant sur les réseaux de neurones pour l'identification d'émetteurs radars.

L'auteur est également reconnaissant au Conseil de Recherche en Science Na-

turelles et Génie (CRSNG) du Canada, au Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR) du Québec, et à la Fondation de Polytechnique, pour leur soutien financier.

Finalement, l'auteur tient à remercier sa famille, ses amis et en particulier sa conjointe, Kirsten, pour son support moral.

Résumé

Les systèmes de Mesures de Soutien Électronique (MSE) radar sont exposés à des environnements électromagnetiques qui sont de plus en plus denses et complexes. Cette thèse explore le potentiel des réseaux de neurones artificiels (RNA) pour classifier des signaux radars dans ces systèmes. La thèse comporte quatres contributions qui sont organisées en deux volets. Le premier volet (lié aux trois premières contributions) concerne le triage métrique rapide d'impulsions radars, tandis que le deuxième volet (lié à la dernière contribution) concerne l'identification des types de radar.

Dans la première contribution, quatre RNA auto-organisateurs de type apprentissage compétitif ont été comparés en termes de leur qualité de catégorisation et de leur effort de calcul. Des résultats de simulation avec un ensemble de données radar et des estimations de complexité ont permis de conclure que deux de ces réseaux — le Self-Organizing Feature Mapping et le Fuzzy Adaptive Resonance Theory (ART) — sont d'excellents candidats pour le triage métrique rapide d'impulsions.

La deuxième contribution est la proposition d'une architecture de système intégré à très grande échelle (VLSI), qui permet la mise en oeuvre du RNA fuzzy ART pour

des applications de triage métrique rapide. Cette architecture modulaire partitionne la fonctionalité de fuzzy ART sur plusieurs ASIC cascadables. Son traitement est pipeliné grâce à une architecture systolique en anneau pour la comparaison rapide entre entrées et poids. Un modèle d'estimation AT pour cette architecture a permis d'isoler un ensemble de configurations d'architecture qui peuvent supporter un taux de traitement très élevé, tout en occupant une surface acceptable.

La troisième contribution est la proposition d'une technique pour gérer la manière dont les patrons d'un séquence d'entrée sont appris par un système de catégorisation. Selon cette technique, un patron d'entrée qui mène à une décision ambiguë est emmagasiné dans une file, et son apprentissage est retardé pendant un délai fixe. Des résultats de simulation obtenus avec un ensemble de données radars et deux RNA (ART2A-E et fuzzy ART) ont permis de conclure (1) que le nombre de patrons d'entrée qui mène à une décision ambiguë est indicatif de la dégradation des résultats, et (2) que ce traitement offre une alternative intéressante aux autres techniques en termes du compromis entre la qualité des catégorisations et le temps de réponse.

Dans la quatrième contribution, un RNA à fusion "what-and-where" a été proposé pour l'identification rapide des types de radar associés aux impulsions interceptées. Les paramètres "what" forment l'entrée pour le RNA classificateur qui prédit les types de radar associés aux impulsions. Les paramètres "where" forment l'entrée pour le sous-système de catégorisation en-ligne qui sépare les impulsions transmises par différents émetteurs. Cette séparation permet d'accumuler les réponses du clas-

sificateur pour chaque émetteur, et donc de prédire le type de radar d'un émetteur actif d'après une séquence d'impulsions. Des simulations effectuées pour une mise-en-oeuvre particulière du RNA à fusion "what-and-where" et pour un ensemble de données radars, ont démontrées une amélioration significative des performances par rapport à un classificateur seul.

Abstract

Electronic Support Measures (ESM) systems are being exposed to electromagnetic environments that are increasingly dense and complex. This thesis explores the potential of artificial neural networks (ANNs) for the classification of radar signals within these systems. It comprises four contributions. The first three concern ANNs for fast metric sorting of radar pulses, whereas contribution four concerns the identification of radar types.

The first contribution presents a comparison of four competitive learning neural networks in terms of clustering quality and computational efficiency. Simulation results, obtained using radar pulse data, and complexity estimates have indicated that two of these ANNs, Self-Organizing Feature Mapping and Fuzzy Adaptive Resonance Theory (ART), are excellent candidates for fast metric sorting of radar pulses.

The second contribution proposes a VLSI system architecture that can implement the fuzzy ART neural network for high throughput metric sorting applications. This modular architecture partitions the fuzzy ART algorithm onto several cascadable ASICs. It employs a systolic ring architecture for rapid comparison between input

patterns and synaptic weights. An area-time estimation model has allowed to isolate a set of architecture configurations that can sustain a very high data rate, yet occupy an acceptable silicon area.

The third contribution proposes a technique to manage the way a stream of input patterns are learned by an on-line clustering system. According to this technique, input patterns that lead to ambiguous decisions are stored for fixed time before being learned. Simulation results obtained using a radar pulse data set, and the ART2A-E and fuzzy ART ANNs, have allowed to conclude that (1) the number of patterns leading to ambiguous decisions are indicative of poor clustering quality, and that (2) this technique offers an interesting alternative to other techniques, in terms of a compromise between clustering quality and response time.

Finally, the fourth contribution presents a "what-and-where" fusion ANN for fast identification of radar types associated to intercepted radar pulses. The "what" pulse parameters are fed to an ANN classifier that predicts the radar types linked to pulses, whereas "where" pulse parameters are fed to an on-line clustering system that sorts pulses transmitted from different active emitters. This separation allows to accumulate the classifier's responses for each emitter, and thus predict corresponding radar types from a sequence of pulses. Simulation results on a particular implementation of the "what-and-where" fusion ANN, using a radar pulse data set, has shown significant improvements over the performance of a classifier alone.

Table des matières

Liste des tableaux

Liste des figures

Introduction

Un système de Mesures de Soutien Électronique (MSE) radar a pour but la détection et l'identification passive de signaux radars pour des fins militaires. Son traitement se résume (1) à détecter les impulsions liées aux signaux radars qui sont interceptés et puis à mesurer certains paramètres; (2) à regrouper les impulsions qui sont perçues comme similaires d'après leurs paramètres; et (3) à identifier les types d'émetteurs associés aux groupes formés. La réponse d'un système de MSE radar est critique en guerre électronique, puisqu'elle permet de prévoir les menaces dans un environnement, pour ensuite entreprendre des contre-mesures appropriées. Dans le contexte actuel, les systèmes de MSE radar sont confrontés à des environements électromagnetiques qui sont de plus en plus denses et complexes. En même temps, nos attentes pour la vitesse, la précision et la fiabilité des ces systèmes augmentent. Ces tendances motivent la recherche d'approches alternatives plus puissantes qui peuvent prévenir contre les menaces futures.

Les systèmes de MSE modernes évoluent vers des systèmes autonomes et adaptatifs, qui peuvent traiter la plupart des signaux interceptés avec un minimum d'interact-

ion humaine. Les réseaux de neurones artificiels (RNA) peuvent jouer un rôle important dans cette évolution. Un RNA est une structure pour le traitement d'information qui s'inspire du modèle biologique. Cette structure peut être implantée de façon massivement parallèle. Elle est alors constituée d'un grand nombre de processeurs élémentaires simples (neurones) qui sont très interconnectés. Les connaissances acquises (poids synaptiques) sont distribuées à travers la structure, à l'endroit des interconnexions. En général, on spécifie un RNA par son modèle de neurone, par sa topologie d'interconnexion, et par sa loi d'apprentissage pour ajuster les poids synaptiques.

Le traitement neuronique est fondamentalement différent des approches conventionnelles pour le MSE radar. Un élément qui distingue les RNA des approches conventionnelles est la capacité d'apprendre, de rappeler et de généraliser à partir d'exemples, les règles nécessaires pour (par exemple) classifier des signaux radars. Cet attribut est avantageux lorsqu'il est difficile de modéliser un environnement complexe de façon explicite. De plus, étant donnée l'architecture parallèle, un RNA peut donner un traitement très rapide et robuste s'il est realisé avec un circuit approprié. La performance d'un RNA se dégrade alors progressivement en présence de données bruitées, corrompues et incomplètes.

Cette thèse porte sur l'application des RNA aux MSE radar. Plus spécifiquement, elle explore le potentiel des RNA pour effectuer deux fonctions critiques: le triage métrique d'impulsions radar et l'identification des types d'émetteurs radars.

Cet ouvrage comporte cinq chapitres. Le premier est un chapitre de synthèse, tandis que les quatres derniers contiennent les contributions de la thèse. Le premier chapitre présente un sommaire bref d'un système de MSE conventionnel, dans le but de faciliter la compréhension des chapitres subséquents. Les défis rencontrés par ces systèmes dans le contexte actuel, ainsi qu'un survol des quatres contributions contenues dans cette thèse, sont aussi décrits. Les quatres contributions sont organisées en deux volets. Le premier volet (lié aux trois premières contributions) traite des RNA pour le triage métrique rapide d'impulsions radars, tandis que le deuxième (lié à la dernière contribution) traite des RNA pour l'identification des types de radar.

Le chapitre deux entame la description de la première contribution. C'est une comparaison entre quatre RNA auto-organisateurs de type apprentissage compétitif, qui peuvent supporter le triage rapide de séquences d'impulsions. La performance de ces RNA a été examinée sous trois angles différents: la qualité des catégorisations, le temps de convergence et la complexité de calcul. L'ensemble de données utilisées pour les simulations est dérivé d'impulsions radars recueillies dans le champ par le Centre de Recherches pour la Défense Ottawa (CRDO). Afin d'observer la variabilité des résultats de catégorisation face à l'ordre de présentation, les patrons de cet ensemble ont été organisés selon trois ordres de présentation statistiques. Les résultats de simulation, ainsi que les estimations de complexité ont été analysées pour le triage rapide en MSE radar.

Le chapitre trois contient une description de la deuxième contribution. C'est

la proposition d'une architecture de système intégré à très grande échelle (VLSI) qui permet la mise en oeuvre du RNA fuzzy Adaptive Resonance Theory (ART) pour des applications de triage rapide. L'architecture proposée pour le système est modulaire et cascadable en fonction des besoins de l'application. Elle comprend un comparateur global, ainsi qu'un ensemble de modules élémentaires identiques, qui permettent chacun d'émuler un certain nombre de neurones. À l'intérieur de chaque module élémentaire, le traitement est pipeliné grâce à une architecture systolique en anneau pour la comparaison rapide entre entrées et poids. Un modèle pour l'analyse du coût AT été développé pour cette architecture dans le but d'évaluer l'impact du choix d'une configuration sur sa surface semiconducteur (A) et son temps de traitement (T). Le modèle a été utilisé pour estimer la performance de l'architecture fuzzy ART pour le triage métrique rapide en MSE radar.

Le chapitre quatre décrit la troisième contribution. C'est la proposition d'une technique, nommée *traitement par ré-ordonnancement*, pour gèrer la manière dont les patrons d'un séquence d'entrée sont appris par un système de catégorisation. Lorsqu'un patron d'entrée mène à une décision ambiguë, il est emmagasiné dans une file, et son apprentissage est retardé pendant un délai fixe. Ce traitement est un compromis entre le traitement séquentiel (de base) qui est le plus rapide, et le traitement par lots qui donne les meilleurs résultats. La qualité et la latence requise pour effectuer des catégorisations en ligne ont été comparées pour un système de catégorisation qui utilise le traitement séquentiel, par lot et par ré-ordonnancement. Des simulations

ont été effectuées avec le même ensemble de données radar qu'au chapitre deux, ainsi que deux RNA auto-organisateurs de type apprentissage compétitif. Des mises en oeuvre typiques des techniques de traitement par lot et par ré-ordonnancement ont été développées. Enfin, la théorie sur l'option de rejet a permi de dériver deux modèles pratiques pour faire la détection des cas ambigus.

Finalement, le chapitre cinq décrit la quatrième contribution. C'est la proposition d'un RNA à fusion "what-and-where," qui permet l'identification rapide du type d'émetteur radar associé à chaque train d'impulsions. L'architecture de ce RNA est constituée de trois sous-systèmes: un RNA classificateur, un sous-système de catégorisation en-ligne, et un sous-système d'accumulation de réponses. La séquence d'impulsions interceptées est partitionnée en deux séquences distinctes, qu'on nomme "what" et "where." Les paramètres "what" forment l'entrée pour le RNA classificateur qui prédit les types de radar associés aux impulsions. Entre temps, les paramètres "where" forment l'entrée pour le sous-système de catégorisation en ligne, qui sépare les impulsions transmises par différents émetteurs. Le sous-système d'accumulation de réponses permet de fusionner les réponses du RNA classificateur avec celles du sous-système de catégorisation. L'accumulation permet d'identifier les émetteurs d'après une séquence d'impulsions, pour améliorer la précision. Des simulations ont été effectués pour une mise-en-oeuvre particulière du RNA à fusion "what-and-where." Elle combine une variante du RNA fuzzy ARTMAP (pour faire la classification), et un algorithme qui exécute une association du type plus-proche-voisin et le filtrage de

Kalman (pour faire la catégorisation en-ligne), avec le sous-système d'accumulation. Encore une fois, l'ensemble de données radars qui a été utilisé lors de ces simulations a été collecté dans le champ par le CRDO.

Chapitre 1

Mesures de soutien électronique

1.1 Le système de MSE conventionnel

Le terme Mesures de Soutien Électronique (MSE) radar fait référence à la recherche, l'interception, la localisation et l'analyse de signaux radars dans un contexte de surveillance militaire [63] [121] [139]. Il existe plusieurs types de systèmes de MSE radar qui s'appliquent à différents contextes. À haut niveau, certaines propriétés fonctionnelles sont consistantes pour la plupart de ces systèmes. Cette section décrit la structure et le fonctionnement à haut niveau d'un système de MSE typique. (Il serait difficile de le décrire à plus bas niveau dans le cadre de cette thèse.) Cette brève description sert comme point de référence pour les travaux de la thèse.

Le diagramme bloc à la figure 1.1 présente l'organisation globale d'un système de MSE radar conventionnel. Ce système intercepte des signaux de l'environnement, puis

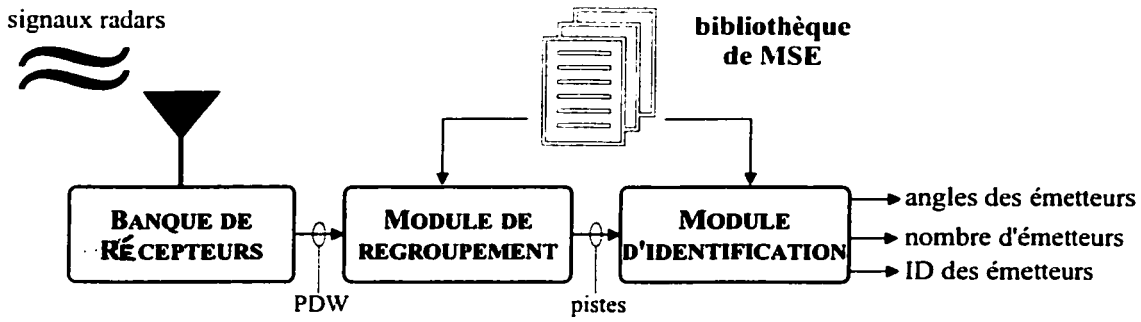


Figure 1.1: Diagramme bloc d'un système automatique pour le MSE radar.

il affiche (par l'entremise d'une interface homme-machine) les résultats d'une analyse de la quantité et des propriétés des émetteurs actifs. Il affiche aussi la ressemblance de propriétés des émetteurs à ceux de modèles radars qui sont connus. En général, la fonctionnalité d'un tel système peut être décomposée en trois tâches [40]: la réception des signaux radars, le regroupement des impulsions, et l'identification des types de radar.

1.1.1 Réception des signaux radars

Selon la figure 1.1, le système capte passivement les signaux radars d'un environnement à l'aide de la banque de récepteurs. Dans un environnement typique, les signaux radars interceptés représentent un mélange complexe d'impulsions[1] électromagnétiques qui sont transmises par plusieurs sources. On peut dire que le train d'impulsion qui est transmis par un émetteur est *entrelacé* avec celui des autres émetteurs actifs. Les signaux traités par le récepteur sont alors des séquences

[1]Dans cette thèse, on fait abstraction des composantes de type continue (communément appelée CW pour "continuous wave") qui peuvent exister dans les signaux radars interceptés.

d'impulsions presque aléatoires en apparence.

Lorsqu'une impulsion est détectée, un récepteur à large bande mesure la valeur de paramètres standards, tels que l'amplitude (PA pour "pulse amplitude"), la durée (PW pour "pulse width"), et le temps d'arrivée (TOA pour "time-of-arrival") de l'impulsion, et la fréquence de porteuse (RF pour "radio frequency"). Un récepteur à recherche de direction mesure aussi l'angle d'arrivée (Brg pour "bearing"), tandis qu'un récepteur avancé mesure aussi la modulation de l'impulsion (MOP pour "modulation on pulse"). Une fois que ces paramêtres ont été mesurés, ils sont quantifiés et concatenés pour former un mot binaire qu'on nomme le "Pulse Descriptor Word" (PDW).

1.1.2 Regroupement d'impulsions

La séquence de PDW générés par la banque de récepteurs est transmise au module de regroupement. Ce module cherche à répérer les trains d'impulsions individuels, ce qui implique un regroupement progressif des PDW qui semblent provenir d'un même émetteur[2]. Chaque groupe est associé à une piste. Une piste est constituée de paramètres statistiques du PDW (*e.g.*, la valeure moyenne du RF), et de paramètres dérivés à partir du groupe de PDWs (*e.g.*, l'intervalle de répétition d'impulsions ou bien PRI pour "pulse repetition interval"). Des plages de paramètres sont associées aux pistes. Elles sont mise-à-jour pour représenter l'évolution dans le temps des

[2]Notez que chaque émetteur représente une instance d'un type de radar, et puis chaque type de radar peut fonctionner sous plusieurs *modes* afin d'effectuer différentes tâches.

caractéristiques d'émetteurs. Alors, la formation de pistes permet de dériver certains paramètres pour l'identification de l'émetteur, et de réduire la densité des données qui sont traitées par le module d'identification.

Il y a trois classes de techniques classiques qui permettent d'effectuer le regroupement d'impulsions [31]:

1. **le désentrelacement selon le temps d'arrivée [40] [100] [139]:** Cette technique consiste à chercher des consistances dans le TOA de chaînes d'impulsions. Une chaîne est un ensemble de PDW qui ont un ou plusieurs paramètres très semblables (*e.g.*, opèrent à la même fréquence RF). L'analyse par histogramme du TOA est un exemple de méthode qui permet de découvrir des consistances dans le TOA d'une chaîne quelconque de PDW. Si on retrouve un patron consistant dans le TOA et que ce patron est correlé avec une des définitions qui est compilée dans la bibliothèque de MSE, alors les PDW correspondants sont regroupés sur la base du PRI.

2. **le triage en cellules [31] [40] [117] [124]:** Cette technique est basée sur l'utilisation de cellules qui sont addressables dans un sous-domaine de l'espace de paramètres comme RF, PW, et Brg. Un "Window Addressable Memory" (WAM) [124] est un exemple de méthode pour le triage par cellule. C'est un filtre numérique constitué d'un ensemble de comparateurs qui sont fixés sur des combinaisons de paramètres. Les PDW sont comparés avec l'étendue des paramètres de cellule, chacun correspondant aux paramètres plausibles

pour un émetteur. On assigne à chaque PDW une cellule active (identifiée antérieurement). Une nouvelle cellule est créée pour des PDW avec une combinaison trop différente de paramètres. Les cellules peuvent avoir une définition fixe [40] ou variable [31] [117].

3. **le triage par techniques métriques [3] [43] [45] [140]:** Avec cette technique, le regroupement est formulé comme un problème de catégorisation: une catégorie (*i.e.*, mode d'un émetteur actif) est assignée à chaque patron (*i.e.*, PDW). La version en-ligne de k-means [45] est un exemple de méthode pour effectuer le triage métrique d'une séquence de PDWs. Comparé au triage en cellules, les PDW sont regroupés sans aucune information a priori des émetteurs. La formation de catégories est basée principalement sur l'utilisation d'une mesure de proximité entre PDW dans l'espace de paramètres comme RF, PW, et Brg. On utilise souvent cette forme de triage avec des paramètres non-standards comme le MOP, afin de résoudre des environnements plus complexes.

Un module de regroupement peut exploiter simultannément une ou plusieurs techniques de regroupement. Un exemple d'architecture qui permet de réduire la latence pour identifier de nouveaux émetteurs est présentée à la figure 1.2 [118] [124] [143]. Sachant que certains PDW appartiennent à des émetteurs qui ont déjà été identifiés par des MSE, il est possible de réduire le débit, car le regroupement des PDW "connus" n'est pas nécessaire. L'architecture de la figure 1.2 permet au système de concentrer ses efforts sur les PDW "inconnus," ce qui permet de supporter des débits

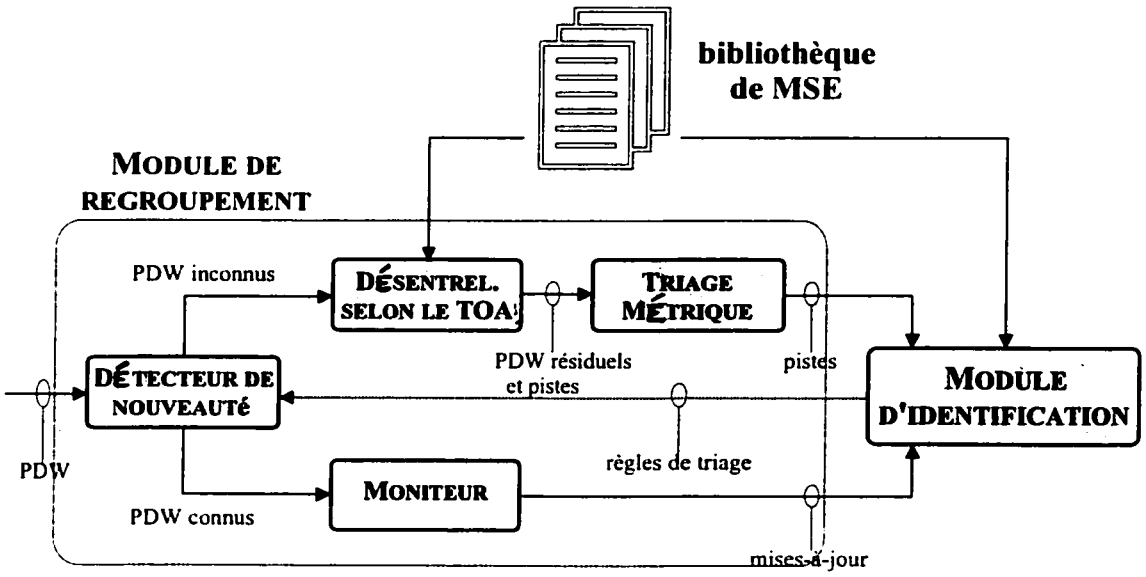


Figure 1.2: Structure interne d'un module de regroupement.

plus élevés et (possiblement) d'améliorer ses performances.

Dans la figure 1.2, un *détecteur de nouveauté* est réalisé par une technique de triage en cellules. Les PDW correspondants aux émetteurs déjà identifiés sont redirigés vers un *moniteur*. Ce dernier module suit l'évolution des émetteurs et fait la mise à jour de certains paramètres (*e.g.*, Brg). Les PDW "inconnus" sont regroupés par un processus en deux temps. Premièrement, un désentrelacement selon le TOA permet de regrouper les PDW avec un PRI qui est identifiable (selon la bibliothèque de MSE). Ensuite, le résidu du désentrelacement est traité par un triage par technique métrique. Celui-ci caractérise les pistes d'émissions plus complexes.

1.1.3 L'identification des types de radar

Le module d'identification analyse les pistes qui ont été formées par le module de regroupement. Une bibliothèque de MSE, qui contient les descriptions paramétriques de modèles radars connus, interagit avec le module d'identification. On tente alors d'assigner un type de radar dans la bibliothèque à chaque piste.

Dans une bibliothèque de MSE, l'étendue des paramètres qui décrivent les types de radar peut conduire à des chevauchements. Alors, le module d'identification produit souvent plusieurs types de radar pour une même piste. On suit souvent l'évolution d'une liste des types probables qui est générée avec le niveau de confiance des prédictions, la menace qu'ils posent, la dernière valeur du Brg, etc. Ces résultats sont ensuite transmis et affichés à l'interface homme-machine.

Un module d'analyse tactique de situation (pas inclus sur la figure 1.1) peut révéler des changements dans le mode des émetteurs, des liens entre émetteurs, ainsi que des plateformes dans l'environnement. Finalement, le système de MSE fournit la signalisation requise pour entamer des contre-mesures.

1.2 Problématique

L'efficacité globale d'un système de MSE radar conventionnel dépend des techniques employées pour le traitement des signaux, ainsi que la qualité de la bibliothèque de MSE. Peu importe le système spécifique, la complexité et la prolifération croissante

des signaux radars compliquent la tâche.

Dans le contexte actuel, les systèmes de MSE peuvent être confrontés à des environnements qui évoluent rapidement, avec un nombre croissant d'émetteurs. De plus en plus d'émetteurs transmettent des ondes à haute densité. Dans certaines bandes de fréquence, on peut s'attendre à intercepter jusqu'à 10^6 impulsions par seconde. L'augmentation de la densité des signaux radars implique une diminution du temps de réponse pour la reconnaissance d'émetteurs. Cependant, le temps de réponse pour un système de MSE est un facteur critique pour la prévention des menaces.

Les systèmes de MSE sont illuminés par une diversité croissante d'émetteurs, d'un même ou de plusieurs types différents, qui peuvent fonctionner avec plusieurs modes différents. Avec les avancements technologiques, l'agilité et le chevauchement de paramètres comme RF et PRI contribuent davantage à la dégradation du processus de regroupement. Ceci mène à des pistes qui sont mal caractérisées, et à des ambiguïtés dans l'identification de types.

Finalement, il est devenu difficile et coûteux de maintenir des bibliothèques de MSE qui peuvent refléter avec précision chaque environnement opérationnel. La construction d'une bibliothèque de MSE consiste à modéliser la distribution paramétrique de systèmes radars connus à partir de connaissances et de données a priori. Puisque certains types de radar deviennent difficiles à décrire, cette tâche s'avère plus complexe, et plus susceptible à l'erreur. De plus, la multiplication des modes d'un radar implique des descriptions plus complexes et volumineuses. Les bibliothèques ont alors

de plus en plus tendance à contenir des descriptions incomplètes ou erronnées.

Etant donné les tendances actuelles, les systèmes de MSE doivent tenter d'améliorer la précision, la vitesse et la robustesse de leurs traitements. La recherche d'approches alternatives pour la reconnaissance d'émetteurs radars est importante afin de prévenir les menaces futures.

1.3 Contributions de la thèse

Des nouvelles approches, qui s'inspirent de techniques en intelligence artificielle, comme les algorithmes génétiques, les systèmes experts, la fusion de données, la logique floue, et les réseaux de neurones artificiels, sont prometteuses pour la reconnaissance d'émetteurs [123]. À date, deux approches principales ont été appliquées au traitement en MSE radar: les systèmes experts et les réseaux de neurones artificiels.

Les systèmes experts, *i.e.*, les techniques à base de connaissances ("knowledge based"), effectuent l'association des pistes aux émetteurs (dans un module d'identification), et des émetteurs aux plateformes (dans un module d'analyse tactique de situation). Ces techniques permettent de combiner des sources supplémentaires d'information — plateformes présentes dans l'environnement, histoire de comportement de l'émetteur, etc. — aux connaissances standards lors de l'identification d'un émetteur. Elles peuvent représenter et combiner des informations acquises graduellement afin de réduire l'ambiguïté des décisions [2] [115] [116] [134] [137].

Pour leur part, les réseaux de neurones artificiels (RNA), sont utilisés pour ef-

fectuer le triage (dans un module de regroupement) et la classification (dans un module d'identification) d'impulsions. L'utilisation d'un RNA implique un traitement des signaux par une structure qui peut devenir massivement parallèle. Les connaissances distribuées d'un RNA sont acquises par un processus d'apprentissage d'exemplaires. Les RNA sont très bien adaptés pour résoudre plusieurs problèmes complexes en reconnaissance de formes [4] [79] [96] [101] [116] [124] [135].

Cette thèse de doctorat s'inscrit dans le cadre d'une étude sur le potentiel des RNA[3] pour faire la classification rapide d'impulsions radars dans les systèmes de MSE. Elle est organisée en deux volets et quatres contributions. Le premier volet (A) traite de l'application des RNA de type apprentissage non-supervisé (pour la catégorisation en-ligne) au triage métrique rapide de séquences continues d'impulsions radars. Le deuxième volet (B) traite de l'application des RNA de type apprentissage supervisé (pour la classification) à l'identification des types de radar. On suppose ici l'existence d'un ensemble de données pour entrainer un RNA supervisé.

En particulier, cette thèse comporte les quatre contributions suivantes:

1. la comparaison de RNA auto-organisateurs à apprentissage compétitif qui peuvent supporter le triage métrique rapide de séquence d'impulsions radars [55];

2. la proposition d'une architecture VLSI qui permet la mise en oeuvre du RNA fuzzy Adaptive Resonance Theory (ART) pour des applications de triage métrique

[3]La litérature concernant les RNA est abondante. Étant donnée la richesse du domaine de recherche, on cite seulement les ouvrages classiques de Christopher Bishop [11], de Simon Haykin [72] et de Jacek Zurada [145] comme références.

à haute vitesse [54];

3. la proposition d'une technique – nommée le *traitement par re-ordonnancement* – qui gère la manière dont les patrons d'entrée sont apprises dans un système de catégorisation, pour améliorer la qualité des résultats [62];

4. le développement d'un RNA classificateur à fusion "what-and-where" qui permet l'identification rapide du type d'émetteur radar [60].

Les trois premières contributions — 1, 2 et 3 — découlent du volet (A), tandis que la dernière contribution — 4 — découle du volet (B).

Les quatres prochains chapitres de cette thèse décrivent, respectivement, ces quatres contributions. Quatre articles publiés ou soumis dans des revues scientifiques décrivent les travaux liés à chacune des contributions. Chaque chapitre contient une mise en situation, un sommaire des travaux accomplis, l'article de revue correspondant et une synthèse des résultats. Étant donnée la nature d'une thèse par articles, afin d'éviter la redondance dans le contenu, la revue de litérature liée à chaque contribution est réservée aux chapitres correspondants.

Chapitre 2

Une comparaison de divers réseaux de neurones auto-organisateurs

Dans le premier volet de cette thèse, les RNA non-supervisés sont appliqués au triage (par technique métrique) d'impulsions radars selon leur émetteur. On suppose que ces impulsions arrivent à un débit élevé, et qu'elles doivent être traitées sans connaissance *a priori* des émetteurs actifs. Plus précisement, un RNA est requis pour effectuer l'apprentissage en ligne de catégories (*i.e.*, modes d'émetteurs) à partir d'une séquence continue de patrons (*i.e.*, impulsions radars).

Les RNA auto-organisateurs basés sur l'apprentissage compétitif [72] [145] sont très bien adaptés à ce type de problème. En effet, ils peuvent catégoriser des patrons d'entrée en ligne, sans information *a priori* sur le nombre et les caractéristiques des catégories à former. Leur apprentissage non-supervisé de patrons se fait séquentiellem-

ent, ce qui se traduit par une définition adaptative des dimensions, du nombre et du placement des catégories. La représentation des catégories par des prototypes signifie qu'on peut éviter le stockage à long-terme des patrons à catégoriser. Finalement, ces algorithmes se prètent bien à des mises en oeuvres parallèles, ce qui favorise le traitement en temps réel rapide.

Ce chapitre présente la première contribution du premier volet — la comparaison entre RNA auto-organisateurs de type apprentissage compétitif qui peuvent supporter le triage métrique rapide de séquences d'impulsions, sans connaissance a priori. Une étude approfondie de ce type de RNA a permi d'isoler quatre réseaux — le "Fuzzy Adaptive Resonance Theory" (FA) [21], le "Fuzzy Min-Max Clustering" (FMMC) [126], le "Integrated Adaptive Fuzzy Clustering" (IAFC) [84] et le "Self-Organizing Feature Mapping" (SOFM) [86] — qui sont prometteurs.

En MSE radar, un système idéal pour le triage métrique rapide se sert d'un algorithme efficace pour produire des catégorisations très précises. Le temps de réponse du système de catégorisation est aussi pertinent que la précision des résultats. Dans la comparaison entreprise, la performance des quatres RNA a donc été examinée sous trois angles différents — la qualité des catégorisations, le temps de convergence et la complexité de calcul. Les deux derniers angles donnent une idée de l'effort de calcul qui est requis pour obtenir une certaine qualité de catégorisation. La qualité des catégorisations et le temps de convergence ont été déduits à partir de simulations, tandis que la complexité de calcul a été estimée par le temps de pire-cas de traite-

ment d'un patron. La qualité des catégorisations a été mesurée avec les mesures de similarité *Rand Adjusted* [76] et celle de Jaccard [43]. Le temps de convergence a été mesuré par le nombre de présentations des données avant que les poids synaptiques ne se stabilisent.

L'ensemble de données pour les simulations est constitué d'impulsions radars recueillies[1] dans le champ par le Centre de Recherche pour la Défense d'Ottawa. Avant chaque simulation, les patrons de l'ensemble ont été organisés selon un des trois ordres de présentation - un ordre aléatoire, et deux ordres représentatifs d'un environment radar. Ceci a permi d'observer la variabilité des résultats de catégorisation face à l'ordre de présentation statistique. Les résultats de plusieurs simulations ont été combinés pour donner des valeurs moyennes pour la qualité de catégorisation et le temps de convergence. Les résultats de simulation, ainsi que les estimations de complexité ont été analysées pour des applications cibles (le triage rapide en MSE radar).

Des détails plus extensifs sur cette comparaison sont exposés dans l'article suivant:

GRANGER, E., SAVARIA, Y., LAVOIE, P., et CANTIN, M.-A.,

"A comparison of self-organizing neural networks for fast clustering of radar pulses,"

Signal Processing, **64:3**, 249-269 (1998).

Une copie de cet article est reproduite ici. Afin d'améliorer la compréhension, l'énoncé spécifique des algorithmes utilisés lors de la comparaison a été ajouté comme annexe C

[1] Cet ensemble représente des impulsions avec paramêtres de type MOP qui peuvent êtres utilisés en MSE radar pour le triage métrique sans information à priori.

de l'article en question. Ensuite, la dernière section du chapitre aborde une discussion sur l'impact des résultats de cette contribution.

A comparison of self-organizing neural networks for fast clustering of radar pulses

Eric Granger[1], Yvon Savaria[1], Pierre Lavoie[2] and Marc-André Cantin[3]

[1] Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C. P. 6079, Station "centre-ville," Montreal, Quebec, H3C 3A7, Canada.

[2] Defence Research Establishment Ottawa, Department of National Defence, 3701 Carling Ave., Ottawa, Ontario, K1A 0Z4, Canada.

[3] Department of Computer Science, Université du Québec à Montréal, C. P. 8888, Station A, Montreal, Quebec, H3C 3P8, Canada.

Abstract

Four self-organizing neural networks are compared for automatic deinterleaving of radar pulse streams in electronic warfare systems. The neural networks are the Fuzzy Adaptive Resonance Theory, Fuzzy Min-Max Clustering, Integrated Adaptive Fuzzy Clustering, and Self-Organizing Feature Mapping. Given the need for a clustering procedure that offers both accurate results and computational efficiency, these four networks are examined from three perspectives — clustering quality, convergence time, and computational complexity. The clustering quality and convergence time are measured via computer simulation, using a set of radar pulses collected in the

field. The effect of the pattern presentation order is analyzed by presenting the data not just in random order, but also in radar-like orders called burst and interleaved. Estimation of the worst-case running time for each network allows for the assessment of computational complexity.

2.1 Introduction

The purpose of radar electronic support measures (ESM) is to search for, intercept, locate, and analyze radar signals in the context of military surveillance [40] [63] [121]. When a radar ESM system is illuminated by several radars, it typically relies on pulse repetition interval (PRI) parameters to deinterleave the intercepted pulse trains. Unfortunately, the multiplication of intricate PRI patterns in about every radar model has spurred an inordinate complexity in deinterleaving algorithms. Alternative approaches for clustering radar pulses are thus needed, ones that rely on direction of arrival, frequency, pulse width and other such parameters that can be obtained from individual pulses. The selection of parameters can vary greatly from one ESM system to the next due to installation, cost, size, tactical and other considerations. Yet, beyond parameter choice, clustering the intercepted radar pulses constitutes a challenging problem. This clustering must sustain a very high data rate since, in certain radar bands, the signal densities encountered can reach up to 10^6 radar pulses per second.

Self-organizing neural networks (SONNs) appear very promising for this type of

clustering application, since they can cluster patterns autonomously, and lend themselves well to very high speed, parallel implementation [4] [79] [87]. Several innovative SONNs have been reported in the literature, each one demonstrating unique and interesting features. This paper presents a comparative study of four of them, all potentially suitable for solving very high throughput clustering problems. They are the Fuzzy Adaptive Resonance Theory [21], Fuzzy Min-Max Clustering [126], Integrated Adaptive Fuzzy Clustering [84], and Self-Organizing Feature Mapping [86].

The performance of these four neural networks is examined from three points of view — clustering quality, convergence time, and computational complexity. Indeed, in many practical applications, the accuracy of clustering results, and the computational efficiency of the clustering procedure are equally important. In this paper, *clustering quality* refers to the degree of similarity between the partitions (clusters) produced by a SONN, and a reference partition based on known category labels. This similarity is assessed by applying the Rand Adjusted [76] and the Jaccard [43] measures to partitions obtained by computer simulation. *Convergence time* is defined as the number of successive presentations of a finite input data set needed for a SONN's weight set to stabilize. This time is easily determined from computer simulation. *Computational complexity* is estimated from the maximum execution time required by a SONN algorithm to process one input pattern. In order to estimate this worst-case running time, we assume that the algorithm is implemented as a computer program running on an idealized random access machine (RAM) [37].

The main data set used in our simulations describes electromagnetic pulses transmitted by shipborne navigation radars. These pulses were collected from ashore by the Defence Research Establishment Ottawa (DREO) using a directional antenna, a superheterodyne tuner, a high-accuracy I/Q demodulator [90], and a two-channel, 10-bit digital oscilloscope. The radars were observed one at a time to ensure that each file contained pulses from a single radar. The identity of each ship was obtained from a harbor control tower and used to label each file. These labels can serve as references to measure the effectiveness of clustering techniques for the application.

In this study, the outcome of numerous computer simulations are combined to yield an average similarity measure and convergence time. In addition to the Radar data set described above, two other sets, called Wine Recognition and Iris, are used for comparison. For each set, the patterns are presented in three statistically different orders: random, by burst, or interleaved in a radar-like manner. The data sets and presentation orders were selected to illustrate the commonalities and differences between the four neural networks. Simulation results and complexity estimates are analyzed with very high data rate clustering applications in mind.

The rest of this paper is organized as follows. In the next section, the main features of the four SONNs selected for this study are briefly outlined. In Section 2.3, the methodology used to compare these SONNs (namely, the performance measures, data sets, and data presentation orders) is described. Finally, the results are analyzed and discussed.

2.2 Self-organizing neural networks

A clustering method suitable for radar ESM should have the following properties. First, it should not require prior knowledge of the number or characteristics of categories to be formed. Second, since the variable input arrival rate may reach 10^6 patterns per second, it should be able to cluster non-stationary streams of input patterns sequentially, without requiring their long-term storage. Lastly, the sequence of operations needed for implementing the method using current technology should lend itself well to high speed hardware realizations. Given that most of the popular, well-established classical [3] [43] [45] [130] and fuzzy [9] [10] clustering algorithms require prior knowledge on either the number or the characteristics of clusters sought, several iterations with the whole data set, or storage of the entire data set in memory, none of them were considered for this comparison.

The unsupervised learning paradigm used in self-organizing neural networks (SONNs) is related to clustering, since it permits the assignment of adaptively defined categories to unlabeled patterns [87] [72]. SONNs appear promising for high data rate sequential clustering applications, since they can cluster patterns autonomously, in most cases without prior knowledge of the number of categories. Moreover, they do not require long-term storage of the input patterns, and permit adaptive determination of the shape, size, number, and placement of categories, while operating in parallel [126]. Self-organizing "neuro-fuzzy" networks have recently been developed, where fuzzy logic concepts are integrated into the SONN framework. Categories are

then modeled as fuzzy sets that allow encoding the input scene's vagueness [91].

An important family of SONNs is derived from the basic idea of competitive learning [65] [66] [99], a type of unsupervised learning. In short, competitive learning neural networks seek to determine decision regions in the input pattern space. The most elementary of these networks consists of a single layer of identical output neurons, each one fully connected to the input nodes with feedforward excitory weights, which encode the categories learned by the network. A category label is associated with each output neuron, whose features are represented by the numerical values of the set of weights connecting it to all input nodes. When a pattern is presented to the network's input nodes, it is propagated through the weights to output neurons, which enter a winner-take-all competition. The one with the strongest activation (the winner) is allowed to adapt its set of weights to incorporate the input's novel characteristics. Through this process, output neurons become selectively tuned to respond differently to given input patterns, and therefore learn to specialize for regions of the input space: they become feature detectors [119].

Four SONNs that are based on competitive learning were selected for this study: Fuzzy Adaptive Resonance Theory (FA) [21], Fuzzy Min-Max Clustering (FMMC) [126], Integrated Adaptive Fuzzy Clustering (IAFC) [84], and Self-Organizing Feature Mapping (SOFM) [86]. Although the classical ISODATA clustering method [5] [43] [45] [130] does not offer a practical solution to the problem (since it requires significant prior knowledge of the data), it is included as a reference point for clustering quality com-

parison, given its widespread use in statistical multivariate data analysis.

2.2.1 Overview of the four neural networks

The first three SONNs (FA, FMMC and IAFC) subscribe to the basic Adaptive Resonance Theory (ART) [20] control structure. Essentially, ART networks categorize familiar inputs by adjusting previously learned categories, and create new categories dynamically in response to inputs different enough from those previously seen. A vigilance parameter ρ regulates the maximum tolerable difference between any two input patterns in a same category. Each one of these three SONNs integrates fuzzy logic concepts into the binary input ART1 [20] neural network processing framework.

Structurally, the FA neural network, proposed by Carpenter, Grossberg and Rosen [21], consists of two layers of neurons that are fully connected: an input layer $F1$ with $2M$ neurons (two per input feature) and an output layer $F2$ with N neurons (one per category). An adaptive weight value w_{ji}, represented as a real in the interval [0,1], is associated with each connection. The indices i and j denote the neurons that belong to the layers F1 and $F2$ respectively. For each neuron j of F2, the category prototype vector $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{j2M})$ contains the set of characteristics defining the category j. When complement coding is used, this vector may be interpreted as a hyperrectangle in the M-dimensional input space.

The FMMC neural network, proposed by Simpson [126], consists of 2 layers of neurons: an input layer $F1$ with M neurons (one per input pattern feature), and an

output layer $F2$ with N neurons (one per category). Two weighted connections link every $F1$ neuron to an $F2$ neuron. For the $F2$ output neuron j ($j = 1, 2, ..., N$), the 2 sets of M connected weights encode a min ($\mathbf{u}_j = (u_{j1}, u_{j2}, ..., u_{jM})$) and a max ($\mathbf{v}_j = (v_{j1}, v_{j2}, ..., v_{jM})$) point. These two points define a hyperrectangle fuzzy set B_j, the network's representation of category j.

FA and FMMC are both fuzzy min-max clustering networks, whereby category prototype vectors are represented as fuzzy set hyperrectangles in the M-dimensional input space, each one entirely defined by a min and a max point [82]. Both accomplish fuzzification of the ART1 network by essentially replacing crisp logic AND operators with fuzzy logic AND (min) operators. Despite a superficial resemblance to FA, FMMC is a somewhat different network, yielding categories associated to hyperrectangles that never overlap one another. A contraction procedure is used to prevent the occurrence of hyperrectangle overlap.

IAFC is a fuzzy clustering algorithm proposed by Kim and Mitra [83] [84]. As with FMMC, IAFC contains an M neuron input $F1$ layer (one per feature), and an N neuron output $F2$ layer (one per category). These neurons are fully interconnected by bottom-up and top-down weights. The categories formed, however, are hyperspherical in shape and are represented as centroids in the M-dimensional input space. Two prototype vectors, $\mathbf{v}_j = (v_{j1}, v_{j2}, ..., v_{jM})$ and $\mathbf{b}_j = (b_{j1}, b_{j2}, ..., b_{jM})$, are associated with category j's centroid. The network's top-down weights v_{ji} encode the actual cluster centroids, whereas each bottom-up weight b_{ji} is a normalized mirror of v_{ji}, for

$i = 1, 2, ..., M$ and $j = 1, 2, ..., N$.

The objective of IAFC is to obtain improved nonlinear decision boundaries among closely located cluster centroids by introducing a new vigilance criterion and a new learning rule into ART1 processing. The new vigilance criterion combines a fuzzy membership value and the Euclidean distance to form more flexible categories. The new learning rule uses a fuzzy membership value, an intracluster membership value, and a function depending on the number of data set iterations with a Kohonen-type learning rule.

For our purposes, Kohonen's Self-Organizing Feature Mapping (SOFM) [85] [86] defines a mapping from the M-dimensional input space onto a two-dimensional grid of output neurons, whose size is user-defined. The SOFM method creates a vector quantizer within the neural structure by adjusting the weights of connections linking the M input neurons of the $F1$ layer to all the N' output $F2$ map neurons. A prototype vector $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{jM})$ is associated with every one of these map neurons. When an input pattern is presented, each map neuron computes the Euclidean distance of its prototype vector to the input. The closest prototype corresponds to the best-match map neuron. The weights connected to this neuron, as well as those of the other neurons in its neighborhood are adjusted to learn from the input. If the lateral map distance from the best-match neuron to the location of another one falls within the perimeter defined by the *neighborhood radius*, then this other neuron belongs to the neighborhood. Throughout learning, the neighborhood's radius, $r(t)$,

and the learning rate, $\alpha(t)$, are progressively decreased in a linear way from their initial values, $r(0)$ and $\alpha(0)$, until they become equal to 1 and 0, respectively.

As learning evolves, weights become organized such that topographically close map neurons are selectively tuned to inputs that are physically similar. SOFM clustering is characterized by the formation of a topographic feature map in which the spatial coordinates of its neurons correspond to intrinsic features of the input patterns. Therefore, SOFM can be seen as a network producing an unsupervised nonlinear projection of the probability density function of the high-dimensional input data onto a two-dimensional display [86].

2.2.2 Modifications for radar pulse clustering

Neither one of the four networks in its original form is suitable for sequential clustering of radar pulses in practical ESM systems. The aim of this subsection is to propose appropriate modifications.

FA, FMMC, and IAFC are ART-type networks, whereby output neurons are selected using a potentially demanding search process. The computational cost of the iterative category choice and corresponding expansion testing, through N output neurons, is $O(N^2)$. However, performing the category expansion test on all N output neurons, prior to a direct category choice, can significantly accelerate processing in cases where several output neurons would fail the expansion test. The computational cost of the resulting search process is $O(N)$.

The plasticity of ART-type networks like FA and FMMC has been well documented. When an input resembles none of the stored prototypes, an output neuron is committed, which amounts to creating a new category with the input as its prototype. For radar ESM applications, this implies that these networks can learn a new category from a single, or very few radar pulses. One drawback is that they can produce artifact categories, and thus possibly trigger false alarms, if corrupt pulses are encountered. In practice, some means of managing such false alarms would be required. Notice that IAFC is also an ART-type network, where plasticity is controlled by a learning rate parameter, λ, which varies with the number of data set iterations, and with the mismatch between inputs and winning category prototypes.

Since the environment around an ESM system changes over time as radars come and go, categories that have not been activated for a long time must be freed. In this paper, the data sets are sufficiently small that we need not bother with category removal. In practice however, category reuse would be necessary for maintaining an up-to-date representation of the environment, for accuracy, and for making efficient use of computing resources.

SOFM, and to some extent IAFC, permit stable learning of categories by progressively decreasing plasticity, that is, by reducing the influence of subsequent inputs on existing weights. Plasticity is controlled by the parameters $r(t)$ and $\alpha(t)$ in SOFM, and $\lambda(l)$ in IAFC. In both cases, the parameter values are gradually decreased over time, and the networks eventually loose their ability to react to new information.

This approach is acceptable for batch data processing like in this paper. In practice, however, new radars can appear in the theater of operation at any time, and both IAFC and SOFM would require modification to learn continuously.

In FA, FMMC, and IAFC, input patterns are regrouped into a *variable* number of categories, each one represented by a prototype vector associated with a single output neuron. By contrast, SOFM clusters inputs into a *fixed* number of output neurons, organized into a 2-dimensional topographic map. During training, the network adapts the prototype vectors for all its map neurons, and assigns an actual data cluster to a region in the map, which may contain one or more map neurons. This process raises the problem of locating categories in the map. Besides, it can be difficult to specify a feature map size when the potential solutions have a relatively small number (2 or 3) of clusters [104]. A number of interpretation procedures have been proposed to assist SOFM for clustering [79] [104]. In order to ease the interpretation of the feature maps in this paper, and to permit a fair comparison with the other ART-type SONNs, an additional mechanism was incorporated into SOFM's processing. This mechanism is activated after the initial topographical ordering phase has ended, when the map is being fine-tuned for statistical accuracy. Define a *category center* as a map neuron which has been assigned to more than ζ input patterns. All the other map neurons can be related to the category centers using a minimum Euclidean distance criterion. Thus, when an input pattern selects one of these other neurons as the best-match, the label of the category center neuron with the closest prototype vector is output

by the modified network. This modification makes it possible to locate and count categories, whose number may vary as learning unfolds.

2.3 Comparison method

The methodology used is based on three different performance measures — clustering quality, convergence time, and computational complexity. This methodology is different from that of other comparisons reported in literature (for example, [81] [82] [108] [109] [136] [138]), in which clustering speed and response time are less critical than in our application. These measures are defined in this section.

2.3.1 Clustering quality

A partition of n patterns into K groups defines a *clustering*. This can be represented as a set $A = \{a_1, a_2, ..., a_n\}$, where $a_h \in \{1, 2, ..., K\}$ is the category label assigned to pattern h. The degree of match between two clusterings, say A and B, may be compared by constructing a contingency table, as shown in Table 2.1. In this figure, c_{11} (c_{22}) is the number of pattern pairs that are in a same (different) cluster in both partitions. The value c_{21} is the number of pattern pairs that are placed in a same cluster by A, but in different clusters by B. The value c_{12} reflects the converse situation. The sum of all elements $m = c_{11} + c_{12} + c_{21} + c_{22} = n(n-1)/2$ is the total number of combinations of two out of n patterns. The four variables within the contingency table have been used to derive measures of similarity between two clus-

Tableau 2.1: Contingency table used to compare two clusterings.

		Clustering A	
		same	**different**
Clustering B	**same**	c_{11}	c_{12}
	different	c_{21}	c_{22}

terings A and B [3] [43]. These measures are known in pattern recognition literature as external criterion indices, and are used for evaluating the capacity to recover true cluster structure. Based on a previous comparison of these similarity measures [102], the Rand Adjusted [76], defined by:

$$S_{RA}(A,B) = \frac{2(c_{11}c_{22} - c_{12}c_{21})}{2c_{11}c_{22} + (c_{11}+c_{22})(c_{12}+c_{21}) + c_{12}^2 + c_{21}^2} \tag{2.1}$$

and Jaccard statistic [43], defined by:

$$S_J(A,B) = \frac{c_{11}}{c_{11}+c_{12}+c_{21}} \tag{2.2}$$

have been selected to assess clustering quality for this study. It is worth noting that variable c_{22} does not appear in $S_J(A,B)$.

Since correct classification results are known for the data sets used, their patterns

are all accompanied by category labels. These labels are withheld from the SONN under test, but they provide a reference clustering, R, with which a clustering produced by computer simulation, A, may be compared. Then, variables c_{11} and c_{22} in Table 2.1 represent the number of pattern pairs which are properly clustered together and apart, respectively, while c_{12} and c_{21} indicate the improperly clustered pairs. In this case, Eqs. 2.1 and 2.2 yield *scores* that describe the quality of the clustering produced by a SONN. Both the Rand Adjusted and Jaccard measures yield a score ranging from 0 to 1, where 0 denotes maximum dissimilarity, and 1 denotes equivalence. The closer a clustering A is to R, the closer the scores are to 1. Notice the dependence of these scores on the number of clusters in A and R.

2.3.2 Convergence time

During a computer simulation, each complete presentation of an input data set to a SONN is called an *epoch*. Convergence time is conveniently measured by counting the number of epochs needed for a SONN to converge. Once convergence is reached, weight values remain constant during subsequent presentations of the entire data set in any order. This measure is independent from computational complexity (as defined in the following subsection), since an algorithm may require several epochs to converge using very simple processing, or vice-versa. The product of the two measures provides useful insight into the amount of processing required by each SONN to produce its best asymptotic clustering quality, while sustaining a desired clustering rate.

2.3.3 Computational complexity

A first order approximation of the computational complexity for the SONN algorithms may be obtained by assessing their execution time on an idealized computer. Thus, the time complexity, T, combined with a fixed computing area C, allows for comparison of area-time complexities, CT. To that effect, assume that the SONN algorithms are implemented as computer programs running on a generic, single processor, random access machine (RAM) [37], where instructions are executed one after the other. This generic machine is capable of no more than one operation per cycle (*i.e.*, neither VLIW or superscalar). Using the RAM model avoids the challenging task of accurately determining the performance of specific VLIW or superscalar machines, which is beyond the scope of this analysis.

Time complexity can be estimated from the maximum execution time required to process a single input pattern. The result is a total worst-case running time formula, T, which summarizes the behavior of a SONN algorithm as a function of two key parameters: the dimensionality of the input patterns, M, and the number of $F2$ output neurons, N. Specifically, T can be defined as the sum of the worst-case running times T_p for each operation p that is required to process an input [37]:

$$T = \sum_p T_p = \sum_p o_p \cdot n_p \tag{2.3}$$

where o_p is the constant amount of time needed to execute an operation p, and n_p is

the number of times this operation is executed.

For simplicity, we assume that p can take one out of two values, 1 or 2, where o_1 is the time required to execute an elementary operation such as: $x+y$, $x-y$, $\max(x,y)$, $\min(x,y)$, $1-x$, $x<y$, etc., and o_2 is the time needed to compute a division x/y, a multiplication $x \cdot y$, a square root $\sqrt{x}$, or an exponent e^x. In addition, x and y are assumed to be real numbers represented by an integer with a b bit resolution, where b corresponds to the number of bits needed to represent each SONN's elementary values (*i.e.*, synaptic weights, input pattern elements, and parameters) with a sufficient precision. Operations of type $p=2$ are more complex than those of type $p=1$, and their complexity, which is a function of b, depends on their specific implementation. Nevertheless, to simplify the analysis, we presume that $o_2 \simeq F(b) \cdot o_1 = F \cdot o_1$, where F remains as an explicit parameter in the complexity formulas.

2.3.4 Data set

The data gathered by DREO consists of 800 radar pulses from 12 different ship-borne navigation radars. As with any field trial, the recorded signals show imperfections in the radar transmitters, and exhibit distortion and noise due to the propagation channel and the collection equipment. For instance, the signal-to-noise ratio varies greatly from pulse to pulse, and from file to file (from 15 to 45 dB). This is due to the circular scan of the radars, and their varied power and distance from the collection site. Also, one file contains pulses with unusual characteristics indicative of

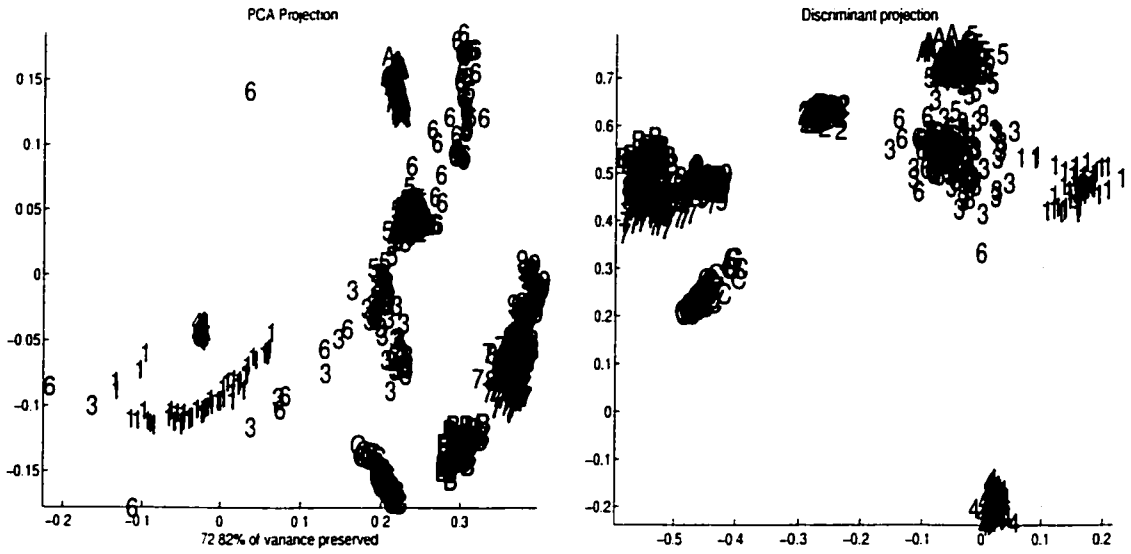
a radar operating with a defective magnetron tube. These pulses were not removed, for they reflect anomalies sometimes encountered in the field.

After the trial, DREO reduced the dimensionality of the data to simplify computer simulation and eventual implementation of the clusterer. Thus, 16 real-valued features were extracted from each radar pulse. The feature extraction algorithm has its own limitations, and produced a few outliers, which were not removed. The final data set was then normalized using a linear transformation so that values in every dimension range between 0 and 1.

Note that special attention was paid to ensure that clustering the data set would be difficult enough to fully exercise the algorithms. One should therefore focus on their relative, rather than absolute, performance.

The number of pulses per file is nominally 50, but two files contain 100 and 200 pulses, respectively. An uneven number of pulses per radar is typical of the application. Indeed, some radars can transmit in excess of 300000 pulses/second in their pulse Doppler mode, and even a few thousand pulses/second in their medium PRI mode. The small number of pulses per file in the data set is a consequence of the collection equipment capabilities. Nevertheless, a train of 50 pulses corresponds to a few beam illuminations by a radar in medium PRI mode, and should be sufficient for ESM detection.

Figure 2.1 shows the result of a two-dimensional principal component analysis (PCA) and linear discriminant analysis (LDA) projections for the data set. Although



(a) Principle component analysis (PCA) (b) Linear discriminant analysis (LDA)

Figure 2.1: Two-dimensional projections for the Radar data set.

both PCA and LDA are linear projection methods, LDA benefits from *a priori* cluster label information. Clearly, the clusters cannot be described by the same statistics. Some are not Gaussian distributed, and they sometimes overlap one another in 2D.

The Radar set described above represents data from a stream of pulses collected for one realistic scenario, under which a SONN would be called upon for automatic deinterleaving. Even though the data set is representative, it is far from comprehensive: there are several types of radar ESM systems, numerous different theaters of operation, and an ever growing variety of radar sources. Consequently, two standard data sets, called Wine Recognition and Iris, were used to broaden the scope of this research. A brief description, and the simulation results obtained on these sets can be found in Appendix A.

2.3.5 Data presentation order

Since sequential clustering results depend on the order of presentation of the patterns, three types of presentation orders are considered in this comparison.

A *random* presentation order consists of random permutations of the input data set patterns, regardless of the class.

A *burst* presentation order is defined by a random sequence of groups of patterns called bursts, where each burst contains patterns from the same class. The specific patterns which form the burst, and the class associated with the burst are both selected at random (from classes with remaining patterns). The burst size is a random variable with a normal distribution, whose average is equal to 5 and whose standard deviation is equal to 3. Bursts of negative size are ignored.

A radar-like *interleaved* presentation order is obtained as follows. First, the patterns in each class are permuted at random. Then, four fictitious radar parameters are chosen at random for each class. Initial scan time (IS) and scan period (SP) are chosen from a uniform distribution between 0 and 1. Beamwidth (BW) and pulse repetition interval (PRI) are selected from uniform distributions between 0 and $0.25 \cdot SP$, and between 0 and $0.25 \cdot BW$, respectively. These parameters allow the computation of a time of arrival (TOA) for every pattern of each class. Sorting the patterns by TOA yields an interleaved, radar-like, order of presentation.

2.4 Comparison results

2.4.1 Clustering quality and convergence time

For simulation purposes, MATLAB programs were written for FA, FMMC, IAFC, and ISODATA algorithms. SOFM simulations were done using the "SOM_PAK" software[2]. Prior to each simulation, the normalized input patterns of the data set were organized according to one of the three statistical orders (random, burst or interleaved). The sequence of patterns was then repeatedly presented to the algorithms under test until weights remained stable for two successive epochs, yielding the clustering A. The data set presentation order was kept constant from one epoch to the next. After the simulation, the Rand Adjusted and Jaccard scores were computed, using the reference clustering, R, available for the data. The scores were stored along with the number of epochs needed for convergence. The results of 20 distinct simulations were combined to extract central tendencies, and to determine the effect of presenting the input patterns in different orders.

Simulation results for the Radar data set are given in Table 2.4.1. (Similar results obtained using the Wine Recognition, and Iris data are given in Appendix A.) The table contains the mean and standard deviation of the best scores that can be obtained from the four SONNs and ISODATA for all three statistical presentation orders. Convergence time, and parameter values[3] as obtained by trial and error are also

[2]This software was obtained via anonymous FTP from cochlea.hut.fi.

[3]The reader is referred to the notation used for parameters in the original papers describing

shown. These parameter values always reflect the least amount of processing needed to produce the best clustering score.

For Table 2.4.1, the Rand Adjusted and Jaccard scores appear to describe similar trends. ISODATA confirmed our expectations by yielding the highest scores for all but one simulation, and thus offers a realistic performance target for the SONNs.

SOFM is the SONN that achieves the best overall scores, or rather score, since this network produces consistent clusterings that are the same across all presentation orders. This is a direct consequence of its slow convergence, combined with the map interpretation post-processing. This excellent score ($S_{RA}(A, R) = 0.95$) is comparable to those of ISODATA, but it is obtained after no less than 50 epochs. Remember that convergence speed is under user control, and it was set to yield the best possible score.

FA usually scores second best, achieving results that are significantly better with the interleaved presentation order ($S_{RA}(A, R) = 0.80$) than with either the random ($S_{RA}(A, R) = 0.61$) or the burst ($S_{RA}(A, R) = 0.59$) presentation orders. These scores are significantly lower than those of SOFM and ISODATA, but they are obtained after only 3 to 4 epochs, indicating that it is a very responsive network.

ISODATA, FA, FMMC, IAFC, and SOFM.

Tableau 2.2: Summary of best simulation results for the Radar data set.

ALGORITHM: (parameter values) · *presentation order*	**PERFORMANCE MEASURES**				
	Clustering quality				**Convergence time**
	$S_{RA}(A,R)$		$S_J(A,R)$		Number of epochs
	Mean	Std. dev.	Mean	Std. dev.	
ISODATA: ($2 \leq L \leq 4; \forall N_c; .1 \leq \theta_c \leq .2; .1 \leq \theta_s \leq .5; 14 \leq \theta_N \leq 22$)					
· *random*	0.93	0.122	0.99	0.043	$I \geq 15$
· *burst*	0.96	0.124	0.99	0.041	$I \geq 15$
· *interleaved*	0.96	0.163	0.99	0.062	$I \geq 15$
Fuzzy ART (FA): ($.85 \leq \rho \leq .95; \beta = 1, \alpha = 1$)					
· *random*	0.61	0.056	0.93	0.041	3.8
· *burst*	0.59	0.109	0.93	0.049	3.2
· *interleaved*	0.80	0.130	0.96	0.050	3.6
Fuzzy Min-Max Clustering (FMMC): ($\gamma = 5; .01 \leq \theta \leq .25$)					
· *random*	0.45	0.062	0.92	0.038	2.2
· *burst*	0.67	0.064	0.94	0.062	2.0
· *interleaved*	0.72	0.099	0.95	0.067	2.0
Integrated Adaptive Fuzzy Clustering (IAFC): ($\gamma = 1$; $m = 2$; $0 \leq \sigma \leq .6$; $.1 \leq k \leq 1$; $.5 \leq \tau \leq .6$)					
· *random*	0.62	0.002	0.88	0.001	17.6
· *burst*	0.63	0.060	0.89	0.025	20.9
· *interleaved*	0.66	0.046	0.90	0.021	21.2
Self-Organizing Feature Mapping (SOFM): (5×5 map; $2 \leq \zeta \leq 16; .03 \leq \alpha(0) \leq .5; 2 \leq r(0) \leq 3$)					
· *random*	0.95	0	0.99	0	50
· *burst*	0.95	0	0.99	0	50
· *interleaved*	0.95	0	0.99	0	50

FMMC converges even faster than FA, and this may explain that it yields mixed results. Noteworthy is the low Rand Adjusted score ($S_{RA}(A, R) = 0.45$) obtained for the random order of presentation, in contrast to the corresponding Jaccard score ($S_J(A, R) = 0.92$), which is not as bad. Scores for the burst and interleaved orders are comparable to those of FA.

IAFC takes from about 17 to 22 epochs to converge, but yields scores that are no better than those of FA and FMMC. Nonetheless, the standard deviation of the scores is low, meaning that the results obtained are consistent from one simulation to the next.

An attempt was made to improve FA scores by means of a post-processing mechanism somewhat similar to SOFM's map interpretation procedure. Unfortunately, this did not yield higher overall scores, and thus the results are not shown in Table 2.4.1. It did, however, allow good scores to be obtained over a broader range of vigilance parameter values.

The results in Table 2.4.1 clearly indicate that the data presentation order has a significant impact on performance. With the exception of SOFM, the algorithms usually perform better as the input presentation order moves from the completely random case to the more structured ones, where patterns from a category may be presented together. The standard deviation of the scores are also dependent on the data presentation order. For instance, with FA and FMMC, the scores often vary more for the interleaved than for the burst, and more for the burst than for the random

orders. In most cases, high scores appear to be correlated with high variance for the burst and interleaved presentation orders. With these orders, sequences are defined by mixtures of bursts of patterns from the same cluster. If we accept that scores depend on the size of the bursts, then the randomly defined burst sizes may explain the high variance of the scores. Burst sizes have no notable effect on convergence time.

2.4.2 Computational complexity

For brevity, only the final results of our analysis are shown here. The reader is referred to Appendix B for further details. Table 2.3 shows the total worst-case running times per input pattern in a normalized format (T/o_1), when parameter F is the same for all operations of type $p = 2$. The corresponding growth rates, valid when the parameters M and $N \gg 1$, and the parameter F is constant, are also given. It turns out that the complexity of all SONNs is described by the same asymptotic growth rate, $O(NM)$.

Tableau 2.3: Summary of the computational complexity estimations

SONN:	COMPUTATIONAL COMPLEXITY	
	Worst-case running time (T/o_1)	**Growth rate**
FA:	$6NM + NF + 4MF + 3N + 5M$	$O(NM)$
FMMC:	$3NMF + 25NM + 2NF - MF + 2N - 11M$	$O(NM)$
IAFC:	$2NMF + NM + 7NF + 2MF + 5N + 9F + 9$	$O(NM)$
SOFM:	$3N'MF + 4N'M + 5N'F - MF + 6N' - M + 3F + 3$	$O(N'M)$

Overall, the results in Table 2.3 indicate that FA's worst-case processing is the least sensitive to the parameters M, N and F. Its processing time is consumed mainly by the winner-take-all competitions between output neurons (choice function computations). Among the other networks, IAFC has the second lowest worst-case running time per input pattern, followed by SOFM and FMMC. Although FA, FMMC, and IAFC have similar ART-type control structures, FMMC incurs an overlap test and contraction procedure to eliminate hyperrectangle overlap, whereas IAFC incurs two membership measures (fuzzy and intra-cluster), and a supplementary category choice procedure based on Euclidean distances. SOFM processing is more straightforward than that of ART-type SONNs, since there is no category expansion test, nor search process. The selection of a winning map neuron has a complexity comparable to FMMC and IAFC choice functions, even though the number of map neurons, N', is usually greater than N (several map neurons may be needed to represent a single category with SOFM). However, the weight update step may involve several map neurons, instead of only one with ART-type SONNs. This m-learn strategy is compounded by the map interpretation post-processing.

The worst-case processing time as a function of the number of neurons N is an important consideration for implementation. As an example, suppose that one wishes to implement a SONN having $M = 16$ dimensions using a factor $F = 10$. Figure 2.2 shows T/o_1 versus N for all four SONNs. The number of map neurons in SOFM is assumed to be three times the number of categories ($N' = 3N$). FA is clearly the

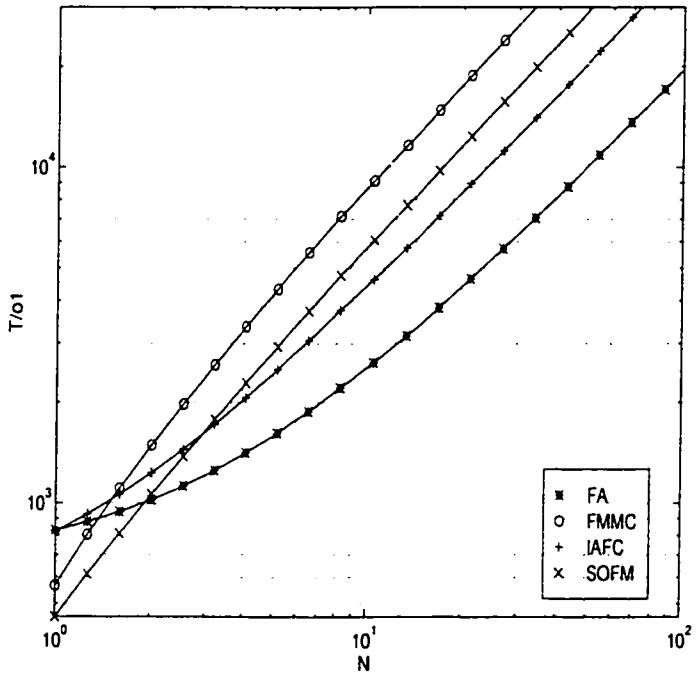


Figure 2.2: Worst-case running time (T/o_1) versus N, assuming that $M = 16$ and $F = 10$. The number of map neurons in SOFM is set to $N' = 3 \cdot N$.

fastest to process an input, followed by IAFC, SOFM and FMMC.

Another consideration is storage of weight values. FA and FMMC require $2MNb$ bits for storage of their weights. IAFC needs only half of that memory when input patterns are normalized. A memory size of $N'Mb$ bits is required with SOFM, but N' is normally greater than N (we have assumed a ratio of 3 to 1 so far).

Although this complexity analysis is based on a generic single processor random access machine, all these networks are suitable for parallel implementation using a multitude of VLSI architectures. The time complexity estimations obtained in our analysis then provide an indication of the semiconductor area required by the parallel

hardware to meet a desired input pattern rate.

2.4.3 Discussion

Simulation results on Radar data show that SOFM can yield excellent clustering scores if it is granted sufficient convergence time. By contrast, FA converges much faster, is computationally inexpensive, but yields lower scores than SOFM. In the case of FMMC and IAFC, the clustering quality is comparable to FA, but their processing requirements are significantly higher. These fundamental differences suggest that SOFM and FA could be attractive alternatives suitable for different radar ESM applications.

Consider the following example. The highest clustering score for the Radar data presented in the interleaved order is obtained using SOFM ($S_{RA}(A, R) = 0.95$). This is a very satisfactory score for radar ESM. However, SOFM converges very slowly and is computationally intensive. Assuming $M = 16$, $F = 10$ and $N' = 3 \cdot N = 30$ map neurons, it requires approximately 18000 elementary operations (T/o_1) for every input pattern. FA, on the other hand, achieves the second highest score for the interleaved radar data ($S_{RA}(A, R) = 0.80$). This is significantly lower than that of SOFM. But, assuming $M = 16$, $F = 10$ and $N = 10$ output neurons, FA requires just 2500 elementary operations per input pattern, and converges with almost 14 times fewer input pattern presentations than SOFM (3.6 epochs instead of 50).

Based on such numbers, SOFM could be well suited for radar ESM systems em-

ployed for long-range surveillance, intelligence, or targeting. In these tasks, accuracy prevails over timing: some delay can be tolerated in exchange for enhanced precision. Also, the high computational cost of SOFM may be less of a problem in shipborne or landbased installations, where equipment of some weight and size can be accommodated.

As for FA, it could better address the requirements of radar ESM systems used for threat alert. In such systems, reaction time must be minimized in order to engage timely protection measures against missiles, anti-aircraft artillery, intercept aircraft and other threats. The fast convergence of FA would thus be an indisputable asset. Also, the low computational complexity of FA would permit its use where space is scarce, and cost must remain low, like in small aircraft radar warning receivers.

2.5 Conclusion

A comparison of four self-organizing neural networks (SONNs) that are potentially suitable for high throughput clustering of radar pulses in ESM systems has been presented. These SONNs are the Fuzzy Adaptive Resonance Theory (FA), Fuzzy Min-Max Clustering (FMMC), Integrated Adaptive Fuzzy Clustering (IAFC), and Self-Organizing Feature Mapping (SOFM).

The SONNs have been compared from three standpoints — clustering quality, convergence time, and computational complexity. Computer simulations have been fed with a Radar data set presented in several statistical orders. Convergence time has

been measured by counting successive presentations of the data set prior to weight stabilization. Clustering quality has been assessed using the Rand Adjusted and Jaccard similarity measures.

Simulation results have shown that: (1) SOFM usually yields the best scores followed by FA, FMMC, and IAFC, (2) FMMC and FA converge the fastest, followed by IAFC and SOFM, (3) the data presentation order has a significant impact on the scores of ART-type networks, and their variability, and (4) SOFM, when granted a long convergence time, is insensitive to the data presentation order. The SONNs have then been studied from an algorithmic perspective to estimate worst-case running times, and consequently, computational complexity. Of the four SONNs, FA has been shown to have the lowest complexity, followed by IAFC, SOFM, and FMMC.

SOFM and FA emerge as suitable candidates for sorting pulses in radar ESM, albeit for different reasons. On the one hand, SOFM can achieve excellent clustering scores at the expense of a high complexity and potentially long convergence time. It would therefore be well suited for long range surveillance, intelligence and targeting. On the other hand, FA has the potential be to fast enough for use in threat alert systems. The selection of either SONN would ultimately depend on the specifics of the ESM application, and effect a tradeoff between clustering accuracy and computational efficiency.

Acknowledgments

This work was supported in part by the Defence Research Establishment Ottawa, the Canadian Microelectronics Corporation, le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (province of Quebec), and the Natural Sciences and Engineering Research Council of Canada.

2.A Simulation results for the Wine Recognition and Iris data sets

The *Wine Recognition* data set[4] is the result of a chemical analysis of wines from 3 different cultivars. The problem consists in determining the origins of 178 wine samples, where 13 attributes characterize the constituents found in each sample. The well-known *Iris* data set [46] contains 150 flowers belonging to 3 species of iris flowers: satosa, versicolor and virginica. Each species contains 50 flowers that are characterized by 4 features: petal and sepal length and width.

Simulation results for the Wine Recognition and Iris data sets are given in Tables 2.A and 2.A, respectively. As with the Radar data, SOFM most often achieves the highest SONN scores, usually followed by FA. The only exceptions are found when the Wine and Iris data are presented in the interleaved order, in which case FA scores the best. IAFC performs almost as well as FMMC for the Wine data, and

[4]This set was obtained from the UCI Machine Learning Repository (www.uci.edu/~mlearn).

surpasses FMMC for the Iris data. In this last case, IAFC almost ties FA for second best. Overall, IAFC appears to be more sensitive than the others to the distribution of the patterns in input space.

It is worth noting that the gap between SOFM scores and those of the other three SONNs is not as wide as with the Radar data. Besides, the three ART-type networks usually perform better when data are presented in more structured, radar-like orders. This holds true to the extent that FA scores slightly higher than ISODATA for Iris data presented in the interleaved order.

The SONN convergence times appear to vary in a proportional manner from one data set to the next: FMMC always converges with the least amount of epochs, followed closely by FA, then by IAFC and SOFM. On average, FA and FMMC converge slightly faster for the Iris data than with the Wine data, and faster with the Wine data than the Radar data. IAFC and SOFM converge in much fewer epochs with the Wine data than either the Iris or Radar data. This discrepancy may be due to a sensitivity to data structures with overlapping clusters. This is especially true for IAFC, whose learning rate, λ, scales according to fuzzy and intra-cluster membership functions [84]. Notice that its convergence for the Iris data requires more epochs than for the Radar data set. In any case, there seems to be little correlation between convergence time and presentation order.

Tableau 2.4: Summary of best simulation results for the Wine Recognition data set.

ALGORITHM: (parameter values) · *presentation order*	**PERFORMANCE MEASURES**				
	Clustering quality				**Convergence time**
	$S_{RA}(A,R)$		$S_J(A,R)$		Number of epochs
	Mean	Std. dev.	Mean	Std. dev.	
ISODATA: ($L = 1; N_c \geq 5; .3 \leq \theta_c \leq .6; .1 \leq \theta_s \leq .4; \theta_N \geq 8$)					
· *random*	0.70	0.060	0.84	0.021	$I \geq 4$
· *burst*	0.76	0.095	0.87	0.032	$I \geq 4$
· *interleaved*	0.76	0.099	0.87	0.031	$I \geq 4$
Fuzzy ART (FA): ($\beta = 1; \alpha = 1; .35 \leq \rho \leq .5$)					
· *random*	0.26	0.056	0.68	0.013	2.9
· *burst*	0.29	0.077	0.70	0.020	3.0
· *interleaved*	0.49	0.110	0.77	0.031	2.8
Fuzzy Min-Max Clustering (FMMC): ($4 \leq \gamma \leq 5; .6 \leq \theta \leq .75$)					
· *random*	0.07	0.012	0.67	0.001	2.1
· *burst*	0.18	0.013	0.69	0.004	2.0
· *interleaved*	0.32	0.112	0.71	0.031	2.0
Integrated Adaptive Fuzzy Clustering (IAFC): ($\gamma = 1; m = 2; .6 \leq \sigma \leq .9; .1 \leq k \leq .8; .4 \leq \tau \leq .6$)					
· *random*	0.13	0.013	0.68	0.003	6.5
· *burst*	0.13	0.009	0.68	0.002	7.6
· *interleaved*	0.18	0.009	0.70	0.002	6.8
Self-Organizing Feature Mapping (SOFM): (3×3 map; $2 \leq \zeta \leq 11; .01 \leq \alpha(0) \leq 1; 1 \leq r(0) \leq 2$)					
· *random*	0.48	0	0.73	0	14
· *burst*	0.50	0	0.73	0	14
· *interleaved*	0.50	0	0.73	0	14

Tableau 2.5: Summary of best simulation results for the Iris data set.

ALGORITHM: (parameter values) · *presentation order*	**PERFORMANCE MEASURES**				
	Clustering quality				**Convergence time**
	$S_{RA}(A,R)$		$S_J(A,R)$		Number of epochs
	Mean	Std. dev.	Mean	Std. dev.	
ISODATA: ($1 \leq L \leq 3; \forall N_c; 1 \leq \theta_c \leq 1.5; \theta_s < 1.2; 15 \leq \theta_N \leq 22$)					
· *random*	0.72	0.054	0.83	0.030	$I \geq 6$
· *burst*	0.72	0.063	0.84	0.035	$I \geq 6$
· *interleaved*	0.73	0.068	0.89	0.036	$I \geq 6$
Fuzzy ART (FA): ($.7 \leq \rho \leq .75$; $\alpha = 1$; $\beta = 1$)					
· *random*	0.68	0.077	0.82	0.042	2.3
· *burst*	0.68	0.085	0.83	0.047	2.6
· *interleaved*	0.75	0.109	0.86	0.042	1.8
Fuzzy Min-Max Clustering (FMMC): ($4 \leq \gamma \leq 5$; $.4 \leq \gamma \leq .55$)					
· *random*	0.53	0.078	0.78	0.034	1.8
· *burst*	0.60	0.103	0.79	0.036	2.1
· *interleaved*	0.68	0.218	0.82	0.045	1.6
Integrated Adaptive Fuzzy Clustering (IAFC): ($\gamma = 1$; $m = 2$; $.2 \leq \sigma \leq .5$; $.3 \leq k \leq .7$; $.8 \leq \tau \leq 1$)					
· *random*	0.68	0.046	0.81	0.019	26.3
· *burst*	0.68	0.016	0.83	0.009	24.7
· *interleaved*	0.72	0.033	0.84	0.012	23.0
Self-Organizing Feature Mapping (SOFM): (3×3 map; $7 \leq \zeta \leq 12$; $.07 \leq \alpha(0) \leq .3$; $1 \leq r(0) \leq 3$)					
· *random*	0.69	0	0.84	0	45
· *burst*	0.69	0	0.84	0	45
· *interleaved*	0.69	0	0.84	0	45

2.B Analysis of computational complexity

In this appendix, details of a computational complexity analysis are presented[5] for each SONN algorithm. First, the maximum time required to execute every operation of type p is derived ($T_p = o_p \cdot n_p$). Table 2.6 gives a breakdown of time complexities, organized according to algorithm steps. The time complexity of a step is equal to the product of its cost per iteration and the number of iterations it needs. Notice how the values for some of these steps are obtained by adding the contribution of several sub-steps (shown directly below). Then, the sum of these time complexities yields a total worst-case running time per input pattern, $T = \sum_p T_p$, and a corresponding growth rate. The resulting complexity polynomial is expressed in a normalized format, T/o_1, in terms of network parameters M, N and F, and provides an estimate of computational complexity. The following cost values o_p are used throughout the analysis:

- o_1: the time required to execute elementary, low-level operations (*i.e.*, $x + y$, $x - y$, $\max(x, y)$, $1 - x$, $x < y$, etc.). The values x and y are real numbers represented with a b bit resolution.

- $o_{2,q}$: the time needed to compute a division x/y ($o_{2,1}$), a multiplication $x \cdot y$ ($o_{2,2}$), a square root $\sqrt{x}$ ($o_{2,3}$), or an exponent e^x ($o_{2,4}$). Notice that operations of type

[5] A similar type of analysis has been conducted by Lawrence et al. [89] to approximate the computational complexity of various tasks done by a face recognition system. This type of analysis is important in the context of implementing real-time systems, where complexity scales according to system parameters.

$p = 2$ are subdivided into operations of type $q = \{1, 2, 3, 4\}$ on account for their dissimilar complexities. The complexity of these operations is a function of b ($o_{2,q} = o_{2,q}(b)$), and they differ according to specific implementations. However, in order to ease the analysis and the comparison of results, we assume that $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq o_{2,4} \simeq F(b) \cdot o_1 = F \cdot o_1$, where parameter F reflects the use of higher complexity operations in the analysis. This is a reasonable assumption if dedicated hardware is available to support the requested operations.

As explained previously, this complexity analysis assumes that the SONN algorithms are implemented using a random access machine (RAM) model of computation [37]. They are also supposed to be executed in an order permitting efficient reuse of data (*e.g.*, no values are computed twice). Input patterns are considered to be normalized, and all weight and parameter values are presumed to be initialized prior to the presentation of any data. The time associated with these last two operations is ignored since it is comparable for all the SONNs. The SONNs form clusters progressively by sequential processing of input data patterns. ISODATA is an iterative procedure that requires all the input data at once to form a predefined number of clusters. Since ISODATA cannot be analyzed in the same way as the SONNs, it is omitted from the complexity analysis.

Tableau 2.6: Breakdown of time complexities.

SONN	TIME COMPLEXITY	
Steps of the algorithm:	**Cost per iteration**	**Iteration count**
FA		
A. Complement coding:	$o_1 M$	1
B. Choice functions:	$o_1(6M+1)+o_{2,1}$	N
C. Search process:		
C.1. vigilance test	o_1	N
C.2. category choice	o_1	N
D. Prototype vector update:	$4M(o_1+o_{2,2})$	1
FMMC		
A. Membership functions:	$M(9o_1+2o_{2,2})+o_{2,1}$	N
B. Search process:		
B.1. expansion test	$3o_1M+o_1+o_{2,1}$	N
B.2. hyperrectangle choice	o_1	N
C. Hyperrectangle expansion:	$2o_1M$	1
D. Overlap test:	$12o_1M$	$N-1$
E. Hyperrectangle contraction:	$M(o_1+o_{2,1})$	$N-1$
IAFC		
A. Pre-computations:		
A.1. $\|\|\mathbf{a}-\mathbf{v}_j\|\|$, $\forall j$	$M(o_1+o_{2,1})+o_{2,3}$	N
A.2. $1/(\|\|\mathbf{a}-\mathbf{v}_j\|\|^2)$, $\forall j$	$o_{2,1}+o_{2,2}$	N
A.3. $\sum_{j=1}^{N} 1/(\|\|\mathbf{a}-\mathbf{v}_j\|\|^2)$	o_1	N
B. Choice functions	$o_{2,2}M$	N
C. Search process:		
C.1. vigilance test	$2o_1+o_{2,1}+o_{2,2}+o_{2,4}$	N
C.2. membership test	$o_1+o_{2,1}$	N
C.3. category choice	o_1	N
D. Centroid vector update:	$2o_{2,2}M+9o_1+3o_{2,1}+6o_{2,2}$	1
SOFM		
A. Map neuron selection:		
A.1. Euclidean distances	$M(o_1+o_{2,1})+o_{2,3}$	N'
A.2. best-match choice	o_1	N'
A.3. increment its input count	o_1	N'
B. Update prototype(s):		
B.1. define neighborhood	$3o_1+2o_{2,1}+o_{2,3}$	N'
B.2. adjust weights	$M(2o_1+o_{2,2})$	N'
C. Modify $\alpha(t)$ and $r(t)$:	$2(o_1+o_{2,1}+o_{2,2})$	1
D. Map interpretation:		
D.1. find category centers	o_1	N'
D.2. relate neurons	$M(o_1+o_{2,1})+o_{2,3}$	$N'-1$
D.3. category choice	o_1	1

2.B.1 Fuzzy ART (FA)

In Table 2.6, the breakdown of time complexities shows that operation B, the computation of the output neuron choice functions, is potentially the most costly with FA. The sequence of the FA search process (operation C) has been modified according to Subsection 2.2. The vigilance test is computed only once for all N output neurons prior to category choice. Similar modifications to FMMC and IAFC were also assumed. The total worst-case running time T_{FA} corresponds to the sum of the elements multiplied in the "Time complexity" column:

$$\begin{aligned} T_{FA} &= o_1 M + N[o_1(6M+1) + o_{2,1}] + 2o_1 N + 4M(o_1 + o_{2,2}) \\ &= NM(6o_1) + N(3o_1 + o_{2,1}) + M(5o_1 + 4o_{2,2}) \\ &= o_1(6NM + 3N + 5M) + o_{2,1}(N) + o_{2,2}(4M) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq F \cdot o_1$, then:

$$T_{FA} = o_1 \cdot [6NM + NF + 4MF + 3N + 5M] \tag{2.4}$$

The rate of growth of T_{FA} when F is constant and $M, N \gg 1$ is $O(NM)$.

2.B.2 Fuzzy Min-Max Clustering (FMMC)

As with FA, the computation of output neuron membership functions (operation A) is potentially very demanding. FMMC's simple hyperrectangle expansion

procedure (operation C) is equivalent to FA's fast learning rule (with $\beta = 1$). However, its combination of overlap test and contraction procedure (operations D and E), are also notably time consuming. The total worst-case running time T_{FMMC} is:

$$\begin{aligned} T_{FMMC} &= N[M(9o_1 + 2o_{2,2}) + o_{2,1}] + N[3o_1M + 2o_1 + o_{2,1}] + 2o_1M + \\ &\quad (N-1)12o_1M + (N-1)[M(o_1 + o_{2,1})] \\ &= NM(25o_1 + o_{2,1} + 2o_{2,2}) + 2N(o_1 + o_{2,1}) - M(11o_1 + o_{2,1}) \\ &= o_1(25NM + 2N - 11M) + o_{2,1}(NM + 2N - M) + o_{2,2}(2NM) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq F \cdot o_1$, then:

$$T_{FMMC} = o_1 \cdot [3NMF + 25NM + 2NF - MF + 2N - 11M] \qquad (2.5)$$

The rate of growth of T_{FMMC} when F is constant and $M, N \gg 1$ is $O(NM)$.

2.B.3 Integrated Adaptive Fuzzy Clustering (IAFC)

The computation of the Euclidean distances and related values in operation A (between input **a** and the prototype vectors $\mathbf{v}_j$) is potentially very time consuming. These values are needed at several instances in IAFC's processing. For example, fuzzy membership values are needed for an alternate choice procedure (see operation C.2). The complexity required for the choice functions and the search process is comparable to that of FMMC. Its centroid update (operation D) is as costly as the

prototype vector update of FA. The total worst-case running time T_{IAFC} is:

$$\begin{aligned} T_{IAFC} &= N[M(o_1 + o_{2,1}) + o_1 + o_{2,1} + o_{2,2} + o_{2,3}] + N[o_{2,2}M] + \\ &\quad N[4o_1 + 2o_{2,1} + o_{2,2} + o_{2,4}] + 2o_{2,2}M + 9o_1 + 3o_{2,1} + 6o_{2,2} \\ &= NM(o_1 + o_{2,1} + o_{2,2}) + N(5o_1 + 3o_{2,1} + 2o_{2,2} + o_{2,3} + o_{2,4}) + M(2o_{2,2}) + \\ &\quad 9o_1 + 3o_{2,1} + 6o_{2,2} \\ &= o_1(NM + 5N + 9) + o_{2,1}(NM + 3N + 3) + o_{2,2}(NM + 2N + 2M + 6) + \\ &\quad o_{2,3}(N) + o_{2,4}(N) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq o_{2,4} \simeq F \cdot o_1$, then:

$$T_{IAFC} = o_1 \cdot [2NMF + NM + 7NF + 2MF + 5N + 9F + 9] \qquad (2.6)$$

The rate of growth of T_{IAFC} when F is constant and $M, N \gg 1$ is $O(NM)$.

2.B.4 Self-Organizing Feature Mapping (SOFM)

The SOFM processing is more straightforward than that of the ART-type SONNs, since there is no search process. Most of the workload is concentrated in 3 operations: A, B and D. The selection of a winning map neuron has a complexity comparable to the FMMC and IAFC choice functions, even though N' is usually greater than N (several map neurons may be needed to represent a single category with SOFM).

Notice also the weight update required for several map neurons, instead of just one with ART-type SONNs. The total worst-case running time T_{SOFM} is:

$$\begin{aligned} T_{SOFM} &= N'[M(o_1 + o_{2,1}) + 2o_1 + o_{2,3}] + N'[M(2o_1 + o_{2,2}) + 3o_1 + 2o_{2,1} + o_{2,3}] + \\ &\quad 2(o_1 + o_{2,1} + o_{2,2}) + N'[M(o_1 + o_{2,1}) + o_1 + o_{2,3}] - M(o_1 + o_{2,1}) + o_1 - o_{2,3} \\ &= N'M(4o_1 + 2o_{2,1} + o_{2,2}) + N'(6o_1 + 2o_{2,1} + 3o_{2,3}) - M(o_1 + o_{2,1}) + 3o_1 + \\ &\quad 2o_{2,1} + 2o_{2,2} - o_{2,3} \\ &= o_1(4N'M + 6N' - M + 3) + o_{2,1}(2N'M + 2N' - M + 2) + o_{2,2}(N'M + 2) + \\ &\quad o_{2,3}(3N' - 1) \end{aligned}$$

If parameter F is comparable for all type $p = 2$ operations, $o_{2,1} \simeq o_{2,2} \simeq o_{2,3} \simeq F \cdot o_1$, then:

$$T_{SOFM} = o_1 \cdot [3N'MF + 4N'M + 5N'F - MF + 6N' - M + 3F + 3] \qquad (2.7)$$

The rate of growth of T_{SOFM} when F is constant and $M, N' \gg 1$ is $O(N'M)$.

2.C Summary of SONN algorithms

2.C.1 Fuzzy ART (FA)

The FA's functionality can be described as a five steps algorithm:

1. Weights and parameters initialization: Initially, all the neurons of F2 are uncommitted, and all weight values w_{ji} are initialized to 1. An F2 neuron becomes committed when it is selected for an input **a**, then the corresponding weights w_{ji} can take real values in the interval [0,1].

2. Input vector complement coding: When a new M-element input vector $\mathbf{a} = (a_1, a_2, ..., a_M)$ (where each element a_i is a real number in the interval [0,1]) is presented to the network, it undergoes a preliminary *complement coding* [21]. This gives a network input vector $\mathbf{I}$ of $2M$ elements such that: $\mathbf{I} = (\mathbf{a}; \mathbf{a}^c) = (a_1, a_2, ..., a_M; a_1^c, a_2^c, ..., a_M^c)$, with $a_i^c = 1 - a_i$. This coding is recommended to prevent a category proliferation problem that may occur in analog ART networks [103].

3. Category choice: With the presentation of an input $\mathbf{I}$ to F1, the *choice function* $T_j(\mathbf{I})$ is calculated for each neuron j in F2:

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{2.8}$$

where $|\cdot|$ is the norm operator ($|\mathbf{w}_j| = \sum_{i=1}^{2M} |w_{ji}|$), $\wedge$ is the fuzzy logic AND operator ($\mathbf{I} \wedge \mathbf{w}_j = (\min(I_1, w_{j,1}), \min(I_2, w_{j,2}), ..., \min(I_{2M}, w_{j,2M}))$), and α is a user-defined *choice parameter* such that $\alpha > 0$. F2 is a winner-take-all competitive layer, where

the winner is the neuron $j = J$ with the greatest value of activation T_j for the input $\mathbf{I}$ ($T_J = \max\{T_j : j = 1...N\}$). If the same T_J value is obtained by two or more neurons, the one with the smallest index j wins. The winning neuron J is retained for Step 4.

4. Vigilance test: This step serves to compare the similarity between the prototype vector of the winning neuron $\mathbf{w}_J$ and input $\mathbf{I}$, against a user-defined *vigilance parameter* ρ, through the following test:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{2.9}$$

where $\rho = [0, 1]$. This comparison is carried out on layer F1: the winning neuron J transmits its learned expectation, $\mathbf{w}_J$, to F1 for comparison with $\mathbf{I}$. If the vigilance test (Eq. 2.9) is passed, then J becomes *selected*, and it is allowed to adapt its prototype vector (Step 5). Otherwise, J is deactivated for $\mathbf{I}$: T_J is set equal to -1 for the duration of the current input presentation. The algorithm searches through the remaining F2 layer neurons (Steps 3 and 4), until some other neuron J passes the vigilance test. If no committed neuron from the F2 layer can pass this test, an uncommitted F2 neuron is selected to undergo prototype vector update, and becomes committed.

5. Prototype vector update: The prototype vector of the winning neuron J is updated according to:

$$\mathbf{w}'_J = \beta(\mathbf{I} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J \tag{2.10}$$

where β is a user-defined *learning rate parameter* such that $\beta = [0, 1]$. The algorithm can be set to slow learning, with $0 < \beta < 1$, or to fast learning, with $\beta = 1$. Once this update step is accomplished, the network reactivates all F2 neurons, and can process a new input vector.

2.C.2 Fuzzy Min-Max Clustering (FMMC)

The FMMC network's algorithm may be described with four steps:

1. Category hyperrectangle and parameter initialization: All the category neurons of F2 are considered to be uncommitted, and all hyperrectangles B_j are initialized such that $\mathbf{u}_j = \mathbf{1}$ and $\mathbf{v}_j = \mathbf{0}$.

2. Hyperrectangle expansion: Let $B_j = \{\mathbf{a}, \mathbf{u}_j, \mathbf{v}_j, b_j\}$ define the category hyperrectangle fuzzy set for F2 neuron j, where $\mathbf{a} = (a_1, a_2, ..., a_M)$ is an input pattern, and $b_j = b_j(\mathbf{a}, \mathbf{u}_j, \mathbf{v}_j) \in [0, 1]$ is a membership function defined by:

$$b_j(\mathbf{a}, \mathbf{u}_j, \mathbf{v}_j) = \frac{1}{M}\sum_{k=1}^{M}[1 - f(a_k - v_{jk}, \gamma) - f(u_{jk} - a_k, \gamma)] \tag{2.11}$$

The ramp threshold function $f(\cdot)$ is defined such that $f(x,\gamma) = 1$ if $x \cdot \gamma > 1$, $f(x,\gamma) = x \cdot \gamma$ if $0 \leq x \cdot \gamma \leq 1$, and $f(x,\gamma) = 0$ if $x \cdot \gamma < 0$. When a new input vector $\mathbf{a} = (a_1, a_2, ... a_M)$ is presented to the network, function b_j measures the degree to which it belongs to hyperrectangle j. The closer $\mathbf{a}$ is to the hyperrectangle points, the closer b_j is to 1, subject to a *sensitivity parameter* γ. For a set of committed category neurons in F2, the closest hyperrectangle B_J that can expand (if needed) to include $\mathbf{a}$ is identified. The winner $j = J$ is the category in which $\mathbf{a}$ yields the greatest degree of membership ($b_J = \max\{b_j : j = 1....N\}$). This selected category hyperrectangle is then submitted to an expansion test:

$$\frac{1}{M}\sum_{k=1}^{M}[\max(v_{Jk}, a_k) - \min(u_{Jk}, a_k)] \leq \theta \tag{2.12}$$

where $\theta \in [0,1]$ is a user-defined *maximum hyperrectangle size* parameter. If this last constraint is met, then expansion learning takes place: the min and max points of the selected category hyperrectangle are adjusted so that $u'_{Jk} = \min(u_{Jk}, a_k)$ and $v'_{Jk} = \max(v_{Jk}, a_k)$ for $k = 1, 2, ..., M$. Otherwise, alternate committed category hyperrectangles may be selected. If all else fails, a new category hyperrectangle is committed, and is assigned $\mathbf{a}$.

3. Overlap test: A hyperrectangle represents the portion of a category fuzzy set to which the corresponding patterns have full membership. Hyperrectangles are not allowed to overlap, although the portions of the fuzzy sets with partial membership do overlap. After each hyperrectangle expansion process, the overlap of B_J is quantified

with respect to the other existing hyperrectangles B_h $(h = 1, 2, ..., N, h \neq J)$ for the following 4 cases, along each of the M dimensions with (a): $v_{hk} > v_{Jk} > u_{hk} > u_{Jk}$, (b): $v_{Jk} > v_{hk} > u_{Jk} > u_{hk}$ (for partial hyperrectangle overlap), (c): $v_{Jk} > v_{hk} \geq u_{hk} > u_{Jk}$, and (d): $v_{hk} > v_{Jk} \geq u_{Jk} > u_{hk}$ (for full hyperrectangle overlap), where $k = 1, 2, ..., M$.

4. Hyperrectangle contraction: Overlap between the selected hyperrectangle B_J and another one, B_h, is eliminated using a contraction process on a dimension-by-dimension basis. For the 4 cases described in Step 3, the overlap is eliminated with (a): $u'_{hk} = v'_{Jk} = (u_{hk} + v_{Jk})/2$, (b): $u'_{Jk} = v'_{hk} = (u_{Jk} + v_{hk})/2$, (c): $u'_{Jk} = v_{hk}$, (d): $v'_{Jk} = u_{hk}$, for $k = 1, 2, ..., M$. Then, the network reactivates all F2 neurons, and is ready to repeat steps 2 to 4 with a new input pattern.

2.C.3 Integrated Adaptive Fuzzy Clustering (IAFC)

The IAFC network's algorithm can be summarized with the following four step procedure:

1. Weights and parameters initialization: Initially, all the neurons of F2 are uncommitted, and all weight values v_{ji} and b_{ji} are initialized to 0. An F2 neuron becomes committed when it is selected for an input **a**, then the corresponding weights v_{ji} can take real values greater than 0. The weights b_{ji} are always normalized versions of the v_{ji}.

2. Category choice: When a new input pattern $\mathbf{a} = (a_1, a_2, ..., a_M)$ is presented, it is normalized. The resulting input pattern $\mathbf{I} = (I_1, I_2, ... I_M)$ is applied to the network's F1 neurons, through bottom up weights b_j, and a winner-take-all competition occurs among F2 neurons. The dot product Y_j is found for each F2 neuron j:

$$Y_j = \sum_{i=1}^{M} I_i \cdot b_{ji} = \mathbf{I} \cdot \mathbf{b}_j^T \tag{2.13}$$

The output neuron $j = J$ that receives the largest activation is selected ($Y_J = \max\{Y_j : j = 1, 2, ..., N\}$). By using this correlation-type choice function, the winner is decided by the angle between $\mathbf{I}$ the $\mathbf{b}_j$s. This may cause misclassification since the input-centroid distance is not considered. In such cases, the Euclidean distance may be a better alternative for category choice. For a category choice, IAFC uses the following combined similarity measure. After using Eq. 2.13, and selecting a winner J, the fuzzy membership value μ_J of the input pattern in cluster J is computed:

$$\mu_J = \frac{\left(\frac{1}{||\mathbf{a}-\mathbf{v}_J||^2}\right)^{\frac{1}{m-1}}}{\sum_{j=1}^{N}\left(\frac{1}{||\mathbf{a}-\mathbf{v}_j||^2}\right)^{\frac{1}{m-1}}} \tag{2.14}$$

where $m \in [1, \infty)$ is the weight exponent (experimentally set to 2 [84]), and $||\cdot||$ is the Euclidean distance. If μ_J is less than σ, the algorithm activates an alternate procedure to find a winner: a cluster whose centroid $\mathbf{v}_j$ is closest to $\mathbf{a}$ according to the Euclidean distance ($Y_J' = \min\{||\mathbf{a} - \mathbf{v}_j|| : j = 1, 2, ..., N\}$). Therefore, if σ is set low, the angle between $\mathbf{I}$ and the $\mathbf{b}_j$s becomes dominant in the choice of a winner,

otherwise the Euclidean distance is dominant in this choice.

3. Vigilance test: The choice of winning cluster J is validated according to the following test:

$$e^{-\gamma \cdot \mu_J} \cdot ||\mathbf{a} - \mathbf{v}_J|| \leq \tau \tag{2.15}$$

where γ is a factor that controls the shape of clusters formed (normally set to 1 [84]), and τ is the *vigilance parameter*. This similarity measure considers the Euclidean distance between $\mathbf{a}$ and $\mathbf{v}_J$, and the relative degree of belonging of $\mathbf{a}$ to every committed category. If the winning J satisfies the test, this cluster's centroid is updated (Step 4). Otherwise, neuron J is deactivated, and another cluster is selected as winner (Steps 2 and 3). If no cluster can pass the vigilance test, an uncommitted F2 neuron is selected for centroid update.

4. Update the winning centroids: The centroid of the selected category is updated according to:

$$\mathbf{v}'_J = (1 - \lambda) \cdot \mathbf{v}_J + \lambda \cdot \mathbf{a} \tag{2.16}$$

where $\lambda = f(l) \cdot \pi(\mathbf{a}, \mathbf{v}_J, \tau) \cdot \mu_J^2$. The value l is the number of iterations of the data set, $f(l) = 1/\{k(l-1)+1\}$ is a decreasing function of l, and $\pi(\mathbf{a}, \mathbf{v}_J, \tau)$ is the intra-cluster membership value of $\mathbf{a}$ to the winner J, defined by:

$$\pi(\mathbf{a}, \mathbf{v}_J, \tau) = \begin{cases} 1 - 2(||\mathbf{a} - \mathbf{v}_J||/\tau)^2 & \text{if } 0 \leq ||\mathbf{a} - \mathbf{v}_J|| < \tau/2 \ ; \\ 2(1 - ||\mathbf{a} - \mathbf{v}_J||/\tau)^2 & \text{if } \tau/2 \leq ||\mathbf{a} - \mathbf{v}_J|| < \tau \ ; \\ 0 & \text{if } ||\mathbf{a} - \mathbf{v}_J|| \geq \tau \ . \end{cases} \tag{2.17}$$

A normalized version of $\mathbf{v}'_J$ ($\mathbf{b}'_J$) is also calculated for subsequent processing. The previously deactivated output neurons are enabled, and a new input pattern is accepted at Step 2.

2.C.4 Self-Organizing Feature Mapping (SOFM)

The SOFM algorithm used in this paper is summarized in terms of the following 4 steps:

1. Weights and parameter initialization: Weights $\mathbf{w}_j(0) = (w_{j1}(0), w_{j2}(0), ..., w_{jM}(0))$ from each map neuron to all the input neurons are initialized to small random numbers. The initial learning rate $\alpha(0)$ and neighborhood radius $r(0)$ are also set prior to training. For every neuron j of the map, the *assigned input count* $c(j)$ is reset to 0.

2. Map neuron selection: At time t, the input pattern $\mathbf{a}(t) = (a_1(t), a_2(t), ..., a_M(t))$ is compared with each map neuron prototype vector $\mathbf{w}_j(t)$ using the Euclidean distance $||\mathbf{a}(t) - \mathbf{w}_j(t)||$. The best-matching map neuron J, the one that minimizes the Euclidean distance, is selected according to:

$$J = \arg\min\{||\mathbf{a}(t) - \mathbf{w}_j(t)|| : j = 1, 2, ..., N\} \tag{2.18}$$

The assigned input count of the best-match map neuron is incremented ($c(J) = c(J) + 1$).

3. Weights updating: The input $\mathbf{a}(t)$ is mapped onto the neuron J relative to all other $\mathbf{w}_j(t)$ values. During learning, the neurons that are topographically close to neuron J's location in the map (within a neighborhood) activate each other to learn from the input. Weights connected to J and to other neurons in its neighborhood become more alike through learning [104], thus more responsive to $\mathbf{a}(t)$. They are adjusted according to the learning rule:

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + n_{Jj}(t)[\mathbf{a}(t) - \mathbf{w}_j(t)] \tag{2.19}$$

where $n_{Jj}(t)$ is the neighborhood *kernel* that is a function defined over time, and the map neurons. The kind of kernel used in this paper is named *bubble*, whose definition considers a set of map neurons neighboring the best-match neuron J. This set is denoted $N_J(t)$, and it contains the set of map neurons considered to be in the neighborhood of map neuron J at time t. Then, $n_{Jj}(t) = \alpha(t)$ if neuron j belongs to $N_J(t)$, and $n_{Jj}(t) = 0$ otherwise, where $\alpha(t)$ is the *learning rate* parameter $(0 < \alpha(t) < 1)$. Specifically, the set $N_J(t)$ is a function of $d_{Jj} = |r_J - r_j|$, the lateral map distance of neuron J to neuron j, which is a Euclidean measure in the R^2 output space. The size of the set $N_J(t)$ is regulated by the threshold parameter $r(t)$. If $d_{Jj} \leq r(t)$, the map neuron j belongs to J's neighborhood, and it is allowed to learn from $a(t)$. Then, parameters $\alpha(t)$ and $r(t)$ are adjusted: they are progressively decreased in a slow and linear way from their initial values, $r(0)$ and $\alpha(0)$, to 1 and 0, respectively. If $\alpha(t) \geq 0.5 \cdot \alpha(0)$, then the feature map may be interpreted (Step 4).

4. Map interpretation: Every map neuron j whose assigned input count $c(j)$ is greater than ζ, a user-defined *category assignment* parameter, is interpreted as a category center. If $c(J)$ is smaller, or equal to ζ, find the closest category center K according to:

$$K = \arg\min_{k}\{||\mathbf{w}_k(t+1) - \mathbf{w}_J(t+1)||\} \tag{2.20}$$

At this stage in training, this search may be bound to J's adjacent map neurons, given the topographical relationships. Then, input $\mathbf{a}(t)$ is implicitly assigned to K without additional weight updating. The integer discrete-time coordinate t is incremented, and Steps 2 to 4 are repeated with data set patterns until $\alpha(t)$ is reduced to 0, and the weight values are fixed, at which time a feature map has been formed.

2.6 Synthèse et impact des résultats

Dans l'article précédent, quatre RNA auto-organisateurs de type apprentissage compétitif ont été comparés en termes de la qualité de catégorisation et de l'effort de calcul requis. Les résultats des simulations et des estimations de complexité ont permi la mise en évidence des conclusions suivantes. Premièrement, les résultats de simulation (avec un ensemble de données représentatif en MSE radar) ont révélé que le réseau SOFM procure habituellement la meilleure qualité de catégorisation, suivi le plus souvent par FA, FMMC et IAFC. Les réseaux FMMC et FA convergent les plus rapidement, suivis par IAFC et SOFM. Lorsqu'on lui accorde une période suffisante pour converger, SOFM devient insensible à l'ordre de présentation des données. Cependant, l'ordre de présentation a un impact considérable sur la qualité et la variabilité des catégorisations générées par FA, FMMC et IAFC. Deuxièmement, les résultats d'estimation de complexité ont révélé que FA est défini par une procédure moins complexe que les trois autres réseaux. C'est-à-dire qu'on estime (dans le pire cas) que l'algorithme du RNA FA demande le plus petit temps de traitement pour un patron, suivi par IAFC, SOFM et FMMC.

Selon ces résultats, on peut conclure que les réseaux SOFM et FA sont les plus convenables pour le triage métrique rapide de séquences d'impulsions, sans connaissance a priori. En fin de compte, le choix d'un ou de l'autre dépend des besoins spécifiques de l'application en MSE radar, et fait apparaître un compromis entre la qualité des catégorisations et l'efficacité des calculs. SOFM produit des catégorisations très

précises, mais peut exiger un long délai pour converger et est caractérisé par une complexité de calcul élevée. Alors, celui-ci serait plus approprié pour des systèmes de surveillance, d'intelligence et de ciblage à longue portée, où la précision est plus critique. En revanche, FA produit des catégorisations un peu moins précises que SOFM, mais il possède un grand potentiel pour le traitement à haute vitesse. Celui-ci serait plutôt désirable pour des systèmes d'alerte contre les menaces, où le temps de réaction et/ou la compacité du système est plus critique.

Il est à noter que ni SOFM ni FA ne sont pratiques, comme tels, pour le triage métrique en MSE radar. En effet, SOFM perd progressivement sa capacité d'apprendre de nouvelles informations, ce qui est acceptable dans le contexte d'apprentissage d'un lot d'impulsions radars. Une version modifiée de SOFM serait souhaitable pour retenir un certaine pasticité face à l'environment, ou pour fusionner des résultats obtenus avec différents lots de données. D'autre part, un mécanisme supplémentaire (pour localiser et compter le nombre de catégories formées) a du être conçu afin de faciliter l'interprétation des résultats avec SOFM.

FA n'est pas limité par ces derniers facteurs, mais plutôt par son manque de précision, ainsi que par la dépendance de ses résultats sur l'ordre de présentation des données. Il serait intéressant de mener une étude plus approfondie concernant l'effet de l'ordre de présentation des données sur les résultats d'un RNA de type ART. Finalement, il serait utile de développer des critères pour pouvoir re-initialiser les neurones correspondant aux émetteurs qui ne sont plus actifs. Une telle modifica-

tion peut réduire le montant de resources requises, et offrir un vue plus précise des émetteurs dans l'environnement.

Les deux prochains chapitres présentent des travaux qui font suite aux résultats de l'étude comparative précédente. Ces contributions élaborent des concepts qui sont liés au triage métrique rapide basé sur le RNA FA.

Chapitre 3

Une architecture VLSI pour le réseau de neurones fuzzy ART

Selon la comparaison au chapitre 2, le RNA fuzzy ART a un grand potentiel pour supporter le triage rapide d'impulsions radars. Une réalisation du RNA fuzzy ART avec un circuit dédié VLSI peut permettre d'atteindre une cadence de traitement élevée. Le reste de ce chapitre contient une description de la deuxième contribution du premier volet. C'est la proposition d'une nouvelle architecture de système intégré à très grande échelle (VLSI) qui permet la mise en oeuvre de l'algorithme fuzzy ART pour le triage en temps réel rapide.

Une réalisation convenable en MSE radar devrait supporter un débit élevé, tout en garantissant un contrôle sur la précision, une tolérance adéquate au bruit, et des strategies fiables pour le test. Ceci nous conduit à proposer une approche VLSI

numérique pour la mise en oeuvre. Afin d'obtenir des solutions efficaces quant à la mise en oeuvre d'un circuit dédié VLSI numérique, l'algorithme du RNA fuzzy ART a été reformulé. Cette reformulation séquentielle permet de contourner les exigences d'un algorithme massivement parallèle, et donc d'éviter les coûts excessifs associés aux interconnexions. De plus, la séquence des opérations a été modifiée afin d'éviter la redondance de calculs, d'exploiter la ré-utilisation des résultats et d'augmenter l'efficacité de traitement des données.

Une architecture de système VLSI a été proposée pour la mise en oeuvre de l'algorithme reformulé dans des applications à débit élevé. Étant donné les contraintes qui s'imposent sur la technologie VLSI actuelle, ainsi que les dimensions du problème et la vitesse de traitement qui sont ciblées[1], un système numérique multi-puces a été choisi pour cette mise en oeuvre. L'architecture proposée pour le système est modulaire et cascadable en fonction des besoins de l'application. Elle comprend un comparateur global, ainsi qu'un ensemble de modules élémentaires identiques. Chaque module élémentaire permet d'émuler un certain nombre de neurones de sortie, chacun associé à une catégorie, et est réalisé par un circuit integré ou ASIC cascadable. L'architecture d'un module élémentaire est constituée d'un comparateur local, de diviseurs, de processeurs neuroniques et d'un bloc de mémoire SRAM.

Lorsqu'une impulsion est présentée au système, chaque module élémentaire produit indépendamment un neurone vainqueur au niveau local (parmi les neurones qu'il

[1] On s'est fixé une capacité maximum de 250 catégories d'émetteurs, une densité maximum entre 10^5 et 10^6 impulsions par seconde et des patrons d'entrée à $M = 16$ dimensions.

contient). À l'intérieur de chaque module élémentaire, le traitement est pipeliné grâce à une architecture systolique en anneau pour la comparaison rapide entre entrées et poids. L'ensemble des fonctions d'activation est donc généré très rapidement. L'architecture systolique permet d'émuler le traitement massivement parallèle des RNA avec de la circuiterie numérique. De plus, l'utilisation de mémoire SRAM à l'interne du ASIC minimise le temps d'accès aux poids neuroniques. Le comparateur global choisit le neurone vainqueur au niveau global (entre les vainqueurs locaux), puis il active le module élémentaire correspondant pour la phase d'apprentissage. La configuration systolique de ce module élémentaire permet d'ajuster les poids du neurone gagnant.

Un modèle pour l'analyse du coût AT été développé pour cette architecture dans le but d'évaluer l'impact du choix des paramètres (*e.g.*, le nombre de neurones par processeur neuronique) sur sa surface semiconducteur (A) et son temps de traitement (T). Étant donné des contraintes de performance pour l'application, ainsi qu'un choix de configuration d'architecture, ce modèle permet d'estimer rapidement la complexité surface-temps. Le modèle a été utilisé pour estimer la performance de l'architecture fuzzy ART pour le triage rapide en MSE radar.

De plus amples détails sur cette proposition sont présentés dans l'article suivant:

GRANGER, E., BLAQUIÈRE, Y., SAVARIA, Y., CANTIN, M.-A., et LAVOIE, P., "A VLSI architecture for fast clustering with the fuzzy ART neural network," *Journal of Microelectronic Systems Integration*, **5:1**, 3-18 (1997).

Une copie de cet article est fournie ici. Ensuite, la dernière section du chapitre aborde une discussion sur l'impact des résultats de cette contribution.

A VLSI architecture for fast clustering with fuzzy ART neural networks

Eric Granger[1], Yves Blaquière[2], Yvon Savaria[1], Marc-André Cantin[2] and Pierre Lavoie[3]

[1] Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C. P. 6079, Station "centre-ville," Montreal, Quebec, H3C 3A7, Canada.

[2] Department of Computer Science, Université du Québec á Montréal, C. P. 8888, Station A, Montreal, Quebec, H3C 3P8, Canada.

[3] Defence Research Establishment Ottawa, Department of National Defence, 3701 Carling Ave., Ottawa, Ontario, K1A 0Z4, Canada.

Abstract

Hardware implementation of the fuzzy ART neural network applied to a demanding real time radar signal clustering problem is investigated. To obtain efficient solutions for the implementation of this neural network with dedicated digital VLSI hardware, the network's algorithm is reformulated. A novel fuzzy ART system architecture which can implement this reformulated algorithm for high speed clustering problems is then proposed. This system architecture is composed of a global comparator and several identical elementary modules (EMs), each of which emulates a

number of neurons. The general architecture of each EM consists of a local comparator, dividers, neural processors, and a block of memory. This paper also outlines a cost analysis model for the estimation of area and processing time required with the proposed system architecture. This model offers a framework for rapid estimation of area-time complexity, given a VLSI system architecture configuration and a set of application constraints. As an illustration, this model is employed to estimate the performance of fuzzy ART hardware applied to the clustering of radar pulses in electronic warfare systems.

3.1 Introduction

On-line clustering of intercepted radar pulses is important for data reduction in electronic support measures (ESM) systems. The purpose of radar ESM is to search for, intercept, locate, and analyze radar signals in the context of military surveillance. It is an important preliminary step in all electronic warfare systems, which allows the evaluation of the countermeasures to be undertaken for self-protection [63]. The need for data reduction in radar ESM arises from the signal densities encountered in some theaters of operation, which can reach up to 10^6 radar pulses per second. Grouping radar pulses into categories corresponding to active radar emitters at the early stages of processing could considerably reduce the processing requirements, as well as the hardware cost of ESM systems [4] [39] [79]. Since the number and description of the radar emitters are *a priori* unknown, the radar pulses must be

grouped into categories based on their perceived similarity, a process which is called clustering [3] [43]. This process is hindered by the fact that pulses from one emitter are different from each other because of noise, and because of receiver, environmental and emitter variability [4].

In this paper, a neural network called fuzzy ART [21] is considered for clustering. The network is capable of fast, stable, and unsupervised learning of categories in response to non-stationary sources of arbitrary binary or analog patterns. This network is attractive for ESM applications, since it can cluster patterns autonomously, in real-time, without prior knowledge of the categories.

This paper presents a flexible and modular digital VLSI architecture for fuzzy ART neural networks suitable for very high data rate clustering applications. Flexibility and modularity allow adjusting system parameters to trade off area and speed. Also, due to the modularity, systems derived from the proposed architecture can be partitioned into several identical modules, which can be implemented in individually packaged chips, or as separate semiconductor dies interconnected in a multi-chip package. An area-time estimation model is provided to ease parameter selection, and to assess achievable semiconductor areas and data rates. Radar ESM is used as a case study to demonstrate the architecture and the use of this estimation model.

The paper is organized as follows. Fuzzy ART is reviewed as a neural network and then as an algorithm in the next section. In Section 3.3, the algorithm is reformulated to permit more efficient implementation without entailing a clustering performance

degradation. In Section 3.4, a multi-chip VLSI system architecture based on the reformulated algorithm is proposed. Finally, Section 3.5 outlines a model for the rapid estimation of semiconductor area and processing time, given a VLSI system architecture configuration and a set of application constraints.

3.2 Fuzzy ART neural network

Adaptive Resonance Theory (ART) neural networks [20] can develop stable recognition capability on-line by self-organization, in response to arbitrary sequences of input patterns. Several different neural models based on ART have been proposed: ART1, ART2, ARTMAP, ART3, etc. [23]. The fuzzy ART neural network proposed by Carpenter, Grossberg and Rosen [21] introduces modifications to ART1 (replaces crisp logic AND operators (intersection: $\cap$) with fuzzy logic AND operators (minimum: $\wedge$)), which allow for processing of analog input patterns, as well as binary ones.

3.2.1 Neural network model

The general structure of the fuzzy ART neural network is shown in Figure 3.1. It consists of two layers of neurons that are fully connected: a $2M$ neuron input or *comparison* layer (F1) and an N neuron output or *competitive* layer (F2). A weight value w_{ji} is associated with each connection, where the indices i and j denote the neurons that belong to the layers F1 and F2 respectively. The set of weights $\mathbf{W} =$

$\{w_{ji} : i = 1, 2, ..., 2M; j = 1, 2, ..., N\}$ encodes information that defines the categories learned by the network. These can be modified dynamically during network operation. For each neuron j of F2, the vector of adaptive weights $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{j2M})$ corresponds to the subset of weights ($\mathbf{w}_j \subset \mathbf{W}$) connected to neuron j. This vector $\mathbf{w}_j$ is named *prototype vector*, and it represents the set of characteristics defining the category j. Each prototype vector $\mathbf{w}_j$ is formed by the characteristics of the input patterns to which category j has previously been assigned through *winner-take-all* competitions. Figure 3.1 shows that the fuzzy ART neural network dynamics are governed by 2 subsystems. The *attentional subsystem* is responsible for proposing a winning category neuron, whereas the *orienting subsystem* accepts the proposed candidate or else reorients the search. Further details on these 2 subsystems, as well as on various gain control signals can be found in the literature [20], and are not discussed in this work.

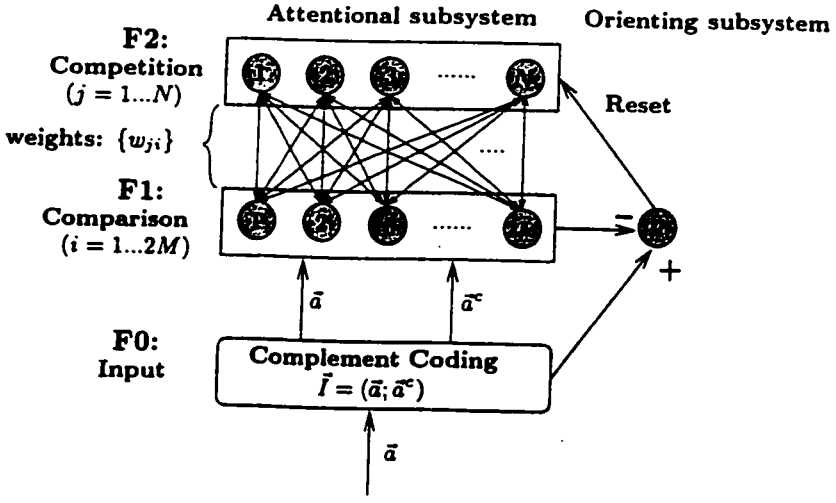


Figure 3.1: The fuzzy ART neural network.

3.2.2 Algorithmic description of fuzzy ART

The fuzzy ART network's functionality may be described by an algorithm [103] (as shown in Figure 3.2). This algorithm can be divided into five execution steps:

1. **Weights and parameters initialization:** Initially, all the neurons of F2 are uncommitted, and all weight values w_{ji} are initialized to 1. An F2 neuron becomes committed when it is selected for an input **a**. Then, the corresponding weights w_{ji} can take values expressed by a real number in the interval [0,1].

2. **Input vector coding:** When a new input vector $\mathbf{a} = (a_1, a_2, ... a_M)$ of M elements (where each element a_i is a real number in the interval [0,1]) is presented to the network, it undergoes a preliminary coding at layer F0. *Complement coding* of **a** results in a network input vector **I** of $2M$ elements such that: $\mathbf{I} = (\mathbf{a}; \mathbf{a}^c) = (a_1, a_2, ..., a_M; a_1^c, a_2^c, ..., a_M^c)$, with $a_i^c = 1 - a_i$. This coding is recommended [21] to prevent a category proliferation problem that may occur in analog ART networks [103].

3. **Category choice:** With each presentation of an input **I** to F1, the *choice function* $T_j(\mathbf{I})$ is calculated for each neuron j in F2:

$$T_j(\mathbf{I}) = \frac{|\mathbf{I} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|} \tag{3.1}$$

where $|\cdot|$ is the norm operator ($|\mathbf{w}_j| = \sum_{i=1}^{2M} |w_{ji}|$), $\wedge$ is the fuzzy logic AND operator ($\mathbf{I} \wedge \mathbf{w}j = (\min(I_1, w_{j,1}), \min(I_2, w_{j,2}), ..., \min(I_{2M}, w_{j,2M}))$), and α is

a user-defined *choice parameter* such that $\alpha > 0$. F2 is a *winner-take-all* competitive layer, where the winner is the neuron $j = J$ with the greatest value of activation T_j for the input $\mathbf{I}$ ($T_J = \max\{T_j : j = 1...N\}$). If the same T_J value is obtained by two or more neurons, the one with the smallest index j wins. The winning neuron J is retained for Steps 4 and 5.

4. **Vigilance test:** This step serves to compare the similarity between the prototype vector of the winning neuron $\mathbf{w}_J$ and input $\mathbf{I}$, against a user-defined *vigilance parameter* ρ, through the following test:

$$\frac{|\mathbf{I} \wedge \mathbf{w}_J|}{|\mathbf{I}|} \geq \rho \tag{3.2}$$

where $\rho = [0, 1]$. This comparison is carried out on layer F1: the winning neuron J transmits its learned expectancy, $\mathbf{w}_J$, to F1 for comparison with $\mathbf{I}$. If the vigilance test (Eq. 3.2) is passed, then neuron J becomes *selected* and is allowed to adapt its prototype vector (Step 5). Otherwise, neuron J is deactivated for the current input $\mathbf{I}$: T_J is set equal to -1 for the duration of the current input presentation. The algorithm searches through the remaining F2 layer neurons (Steps 3 and 4), until some other neuron J passes the vigilance test. If no committed neuron from the F2 layer can pass this test, an uncommitted neuron is selected and undergoes prototype vector update.

5. **Prototype vector update:** The prototype vector of the winning neuron J is updated according to:

$$\mathbf{w}'_J = \beta(\mathbf{I} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J \tag{3.3}$$

where β is a user-defined *learning rate parameter* such that $\beta = [0, 1]$. The algorithm can be set to slow learning, with $0 < \beta < 1$, or to fast learning, with $\beta = 1$. Once this update step is accomplished, the network can process a new input vector (Step 2).

The flowchart representation in Figure 3.2 summarizes the fuzzy ART algorithm. It is assumed that the number of neurons N in layer F2 is fixed but arbitrary.

The fuzzy ART algorithm was simulated with real radar signal data. The data set used was collected by the Defence Research Establishment Ottawa, and it contains 800 radar pulses from 12 different emitters. Each radar pulse in this set is characterized by 8 complex features. The network inputs **a** therefore have $M = 16$ elements in the interval [0,1]. Based on these functional simulations, the following set of network parameters: $\beta = 1$, $\alpha = 1$ and $\rho = [0.75, 0.9]$ was determined. These values correspond to a satisfactory trade-off between clustering quality and algorithm simplicity. Setting $\beta = 1$ in Eq. 3.3 allows for fast learning with a simple prototype vector update equation. Considering the intended application, and the simulation results, it was decided that the network would only be implemented for the fast learning

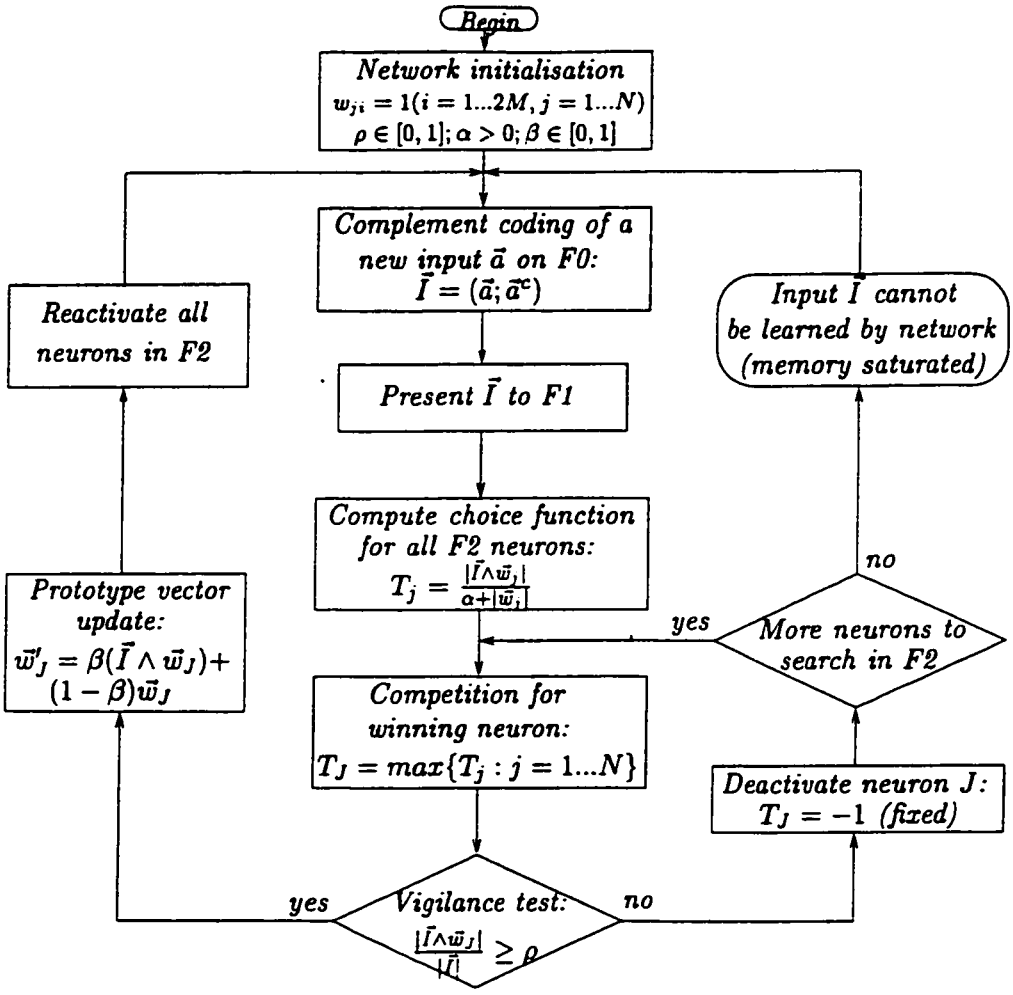


Figure 3.2: Flowchart representation of the fuzzy ART algorithm.

case. Moreover, since our objective is to obtain an effective, resolution-limited digital hardware implementation, it was also determined that the minimum word length b required to represent the values w_{ji}, I_i, and ρ is $b = 11$ bits. This produces errors within 2% of the clustering results obtained with floating-point values.

3.2.3 Hardware realizations of ART neural networks

The authors could find no reference to previous efforts to realize the fuzzy ART neural network in hardware. However, several attempts were made to implement the ART1 [20] (binary input) neural network and, as mentioned earlier, ART1 and fuzzy ART processing is very similar. Different classes of ART1 realizations have been reported. The first class fully or partially implements the time-domain nonlinear differential equations that were originally used to describe ART1 (for example, [93] [131] [132]). Indeed, according to the original modeling of ART1, a set of differential equations describe the dynamics of each neuron and synapse [20]. This network formulation and the derived realizations demand a great deal of computational power. The second class of realization consists in replacing the differential equations of the original model with a set of steady-state nonlinear algebraic equations, which can be executed sequentially to produce the exact same result (for example, [29] [112] [113] [125] [144] for ART1, and [80] for ART2). These algorithmic descriptions require artificial sequencing of the events in the original model, but they are less computationally expensive, allowing for a higher potential throughput. Some of the implementations are based on analog, or mixed analog/digital circuitry (for example [93], [125] [131], [132]). Others are optical based implementations (for example [29], [80] [144]). Yet others are realized as software executing on existing hardware (for example [114]).

Our goal is to realize a class 2 implementation of fuzzy ART with a dedicated

digital VLSI system for maximum throughput with a high degree of accuracy. Rao et al. [112] [113] have reported on a class 2 ART1 network implementation based on digital hardware. The authors exploit a multiplexing technique that permits simultaneous matching of inputs to prototype vectors, and generation of F2 activation values. They reduce ART1's processing to a linear search, best match algorithm. The search's functionality is implemented with a pipelined associative memory architecture: a pipeline of identical, back-to-back processing elements, capable of simultaneously matching several inputs against stored prototype vectors. The architecture achieves a temporal parallelism (since many inputs can be processed at once), which results in a very high throughput for small scale problems (16 possible categories, 9 bit input vector, etc.). This architecture is not appropriate for the target ESM applications where the number of categories is in the hundreds, and where input and prototype vectors contain $2M \times b = 32$ elements $\times$ 11 bits = 352 bits. Moreover, the architecture is somewhat incomplete since it lacks the mechanism needed for the update of already committed categories. These shortcomings have motivated our desire to propose a new system architecture capable of high throughput with a minimum dependence on the number of categories, which incorporates all features of fuzzy ART processing.

3.3 Reformulated fuzzy ART algorithm

It would be possible to directly implement the fuzzy ART neural network's algorithm as described in Figures 3.1 and 3.2. However, neuron interconnections would

occupy excessive semiconductor area ($O(M \cdot N)$), and would eventually limit the size of a feasible digital hardware implementation of the network. Furthermore, simulations with real radar data have shown that a large proportion of the prototype vectors remain unchanged during the update operations. Prior to its implementation in dedicated hardware, the fuzzy ART algorithm (given in Figure 3.2) was reformulated to circumvent its massively parallel processing requirements, and to exploit its inherent potential for re-use of data. The reformulated fuzzy ART algorithm helps to maximize throughput, to minimize communication costs, and is easier to implement in digital VLSI hardware for high throughput applications.

Although the reformulated algorithm is functionally equivalent to the original one, it is expressed as a stream of operations that can be carried out sequentially. Other modifications to the original algorithm are based on observations of the fuzzy ART algorithm. Computation of the choice functions (a division, T_j, for every committed neuron j), and their components ($|\mathbf{I} \wedge \mathbf{w}_j|$ and $|\mathbf{w}_j|$), constitutes a substantial proportion of the total processing effort. Fortunately, once the elements $|\mathbf{I} \wedge \mathbf{w}_j|$ and $|\mathbf{w}_j|$ have been computed, their values can be re-used on several occasions. This permits a more efficient organization of the algorithm's processing. In addition, there is no need to compute the choice function T_j of a neuron j that would not pass the vigilance test. This modification circumvents the original algorithm's search process, since the neuron $j = J$ with the greatest T_j value that can pass the vigilance test is readily selected. The vigilance test should thus occur as soon as the component

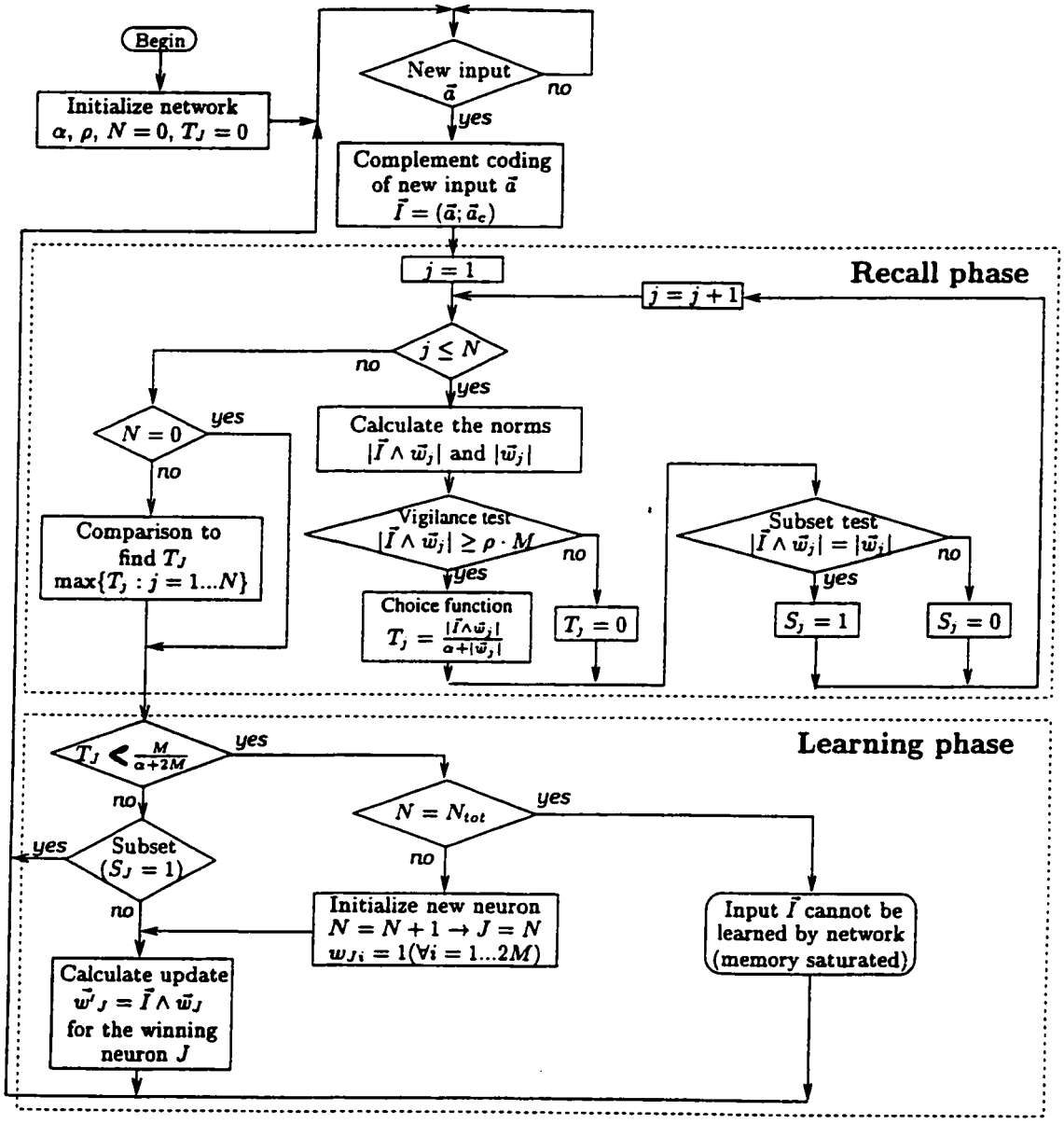


Figure 3.3: Flowchart representation of the reformulated fuzzy ART algorithm.

$|\mathbf{I} \wedge \mathbf{w}_j|$ is available for a neuron j. Further, since complement coding is an input normalization such that, by definition, $|\mathbf{I}| = |(\mathbf{a}; \mathbf{a}^c)| = M$ [21], the vigilance test reduces to $|\mathbf{I} \wedge \mathbf{w}_j| \geq \rho \cdot M$. In parallel with the vigilance test, it is convenient to carry out a *subset test*, since it depends only on the elements $|\mathbf{I} \wedge \mathbf{w}_j|$ and $|\mathbf{w}_j|$. This test detects when a prototype vector $\mathbf{w}_j$ is a subset of $\mathbf{I}$, that is, when $|\mathbf{I} \wedge \mathbf{w}_j| = |\mathbf{w}_j|$. If this neuron j is chosen and passes the vigilance test, then its prototype vector $\mathbf{w}_J$ remains unchanged during the learning phase, since $\mathbf{w}'_J = \mathbf{I} \wedge \mathbf{w}_J = \mathbf{w}_J$ in fast learning mode ($\beta = 1$). Therefore, the learning phase can be bypassed, and the network speed increased.

These observations lead to a reformulation of the fuzzy ART algorithm, as shown in Figure 3.3 for the fast learning case. The algorithm sequentially processes committed neurons only, and N now represents the number of committed neurons, which starts from 1 and increases progressively as learning takes place. The reformulated algorithm is divided into two parts: a *recall phase* and a *learning phase*. During the recall phase, the values T_j and S_j are computed for all committed neurons. If a neuron j is a subset choice of $\mathbf{I}$, then S_j is set equal to 1. If a neuron j passes the vigilance test for $\mathbf{I}$, then the choice function is computed and $T_j > 0$, otherwise T_j is set equal to 0. At the end, the winning neuron J is chosen. During the learning phase, in the case where $T_J \geq M/(\alpha + 2M)$, the winning neuron J corresponds to a committed category that passes the vigilance test. If J is a subset choice for $\mathbf{I}$ ($S_J = 1$), then the prototype vector update is bypassed; otherwise ($S_J = 0$), the prototype vector

$\mathbf{w}_J$ is updated according to Eq. 3.3. In the case where $T_J < M/(\alpha + 2M)$, no committed category has passed the vigilance test. If the maximum number of neurons is not attained ($N < N_{tot}$), then an uncommitted neuron $J = N + 1$ is assigned to **I**; otherwise ($N = N_{tot}$), the network is unable to categorize **I** because its memory is saturated. After the learning phase, the network is ready to accept another input vector.

3.4 Fuzzy ART system architecture

In this section, we present a dedicated VLSI system architecture for the reformulated fuzzy ART algorithm of Figure 3.3. Our main objective is to execute this algorithm at high speeds. It is true that analog implementations can be more compact (since neuron interconnections consume less semiconductor area), and have greater potential for high speed processing. However, digital implementations offer higher computational accuracy, more flexibility in design, better resistance to noise and interference, more reliable testability methods, and less susceptibility to process variations among devices [111]. It is prohibitively costly or not currently feasible to construct a relatively large massively parallel digital network on a single integrated circuit, as with the analog case. Therefore, any attempt to achieve a high degree of parallelism results in very large circuits, whose area is dominated by interconnections [112]. Given the nature of the application, a multi-chip digital implementation was selected in order to ensure that the required accuracy can be maintained.

The actual number of neurons is not necessarily the predominant constraint when considering a digital system implementation, since a neural processor (NP) can sequentially process, and thus emulate, many neurons. Yet, the system performance deteriorates as the number of neurons emulated per NP grows (due to increased processing time and memory size). The parallel nature of the intensive computations carried out by this neural network means that they execute very slowly on sequential processor architectures [77]. The potential for success in terms of throughput of the digital system approach thus highly depends on the feasibility of the required level of parallel execution [88].

Bearing in mind ESM applications, we seek a fuzzy ART network with the following parameters: $2M = 32$ neurons in the F1 layer, $N_{tot} \geq 250$ category neurons in the F2 layer, fast learning ($\beta = 1$), and $b = 11$ bit word length. For proof-of-concept, the fuzzy ART network should accept and categorize a new input vector every 2μs (500000 patterns per second). Given the constraints imposed by current VLSI technology, the high speed processing requirements, and the number of category neurons needed, a dedicated digital VLSI hardware implementation of the network's processing should preferably be distributed over several integrated circuits. For the remainder of this section, a multi-chip fuzzy ART system architecture is presented. In the next section, a detailed area-time estimation shows the performance achievable with the proposed system.

3.4.1 Proposed system architecture

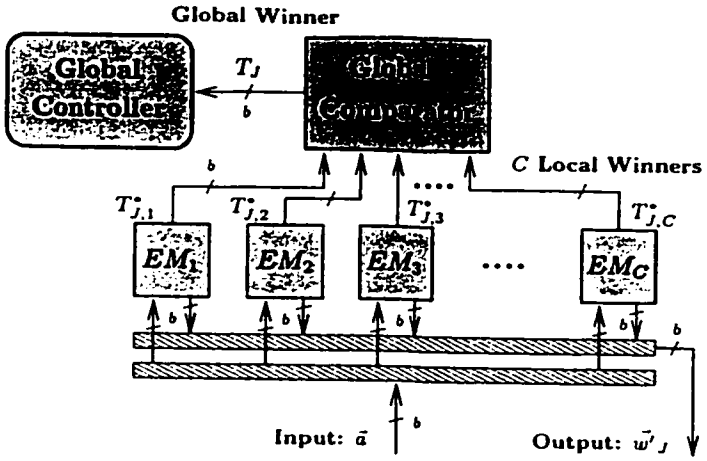


Figure 3.4: Architecture of fuzzy ART system.

The proposed fuzzy ART system architecture (Figure 3.4) is composed of a global comparator, a global controller, and C identical elementary modules (EMs), each of which emulates $N = N_{tot}/C$ neurons. The C EMs can be implemented either in individually packaged chips, or as separate semiconductor dies interconnected together in a multi-chip package. Each EM determines the T_J^* of its local winner J^*, for an input a of M elements, considering the N neurons it emulates. The global comparator then selects the global winner J, and enables the corresponding EM for update. The general data path architecture of an EM (shown in Figure 3.4.2) consists essentially of a local controller, a local comparator, D fixed-point dividers, P neural processors (NPs) - each emulating N/P category neurons -, P/D NPs per divider, and D blocks of random access memory (RAM) for storage of the N prototype vectors.

To reduce the control burden, an entire EM is considered to be committed if any

one of the N neurons it emulates is committed. Recall that uncommitted neurons have been initialized such that $|w_j| = 2M$ (since $w_{ji} = 1\ \forall j, i$). When needed, a new EM is activated, and computes its local T_j^* based on the results from all its internal neurons, even though these neurons may not all be committed. This emulates fuzzy ART's orderly selection of uncommitted neurons: the first uncommitted neuron in an EM is selected only when $T_j < M/(\alpha + 2M)$ for all committed neurons ($\forall j = 1, 2, ..., N$). That is, according to Eq. 3.1, an uncommitted neuron cannot be selected unless all committed neurons have failed the vigilance test or are less active than $M/(\alpha+2M)$.

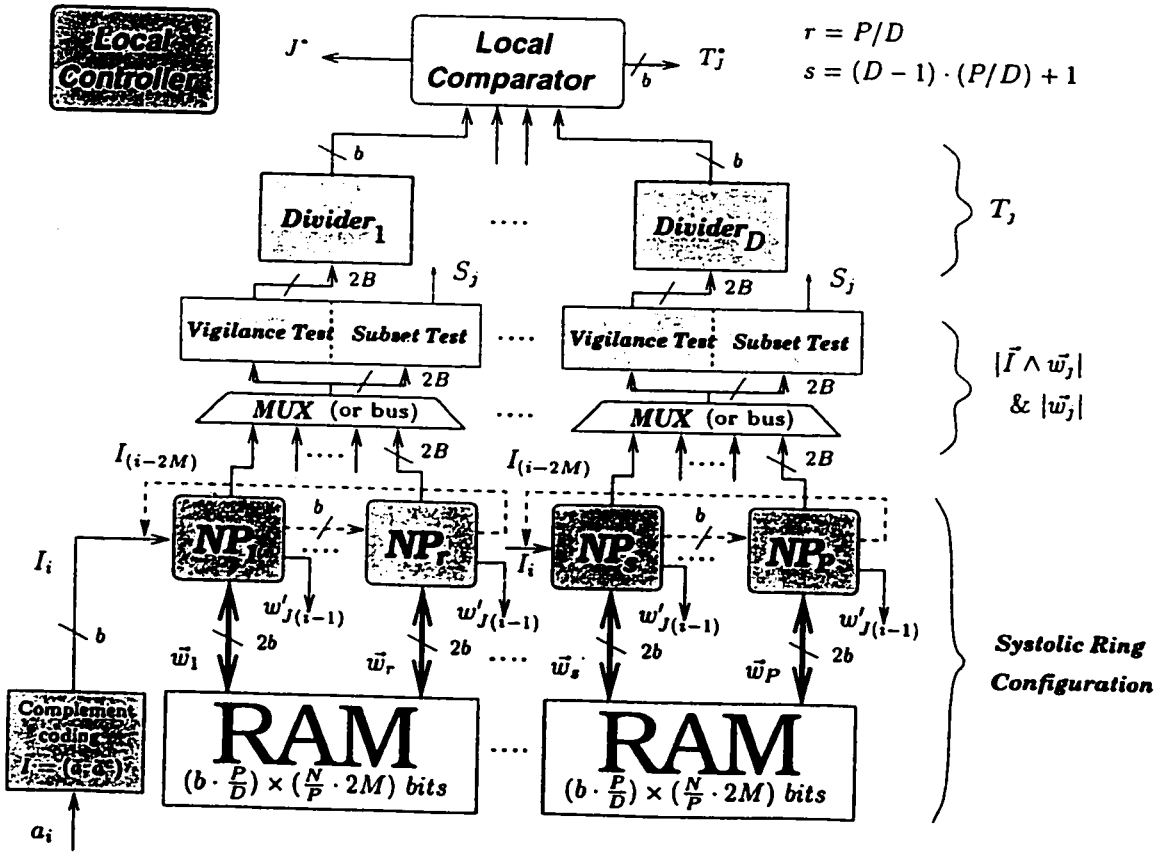


Figure 3.5: Architecture of an elementary module (EM).

The speed of such a system depends mostly on the number of neurons emulated, and on access time to the prototype vectors [111]. Storage of 352-bit prototype vectors for $N_{tot} \cong 250$ neurons requires about $88k$bits of RAM. In general, internal memory is preferred for fast access to weights and moderate I/O pin count, whereas external memory is preferred to obtain a maximum number of emulated neurons. Considering that memory is only one part of the system, for a proof-of-concept implementation with the desired number of neurons, it would be preferable to use a cascade of several identical chips with their own internal RAM. If there are C EMs, then each EM stores the weights of $N = N_{tot}/C$ neurons (with an area $N \cdot 2M \cdot b \cdot A_{membit}$); and if there are P NPs per EM, then the memory associated with each NP contains the weights of N/P neurons. On-chip storage permits very fast processing distributed over C identical EMs with moderate I/O bandwidth requirements. This also allows to balance areas devoted to memory and to functional circuit [111].

Figure 3.6 presents the basic structure of a NP module. In agreement with the reformulated algorithm, a NP can operate in the recall phase or the learning phase. During the recall phase, the value $|\mathbf{w}_j \wedge \mathbf{I}|$ is calculated for every committed F2 neuron j, and during the learning phase, the prototype vector $\mathbf{w'}_J$ (and $|\mathbf{w'}_J|$) of the globally winning neuron is updated. Therefore, the system architecture performs the recall and learning phases of the reformulated fuzzy ART algorithm sequentially using the same hardware. The system's behavior is described for both of these phases in the following subsections.

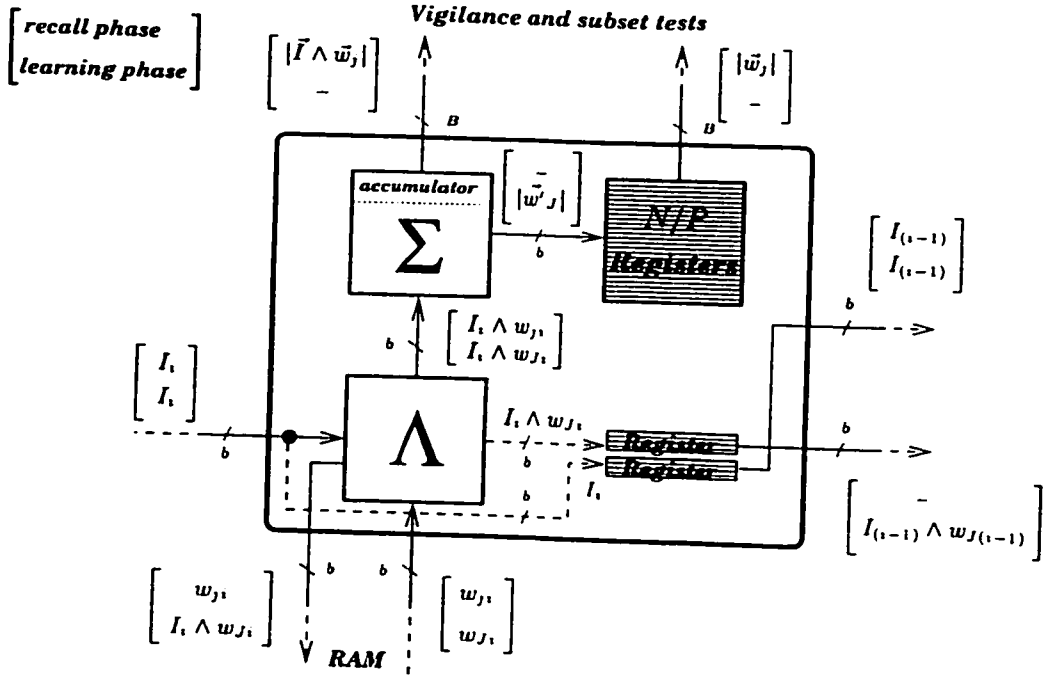


Figure 3.6: The basic structure of a NP module consists of a "min" comparator ($\wedge$), an adder-accumulator ($\sum$), and registers to compute the values $|\mathbf{I} \wedge \mathbf{w}_j|$, $\mathbf{w}'_j$ and $|\mathbf{w}'_j|$. Note the dual labels for recall (top) and learning (bottom) phases respectively.

3.4.2 Recall phase

Complement coding is done inside each EM circuit to reduce I/O requirements since it is a relatively simple operation. This coding has little impact on the system throughput, since the sequential complement coding of the M elements of input vector a overlaps the NPs processing. Accordingly, the delay is equal to t_{cc} and corresponds to the complement coding of the first element of a. As an input a is being encoded inside an EM, the recall phase forms a data processing pipeline starting from the systolic ring configuration, and ending with the global comparators.

A *systolic ring configuration* [76] [77] is proposed for the NPs, to exploit the repet-

itive and regular nature of the $|\mathbf{I} \wedge \mathbf{w}_j|$ computation, and to increase the potential for parallel execution with VLSI technology. This configuration pipelines processing through a chain of locally connected NPs linked with unidirectional connections (see Figures. 3.4.2 and 3.7). The shifting of data between NPs offers a high total inter-NP throughput by using local communications only, yielding a better balance between communications and computations [87]. Further, such regular communications minimize I/O requirements, which allows to implement a large number of NPs per EM. It also simplifies the connections between each neuron and the rest of the circuit. Each systolic ring is connected to one divider.

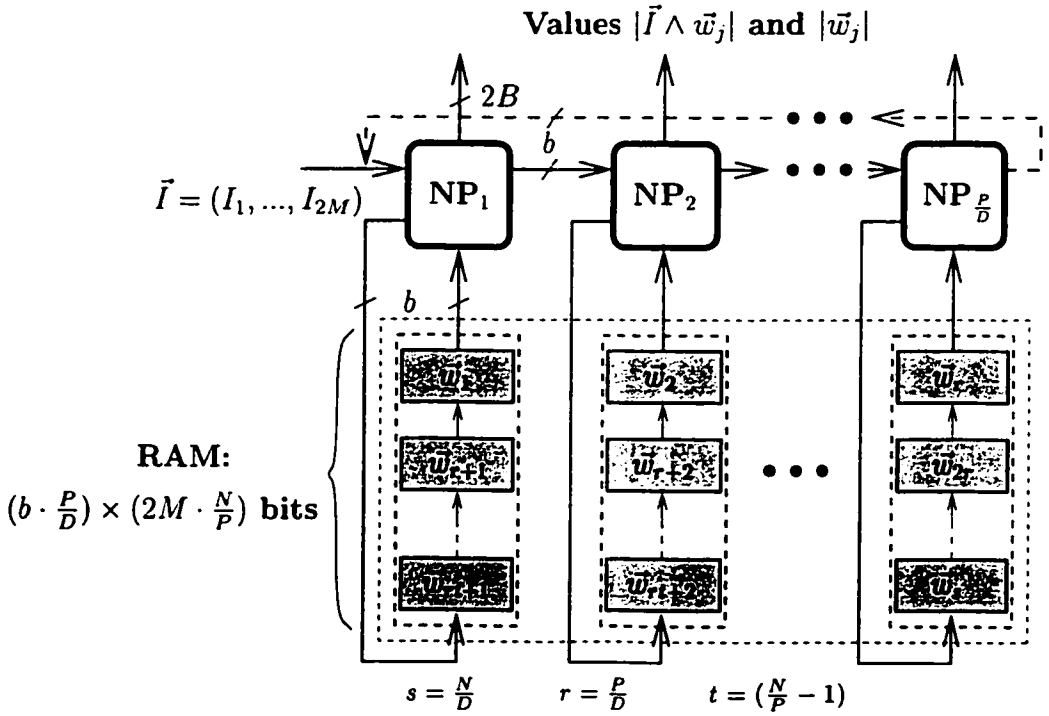


Figure 3.7: Systolic ring configuration used with NPs.

Every NP occupies an area A_{NP}, and communicates with a local memory that

contains the prototype vectors of N/P category neurons of the F2 layer. The $2M$ elements of the complement coded input vector $\mathbf{I}$ are shifted right, in natural order sequence (I_1, I_2, etc.), through the NPs in a ring, while the $2M$ elements of every prototype vector, the weights, are sequentially read from RAM and shifted upwards. Respective input vector and weight elements for each neuron j are combined in a series of $2M$ minimum operations, $I_i \wedge w_{ji}$, so that each NP can accumulate its output value, $|\mathbf{I} \wedge \mathbf{w}_j|$. After $2M + (P/D - 1)$ cycles, all the output values of the systolic rings are ready.

Figure 3.7 shows the interaction between $\mathbf{I}$ and the prototype vectors in one systolic ring. Table 3.1 shows an example of NP processing during the recall phase with $M = 16$, $P/D = 4$ and $N/P = 5$. Notice that for each input $\mathbf{I}$, N/P cycles around each ring are required to process the entire set of emulated neurons. Figure 3.6 shows that the value $|\mathbf{w}_j|$ (computed during the learning phase) is also generated from the NPs in the same sequence as its respective $|\mathbf{w}_j \wedge \mathbf{I}|$. The total time required to process the N/D neurons associated with a divider D is no longer than $[2M(N/P) + (P/D - 1)] \cdot t_{rep}$. The systolic ring response time, t_{rep}, is defined by $t_{rep} = \max\{t_{mem}, t_{NP}\}$, where t_{mem} is the memory access time and t_{NP} is the time needed to compute a "min" comparison ($I_i \wedge w_{ji}$) and an addition in a NP. For simplicity, we assume for the memory access time that $t_{mem} \simeq t_{\text{read}} \simeq t_{\text{write}}$. The factor $2M(N/P)$ is the number of cycles required to compute the norms $|\mathbf{I} \wedge \mathbf{w}_j|$ in the NPs for N/D neurons, whereas $P/D - 1$ is the latency for filling the pipe of the D systolic

rings. We suppose that $1 \leq P/D \leq 2M$. thus the divider used cannot accept results faster than they are produced by the ring of NPs. The block of memory required for each set of P/D NPs linked to a divider is $2M \cdot N/P$ words deep. each of which contains P/D elements of b bits wide. Each NP requires a datapath with a minimum width of $B = b + \log_2(2M)$ bits to prevent overflows with the norm values $|\mathbf{I} \wedge \mathbf{w}_j|$ and $|\mathbf{w}_j|$, since there are $2M$ accumulate operations to be done on b bit words. Considering the relative simplicity of the NPs, and the sequential nature of the processing they perform, several NPs can be assigned to a divider (Figure 3.5). Globally broadcasted signals to each NP control the sequence of operations in the ring.

The values $|\mathbf{w}_j \wedge \mathbf{I}|$ and $|\mathbf{w}_j|$ are transferred from each NP to its respective divider, after passing through vigilance and subset tests, which can conveniently be carried out in parallel. If $t_{rep} \geq t_{comp} = \max\{t_{vig}, t_{sub}\}$, the delay needed for these parallel tests (t_{vig} and t_{sub}) can be absorbed as a simple overhead cycle in the data processing pipe. Each vigilance test module contains the value $\rho \cdot M$ needed to perform the required comparison. It is possible, during processing, to adjust the value ρ in the range $[0, 1]$ to adapt category discrimination as a function of the input environment. If the value $|\mathbf{w}_j \wedge \mathbf{I}|$ of a neuron j passes the vigilance test, then this value is passed on to the divider; otherwise, T_j is set to 0, eliminating j from the competition. The corresponding value S_j is retained for use during the learning phase. The results of the subset test for an entire EM can be stored, for example, in the local controller with D registers of N/D bits each.

Fixed-point dividers are used to compute the choice functions from the numerator $|\mathbf{w}_j \wedge \mathbf{I}|$ and the denominator $(|\mathbf{w}_j| + \alpha)$. Note that the value $(|\mathbf{w}_j| + \alpha)$ is computed as a preliminary step in parallel with the subset and vigilance tests. The numerator $|\mathbf{w}_j \wedge \mathbf{I}|$ and the denominator $(|\mathbf{w}_j| + \alpha)$ are represented with $B = b + \log_2(2M)$ and $B + 1$ bits respectively, and the final result T_j is b bits wide.

The division efficiency is very important to the system's performance: it must be reasonably fast, without sacrificing the accuracy of the results. To maximize throughput, each divider (of delay t_D and area A_D) contains E pipeline stages (of delay $(t_{div} + t_{latch})$) to reduce the clock period. Evenly spread levels of pipelining registers (of delay t_{latch} and area A_{level}) ensure that the divider's critical path delay satisfies $(t_{div} + t_{latch}) = t_D/E \leq t_{rep}$, where E is the number of pipeline stages. With such a pipelining, the NP results are fed sequentially to dividers having E pipeline levels, generating a result T_j after a delay of $t_D = E \cdot t_{rep}$.

Tableau 3.1: Example of NP allocations in a systolic ring during the recall phase with $M = 16$, $P/D = 4$ and $N/P = 5$ (a block of $N/D = 20$ neurons). The results $|\mathbf{w}_j \wedge \mathbf{I}|$ are sequentially computed in the NPs as the inputs I_i and w_{ji} are shifted through. The values $|\mathbf{w}_j|$ (computed during the learning phase) are also generated by the NPs in the same sequence as their respective $|\mathbf{w}_j \wedge \mathbf{I}|$.

Time	**Inputs I**				**Inputs $\mathbf{w}_j$**				**Outputs $\|\mathbf{I} \wedge \mathbf{w}_j\|$**			
(Cycles)	NP_1	NP_2	NP_3	NP_4	NP_1	NP_2	NP_3	NP_4	NP_1	NP_2	NP_3	NP_4
1	I_1				$w_{1,1}$							
2	I_2	I_1			$w_{1,2}$	$w_{2,1}$						
3	I_3	I_2	I_1		$w_{1,3}$	$w_{2,2}$	$w_{3,1}$					
4	I_4	I_3	I_2	I_1	$w_{1,4}$	$w_{2,3}$	$w_{3,2}$	$w_{4,1}$				
5	I_5	I_4	I_3	I_2	$w_{1,5}$	$w_{2,4}$	$w_{3,3}$	$w_{4,2}$				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$(1 \cdot 2M) + 0$	I_{32}	I_{31}	I_{30}	I_{29}	$w_{1,32}$	$w_{2,31}$	$w_{3,30}$	$w_{4,29}$	$\|\mathbf{I} \wedge \mathbf{w}_1\|$			
$(1 \cdot 2M) + 1$	I_1	I_{32}	I_{31}	I_{30}	$w_{5,1}$	$w_{2,32}$	$w_{3,31}$	$w_{4,30}$		$\|\mathbf{I} \wedge \mathbf{w}_2\|$		
$(1 \cdot 2M) + 2$	I_2	I_1	I_{32}	I_{31}	$w_{5,2}$	$w_{6,1}$	$w_{3,32}$	$w_{4,31}$			$\|\mathbf{I} \wedge \mathbf{w}_3\|$	
$(1 \cdot 2M) + 3$	I_3	I_2	I_1	I_{32}	$w_{5,3}$	$w_{6,2}$	$w_{7,1}$	$w_{4,32}$				$\|\mathbf{I} \wedge \mathbf{w}_4\|$
$(1 \cdot 2M) + 4$	I_4	I_3	I_2	I_1	$w_{5,4}$	$w_{6,3}$	$w_{7,2}$	$w_{8,1}$				
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$(5 \cdot 2M) + 3$												$\|\mathbf{I} \wedge \mathbf{w}_{20}\|$

Following the divisions, the resulting T_js are fed to a local comparator. The maximum T_j^* in the current cycle is compared to the maximum T_j^* of the previous cycles, until all the T_j^*s have been compared, and a *local winner* J^* has been found. The index J^* and the value T_J^* of the local winner are passed on to the local controller for global comparison, yielding a *global winner* J. The EM containing this global winner J is then activated for the learning phase. The local and global comparisons can be implemented using $\lceil \log_2(D) \rceil$ and $\lceil \log_2(C) \rceil$ level binary trees of $(D-1)$ and $(C-1)$ comparators, with a processing time that grows as $O(\log_2(D))$ and $O(\log_2(C))$ respectively. If comparison trees are used, the worst case processing delay for finding a local winner is $\lceil \log_2(D) \rceil \cdot t_{lcom}$, and for finding a global winner, $\lceil \log_2(C) \rceil \cdot t_{gcom}$.

3.4.3 Learning phase

Rather than recomputing the $|\mathbf{w}_j|$ for all the neurons in use at every recall phase, only the $|\mathbf{w}_J|$ of the updated neuron is recomputed, using the existing NP hardware. This $|\mathbf{w}_J|$ updating operation is done during the *learning phase*, while updating the actual $\mathbf{w}_J$. During this phase, the prototype vector $\mathbf{w}_J$ of the winning neuron is updated by its respective NP in $4M$ cycles: the weights w_{Ji} (with $i = 1, 2, ..., 2M$) are loaded from memory, processed in the NP, and then written back to memory. This update is bypassed when J is a subset choice for $\mathbf{I}$ (that is, if $S_j = 1$). The processing of the learning phase is similar to that of the recall phase, except that only the winning neuron's prototype elements are shifted upwards through the associated

NP, in sequence with $\mathbf{I}$, to compute $\mathbf{w}'_J = \mathbf{w}_J \wedge \mathbf{I}$: all the other NPs simply shift the input through. At the same time, the elements w'_{J_i} are sequentially shifted outwards from the NP (as shown in Figure 3.6), out of the EM, and then out of the system on a b bit bus (see Figure 3.4). This allows external monitoring of new and updated categories, in real time. This feature is essential if the clustering results are to be communicated to the rest of the system for real time decision making.

For the learning phase, the architecture requires, at most, a delay of $t_{learn} = [4M + \lceil P/D - 1 \rceil] \cdot t_{rep}$, where $(P/D - 1) \cdot t_{rep}$ is an additional latency due to input vector shifting through P/D NPs (in the worst case, J is a multiple of P/D). To reduce the delays of this phase, the EMs can begin updating their local winners while the global comparison is being done. The local results $\mathbf{w}'_J$ can be stored temporarily. If the local winner is not the global winner, the update procedure is interrupted, and the local winner's weights remain unchanged. Otherwise (i.e. if the local winner is the global winner), the $\mathbf{w}'_J$, which is being held in a temporary buffer, can be transferred to replace $\mathbf{w}_J$. This minimizes the influence of the number of chips, C, on the system's performance, but entails additional EM controller complexity and memory.

3.5 Choice of system configuration

This section presents an area-time estimation model for the system architecture described above, and shows its application to a radar ESM problem. The development of this model was motivated by a desire to determine the impact of parameter choices

on the system's processing rate and area. Given the performance constraints of an application, this model allows fast analysis of trade-offs, resulting in the selection of an appropriate system configuration.

3.5.1 Area-time estimation model

A cost function, $A_{tot} \cdot t_{tot}$, that we wish to minimize, is used to assess the system architecture considering a given set of parameters. This function is the product of A_{tot}, the area of the system's datapath, and t_{tot}, the time required to process an M element input vector **a**.

The area A_{tot} corresponds to the sum of global and local comparator areas, divider areas, NP areas, and the area of the memory for prototype vectors. The interconnect area, and a few small modules are neglected. The equation for computing A_{tot} is:

$$\begin{aligned} A_{tot} &= A_{\text{Global Comp.}} + C\,[A_{\text{Local Comp.}} + A_{\text{Dividers}} + A_{\text{NPs}} + A_{\text{Memory}}] \\ &= (C-1)A_{gcom} + C[(D-1)A_{lcom} + D(A_D + EA_{level}) + \\ &\quad PA_{NP} + 2MNbA_{membit}] \end{aligned} \tag{3.4}$$

The time t_{tot} is approximated by making abstraction of communication time and chip I/O delays. It is equal to the sum of the delays associated with the recall and learning phases: $t_{tot} = t_{recall} + t_{learn}$. The parameter t_{recall} consists essentially of the sum of the delays required for the norm computations in the NPs ($|\mathbf{I} \wedge \mathbf{w}_j|$) and the

latency of the complement coding of one element of **a**, one subset or vigilance test, one complete division, one local and one global comparison:

$$t_{recall} = t_{cc} + \left[2M\lceil\frac{N}{P}\rceil + (\lceil\frac{P}{D}\rceil - 1)\right] \cdot t_{rep} + t_{comp} + E(t_{div} + t_{latch}) + \lceil\log_2(D)\rceil \cdot t_{lcom} + \lceil\log_2(C)\rceil \cdot t_{gcom}$$

The parameter t_{learn} consists of the delay associated with the prototype vector updating operation in the corresponding NP ($\mathbf{w}'_J = \mathbf{I} \wedge \mathbf{w}_J$):

$$t_{learn} = [(4M + \lceil\frac{P}{D}\rceil - 1)] \cdot t_{rep}$$

In a single clock synchronous system, the processing rate can be analyzed using a clock period $t_{cycle} = \max\{\ t_{cc},\ \max\{t_{mem}, t_{NP}\},\ \max\{t_{vig}, t_{sub}\},\ (t_{div} + t_{latch}),\ t_{lcom},\ t_{gcom}\ \}$. In the worst case, processing of an input vector **a** starts with complement coding, and ends with the update of the global winner's prototype vector:

$$\begin{aligned} t_{tot} &= t_{recall} + t_{learn} \\ &= t_{cycle} \cdot \left[2 + 2M\lceil\frac{N}{P}\rceil + (\lceil\frac{P}{D}\rceil - 1) + E + \lceil\log_2(D)\rceil + \lceil\log_2(C)\rceil\right] + \\ &\quad t_{cycle} \cdot \left[4M + \lceil\frac{P}{D}\rceil - 1\right] \\ &= t_{cycle} \cdot \left[2M(\lceil\frac{N}{P}\rceil + 2) + 2\lceil\frac{P}{D}\rceil + E + \lceil\log_2(D)\rceil + \lceil\log_2(C)\rceil\right] \end{aligned} \tag{3.5}$$

Notice that ceiling functions are used with the N/P and P/D ratios in Eq. 3.5, since these ratios are only valid as integer values. Clearly, Eqs. 3.4 and 3.5 can be very

useful for the analysis of performance trade-offs in critical applications. For example, they show that, for a constant number of neurons N_{tot} and chips C, an increase in the number of dividers D decreases P/D (and/or N/D), and consequently decreases the number of cycles needed for processing, but it also increases the area of an EM. For an input vector **a** of M elements, N/P is a critical factor for the system's performance: indeed t_{tot} grows rapidly with N/P. Eq. 3.5 also justifies the need for internal memory in fast on-line applications. If the memory's access time also includes chip I/O delays, the system's performance is significantly degraded (since t_{mem} is associated with the factor $N/P \cdot 2M$ in t_{recall}). A relatively large number of neurons N_{tot} may also be emulated by cascading several EM chips, since C is a less significant factor in the system's overall speed. Preliminary results on estimated delays (see Table 3.2) led us to believe that: $t_{cycle} = \max\{t_{mem}, (t_{div} + t_{latch})\} = \max\{t_{mem}, t_D/E\}$, where $t_{rep} = t_{mem}$ is the largest fixed delay, and $(t_{div} + t_{latch})$ varies with E. Beyond some value $E = E_{limit} = \lceil t_D/(t_{cycle} - t_{latch}) \rceil$, t_{cycle} is minimized since it depends entirely on the fixed value t_{mem} ($t_{cycle} = t_{mem} > (t_{div} + t_{latch})$).

3.5.2 Application to radar ESM systems

By estimating the system's performance for a range of feasible configurations, we wish to obtain appropriate values for: N, P, D, N/P, P/D, N/D, when $M = 16$, $N_{tot} \geq 250$, $b = 11$ bits, $\beta = 1$, and the input processing speed is $t_{tot} = 2\mu$s. The simulation results also offer a better understanding of the design trade-offs associated

Tableau 3.2: Delay and area measures used for the performance estimation

Module	Delay ($n$sec)	Area (mm^2)
Fixed-point divider	t_D=75.2	A_D=0.468
Neural processor (NP)	t_{NP}=4.93	A_{NP}=0.192
RAM for prototype vectors	t_{mem}=5.0	A_{membit}=0.000337
Local and global comparator	t_{com}=3.96	A_{com}=0.0372
Pipeline register latch	t_{latch}=1.23	$A_{latchbit}$=0.000463

with the implementation of the fuzzy ART system architecture.

Table 3.2 presents preliminary area and delay measures for the system modules used within the cost function. These measures were obtained through synthesis using Synopsis design tools, with Nortel's 0.8μm BiCMOS technology [70], for a library of cells developed by the Canadian Microelectronics Corporation (CMC). Note that these measures are highly dependent on the technology used and on the degree of optimization invested in each module. Any conclusions drawn on results from the area-times estimation model must consider these factors. A nonrestoring division array [30] was implemented to compute the divisions. With this divider, we estimated the area of a level of pipelining registers to be $A_{level} = [(2B-1)+(E+1)/2] \cdot A_{latchbit}$. The area of a bit A_{membit} was estimated using a BiCMOS 1-Port SRAM block provided by the CMC [36], $A_{latchbit}$ was obtained from an 11 bit shift register, and t_{latch} was obtained by circuit simulation of a rising edge D flip-flop.

Since the delays and areas shown do not account for the effects of interconnects, and the only available parameters are for typical process as opposed to worst case, the

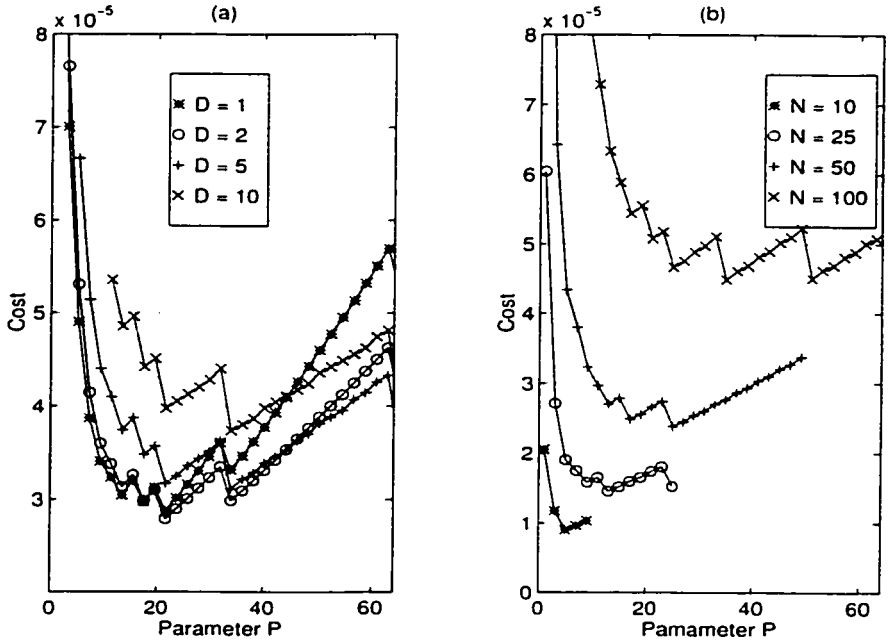


Figure 3.8: Cost ($A_{tot} \cdot t_{tot}$) simulation results: (a) Cost versus P for $N = 64$ (with different D values); (b) Cost versus P for $D = 4$ (with different N values).

final delay values used were all conservatively doubled when used for cost estimation. Areas were used directly but similar doubling should be expected. For example, during the simulations $t_{cycle} = 2 \times t_{mem} = 10ns$ is used. To achieve a minimum processing time and area (*i.e,*. a minimum value $E \cdot A_{level}$ in Eq. 3.4), we set $E = E_{limit}$ and $t_{cycle} = t_{mem} = (t_{div} + t_{latch})$. Then, memory access time t_{mem} is the system's processing speed bottleneck.

Figure 3.8 presents examples of the cost simulation results (from Eqs. 3.4 and 3.5) that can be used to deduce an appropriate set of system parameters for $C = 4$, $P \geq D$, $N \geq P$, and $E = E_{limit} = 20$ levels. Figure 3.8(a) shows the effect of varying P and D on cost values when $N = 64$. The results show that a smaller

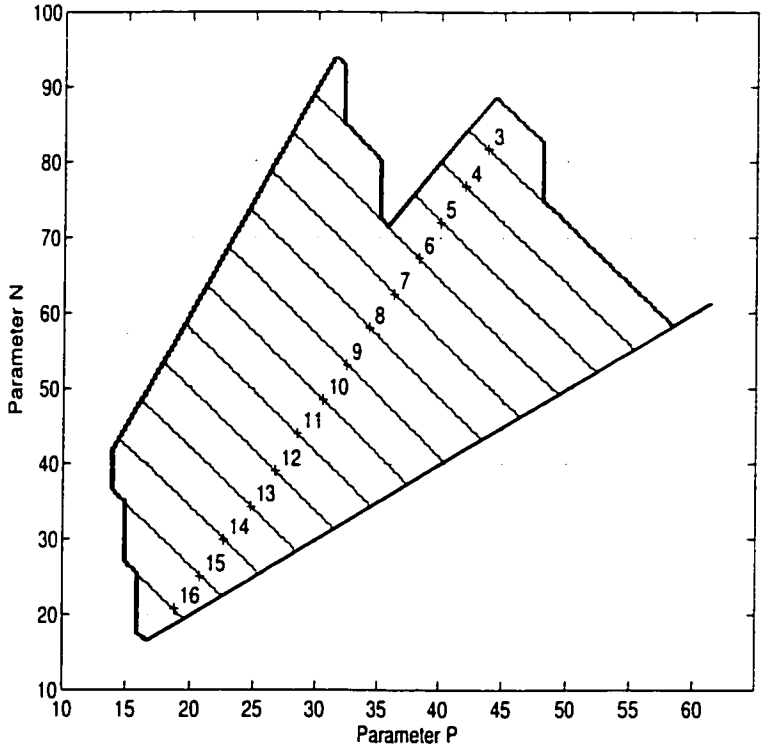


Figure 3.9: Contour plot of N versus P for chip configurations meeting the constraints $A_{tot}/C = 20\text{mm}^2$, $t_{tot} = 2\mu s$, and $t_{cycle} = 10ns$. The number of dividers is given by the diagonal lines.

number of dividers D does not always yield a lower cost. It was also observed that minimum cost values are obtained for lower P/D ratios as the value D grows. For instance, the minimum costs range from $P/D \simeq 20$ with $D = 1$, to $P/D \simeq 2$ with $D = 20$. This indicates that lower costs can be attained with smaller P values as D grows. Figure 3.8(b) shows the effects of varying N and P on the cost values, when $D = 4$. Small N values give lower costs, and optimal N/P ratios grow slowly with N. Specifically, the minimum costs range from $N/P \simeq 2$ with $N = 10$, to $N/P \simeq 2.5$ with $N = 100$. This result emphasizes the importance of selecting a relatively small ratio of neurons per NP, N/P.

Figure 3.9 shows the contour plot of N versus P obtained from Eqs. 3.4 and 3.5, assuming the processing rate constraints $t_{cycle} = 10\text{ns}$ and $t_{tot} = 2\mu\text{s}$, and the area constraint $A_{tot}/C = 20\text{mm}^2$. As a general rule, when t_{cycle} is reduced or t_{tot} grows, more neurons per NP and/or more NPs per divider still allow meeting area and delay requirements. The rather small area limit per chip was selected since module interconnections and pad frame are expected to double it. The region inside the contour plot of Figure 3.9 corresponds to values of N, P, and D that can meet these area-time constraints.

The diagonal lines shown correspond to the number of dividers D, ranging from 2 to 17. Figure 3.9 shows that at least 2 dividers and 14 neural processors are required to meet the speed constraint. As for the area constraint, no more than 17 dividers or 60 neural processors can fit in a chip. Since our system design goal is to maximize the number of neurons per chip, N, or minimize the number of chips ($C = N_{tot}/C$), the upper portions of the region corresponds to good parameter choices. The maximum number of neurons that can be put on one chip while meeting the speed and area requirements is 93. This is obtained by choosing 4 dividers and 31 neural processors. Chip control may be complicated with this setup since not all dividers would have the same number of neural processors, and not all neural processors would emulate the same number of neurons. If P is constrained to be a multiple of D, then 92 neurons can be emulated by using 4 dividers and 32 neural processors. Moreover, if N is constrained to be a multiple of P, than 88 neurons can be emulated by using

2 dividers and 44 neural processors. Both configurations lead to a 3 chip system capable of categorizing 500000 patterns per second. A proof-of-concept, digital VLSI implementation is currently under development at École Polytechnique de Montréal. The design has been described with VHDL, and is being targeted to BiCMOS 0.8 μm technology [35], using Cadence, and the Synopsys simulation and synthesis tool.

3.6 Conclusion

In this paper, the fuzzy ART neural network's algorithm has been reformulated to offer more efficient solutions for its digital VLSI hardware implementation. A dedicated system architecture that partitions this reformulated algorithm into several identical integrated circuits has then been proposed. This system architecture is suitable for high throughput clustering applications where a high degree of accuracy must be maintained. It is currently being considered for on-line clustering of intercepted radar pulses in ESM systems (*i.e.*, data reduction). An area-time estimation model has also been presented for this system architecture. This model may help designers of rapid clustering systems in the selection of an appropriate system configuration (or system parameters), given the constraints of their application. An example of system configuration choice is demonstrated for a radar signal clustering problem. The estimation results support the general approach, and the practicality of a digital VLSI implementation.

The VLSI system architecture presented is just one among many possible archi-

tectures for the fuzzy ART algorithm. For example, it would be possible to distribute bit-serial dividers in each NP, rather than having global dividers for a group of NPs. Another option would be to replace the division by another operation. It would also be useful to include a mechanism to collect statistics relative to the categories created (i.e. time elapsed since last choice, selection, update, etc.). This would permit rapid reinitialization of infrequently selected categories and reduction of the system's global processing requirements. The results of our current implementation, and future exploration should offer greater insight into the trade-offs involved in a fuzzy ART system's implementation, and uncover an even broader range of effective architectural solutions.

Acknowledgments

The authors would like to acknowledge the contribution of Eric Audet. This work was supported in part by the Defence Research Establishment Ottawa, the Canadian Microelectronics Corporation, le Fonds pour la Formation de Chercheurs et l'Aide à la Recherche, and the Natural Sciences and Engineering Research Council of Canada.

3.7 Synthèse et impact des résultats

Dans l'article précédent, l'algorithme du RNA fuzzy ART a été reformulé pour faciliter sa mise en oeuvre numérique. Une architecture de système VLSI dédiée a été proposée pour partitionner la fonctionalité de cet algorithme sur plusieurs ASIC. Ce système a été ciblé au triage métrique d'impulsions radars à débit élevé. Le modèle d'estimation AT pour cette architecture a permi d'isoler un ensemble de configurations qui peuvent accomoder plus de 250 catégories et traiter bien au-delà de 10^5 impulsions par seconde, tout en occupant une surface acceptable.

Il existe plusieurs variantes intéressantes de l'architecture de système VLSI. Pour une gestion intelligente des resources, l'architecture peut intégrer un mécanisme qui emmagasine des statistiques sur les catégories formées. Les catégories associées aux émetteurs non actifs peuvent alors être re-initialisées. D'autre part, puisque la division $T_j = |\mathbf{w}_j \wedge \mathbf{I}|/(|\mathbf{w}_j| + \alpha)$ est un élément critique de l'architecture, remplacer le diviseur par un multiplicateur (de $|\mathbf{w}_j \wedge \mathbf{I}|$ et $(|\mathbf{w}_j| + \alpha)^{-1}$) peut avoir un impact considérable sur la performance. Il est aussi possible de distribuer des diviseurs bit-sériels dans chaque processeur neuronique, plutôt qu'avoir un diviseur global lié à plusieurs processeurs. Finalement, pour augmenter la vitesse des traitements, chaque module élémentaire peut effectuer une phase d'apprentissage préemptive pendant qu'il attend le résultat du comparateur global. Les poids neuroniques peuvent ensuite être modifiés officiellement pour le module qui contient le vainqueur global.

Afin de démontrer le concept, l'architecture d'un module élémentaire a été conçue

et validée, principalement par Marc-André Cantin et Bruno St-Pierre [17] [18] [19]. Plusieurs autres étudiants du Groupe de Recherche en Microélectronique ont aussi contribués à la réalisation de ce ASIC. La version la plus récente de l'architecture a été décrite en VHDL, et la description a été ciblée vers la technologie CMOS 0.35 μm avec les outils de conception Synopsys et Cadence. Quelques douzaines d'ASIC ont été fabriqués à travers de la Société Canadienne de Microélectronique.

Chaque ASIC contient 74k transistors dans un surface d'environ 24 mm^2. Un patron d'entrée peut être traité à chaque 6 μsec avec une fréquence horloge de 50 MHz. Le circuit VLSI accepte des patrons d'entrée à $M = 16$ dimensions[2], et permet d'émuler jusqu'à $N = 32$ neurones de sortie, *i.e.*, categories. Toutefois, selon les besoins de l'application, le nombre total de neurones de sortie (N_{tot}) peut être augmenté en interconnectant plusieurs de ces ASIC en cascade. (Par exemple, un circuit imprimé qui contient ces ASIC peut s'interfacer avec un CPU hôte qui s'occupe du contrôle et de choisir le vainqueur global.)

Cette réalisation de l'architecture est composée de $P = 8$ processeur neuroniques, où chacun émule $N/P = 4$ neurones de sortie, de comparateurs, d'un diviseur à point fixe de 17 bits, et d'une mémoire pour les poids neuroniques (11 blocs de SRAM de 128mots × 8bits). Les poids neuronique w_{ji} et les éléments du patron d'entrée a_i qui sont manipulés par le circuit ont un format fixe. Ils sont représentés avec

[2]Cette version du circuit a été conçu pour traiter des impulsions radars qui ont la même représentation que celle définie au chapitre précédent ou dans [55], dans l'ensemble de données Radar.

une résolution de $b = 11$ bits. Les paramètres et les poids neuroniques du RNA fuzzy ART sont programmables. Finalement, pour effectuer des tests structurels, trois sous-circuits ont été insérés — (1) un "JTAG boundary scan" compatible avec la norme IEEE 1149.1, (2) un "scan chain" avec une couverture de panne d'environ 97%, et (3) un "memory scan" pour la SRAM.

L'algorithme reformulé a aussi été réalisé sur deux différentes plate-formes — la carte XCIM de MiroTech et le TMS320C40 de Texas Instruments [18] [110]. Ces réalisations ont permi d'explorer les alternatives en termes du temps de traitement, du coût de conception et de fabrication, ainsi que de l'espace mémoire pour accomoder les neurones, etc., dans une mise en oeuvre du RNA fuzzy ART.

Chapitre 4

La catégorisation en-ligne par le ré-ordonnancement de patrons ambigus

À haut niveau, un système pour faire l'apprentissage en-ligne de catégories comprend un module de catégorisation qui utilise une technique de traitement. Cette technique gère la manière dont les patrons de la séquence d'entrée sont appris par le module de catégorisation.

Dans ce chapitre, on suppose que le module de catégorisation est un RNA auto-organisateur tel que le fuzzy ART. Celui-ci appartient à l'ensemble des réseaux à apprentissage compétitif dont le taux d'apprentissage demeure constant. Si le système de catégorisation consiste tout simplement d'un RNA de ce groupe (c'est le cas par

défaut), on peut dire qu'il utilise le *traitement séquentiel* pour faire la catégorisation en-ligne d'une séquence de patrons d'entrée. Lorsqu'un patron d'entrée se présente au système, il est appris immédiatement par le RNA. Pour ce faire, le patron est comparé au prototype de chaque catégorie. La catégorie dont le prototype est le plus semblable (selon la fonction d'activation des neurones du RNA) se voit assigné le patron. Le prototype de cette catégorie est immédiatement adapté pour apprendre les caractéristiques de ce patron. En catégorisant une séquence de patrons, les prototypes apprennent à se spécialiser pour différentes régions de l'espace des entrées. On peut dire qu'ils définissent implicitement les bornes de décision entre catégories.

Grâce au traitement séquentiel et au taux d'apprentissage constant du RNA fuzzy ART, le système réussit à conserver sa plasticité face à l'environment, une propriété intéressante pour le triage métrique en MSE radar. Cependant, la comparaison présentée au chapitre 2 indique que la qualité et la variabilité (selon l'ordre de présentation des données) des catégorisations du RNA fuzzy ART laissent à désirer. Ces limitations sont en partie liées aux *décisions ambiguës* et au traitement séquentiel des patrons d'entrée qui tombent près de la borne de décision entre deux ou plusieurs categories. L'apprentissage immédiat et irréversible a comme effet de perpétuer l'impact de cette décision, car seulement une des catégories sera assignée à un tel patron. La qualité des catégorisations obtenues avec les RNA comme fuzzy ART peuvent donc varier considérablement selon l'ordre de présentation des patrons.

L'utilisation du *traitement par lot* permet généralement d'améliorer la qualité

des catégorisations en dépit du temps de réponse. Dans ce cas, les patrons de la séquence d'entrée sont accumulés dans une pile de longueur fixe. Lorsque la pile est pleine, le RNA apprend les patrons correspondants, jusqu'à convergence, avec plusieurs présentations du lot de données. L'apprentissage par le RNA de lots successifs se fait de façon incrémental. Le fait de laisser converger un lot de données sur plusieurs passes de traitement séquentiel permet de minimiser (de façon locale ou globale) la fonction de coût du RNA[1]. Ce traitement permet aussi au RNA de compenser les effets dûs à l'ambiguïté. Le traitement par lot occasionne toutefois un temps de traitement par patron qui est plus important que celui du traitement séquentiel. De plus, ce temps est variable selon les données du lot.

Puisqu'on cible une application à débit élevé, le temps de réponse associé à une technique de traitement est critique. Ce chapitre présente la troisième contribution du premier volet de cette thèse. Il s'agit de la proposition d'une nouvelle approche pour faire la catégorisation en ligne d'une séquence de patrons. Le *traitement par réordonnancement* est une alternative au traitement par lots pour l'amélioration de la qualité des catégorisations, mais qui permet aussi de contrôler le temps de réponse du système. Un RNA d'un système qui exploite ce traitement est modifié afin de pouvoir détecter un patron d'entrée qui mène à une décision ambiguë. Lorsqu'un tel patron est détecté, il est emmagasiné dans une file, et son apprentissage est retardé pendant

[1]Il est à noter que le traitement par lot a été utilisé pour obtenir les résultats de la comparaison au chapitre 2. Dans ce cas, la longueur de la pile est de 800 patrons, soit le nombre de patrons de l'ensemble de données Radar.

un délai fixe. Après ce délai, le patron est appris pour de bon par le RNA. Avec le traitement par ré-ordonnancement, l'ambiguïté est alors utilisée comme critère pour modifier l'ordre d'apprentissage des patrons par le RNA.

Pour démontrer la validité du concept, la qualité obtenue et la latence requise pour effectuer des catégorisations ont été comparées pour un système de catégorisation qui utilise le traitement séquentiel, par lot et par ré-ordonnancement. Des simulations ont été effectuées avec l'ensemble de données Radar (présenté au chapitre 2), ainsi que deux RNA auto-organisateurs de type apprentissage compétitif — le ART2A-E [48] et le fuzzy ART [21]. La mesure de similarité *Rand Adjusted* a été employée pour comparer la qualité des résultats. Afin de comparer le temps de réponse, des bornes inférieures ont été dérivées sur la latence requise pour la catégorisation en ligne avec le traitement séquentiel, par lot et par ré-ordonnancement. Une mise en oeuvre typique des techniques de traitement par lot et par ré-ordonnancement ont été développées, et leur latence a été comparée aux bornes inférieures. Enfin, la théorie sur l'option de rejet a permi de dériver deux modèles pratiques pour faire la détection des cas ambigus.

L'article suivant contient plus de détails sur le traitement par ré-ordonnancement:

GRANGER, E., SAVARIA, Y., et LAVOIE, P.,

"A pattern reordering approach based on ambiguity detection for on-line category learning," *IEEE Trans. Pattern Analysis and Machine Intelligence*, soumis pour publication, l'article a été resoumis pour une 2e phase de revision

(novembre 2001).

Une copie de cet article est reproduite ici. Enfin, la dernière section du chapitre aborde une discussion sur l'impact des résultats de cette contribution.

Il existe une version longue de cet article sous forme d'un rapport technique [61]. Ce rapport fournit en annexe A contient plus de details sur les dérivations mathématiques permettant d'appliquer l'option de rejet à la détection des cas ambigus. Il contient aussi des resultats de simulations plus extensifs obtenus avec (1) des données distribuées de façon Gaussienne et (2) l'ensemble de données Iris [46].

A pattern reordering approach based on ambiguity detection for on-line category learning

Eric Granger[1,2], Yvon Savaria[2] and Pierre Lavoie[3]

[1] Integrated Systems Group, Mitel Networks, 350 Legget Dr., Ottawa, Ontario, K2K 2W7, Canada.

[2] Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C. P. 6079, Station "centre-ville," Montreal, Quebec, H3C 3A7, Canada.

[3] Defence Research Establishment Ottawa, Department of National Defence, 3701 Carling Ave., Ottawa, Ontario, K1A 0Z4, Canada.

Abstract

Pattern reordering is proposed as an alternative to sequential and batch processing for on-line category learning. Upon detecting that the categorization of a new input pattern is ambiguous, the input is postponed for a predefined time, after which it is reexamined and categorized for good. This approach is shown to improve the categorization performance over purely sequential processing, while yielding a shorter

[1]This research was carried out while E. Granger was a scientist at the Defence Research Establishment Ottawa. His corresponding address is: 98 Sweetland Ave., Ottawa, Ontario, K1N 7T8, Canada, email: eric.granger@rogers.com, phone: 1-613-230-8418.

input response time, or latency, than batch processing. In order to examine the response time of processing schemes, the latency of a typical implementation is derived, and compared to lower bounds. Gaussian and softmax models are derived from reject option theory, and are considered for detecting ambiguity and triggering pattern postponement. The average latency and Rand Adjusted clustering score of reordered, sequential and batch processing are compared through computer simulation using two unsupervised competitive learning neural networks and a radar pulse data set.

4.1 Introduction

A number of pattern recognition applications involve high throughput category learning from continuous streams of input patterns. In pattern recognition literature, partitional clustering techniques [3] [43], such as on-line versions of the k-means [97] and leader [71] algorithms, are proposed for on-line category learning. Other algorithms include on-line versions of adaptive vector quantization (AVQ) [51] [64] (*e.g.*, generalized Lloyd [95] and Linde-Buzo-Gray (LBG) [92] algorithms) in communication theory, and unsupervised competitive learning (UCL) [65] [66] [86] (*e.g.*, standard UCL networks [1] [119] and Adaptive Resonance Theory (ART) networks [20]) in neural network theory. These algorithms learn input patterns autonomously on the fly, some without prior knowledge of the number and characteristics of the categories.

A common trait of these algorithms is that they process inputs sequentially. When an input pattern is presented, it is compared to the prototype of every category. The

category with the best matching prototype is assigned to the input, and the prototype of this category adapts to novel characteristics of the input. Over a succession of inputs, category prototypes become tuned to different parts of the input space, and implicitly define inter-category decision boundaries. Since a final decision is taken upon presentation of every input, only information from prior input patterns is available. In addition, the fact that prototypes are stored rather than prior input patterns implies that learning — the consequence of each decision — is irreversible. Information is therefore lost in the process.

Limitations of such *sequential processing* are obvious, particularly if the structure of input data clusters tends to be scattered and/or overlapping. For instance, if upon its presentation, an input pattern lies close to a decision boundary separating two or more categories, the clusterer is forced to make a final decision, despite the ambiguity. Irreversible learning for such decisions yields category structures and decision boundaries that vary significantly according to the pattern presentation order.

The quality of results is generally improved with *batch processing.* This consists in accumulating one batch of patterns from the input stream, while the clusterer converges on a previously-accumulated batch of patterns. Batch processing can diminish or eliminate sensitivity to the input presentation order by allowing the clusterer to recover from early miss-assignments. However, it entails some delay for the accumulation of patterns into batches, and may require data buffering if the computational effort is variable.

In this paper, an alternative called *reordered processing* is proposed for on-line category learning. As a clusterer assigns categories to input patterns, it is granted the ability to postpone, or delay, category assignment and learning when it detects an ambiguity. Delayed patterns are queued, and their categorization is deferred for some fixed time. The overall effect is a modification to the order in which inputs are categorized. Aside from allowing to control the maximum response time, pattern postponement offers the opportunity to circumvent learning for ambiguous decisions.

The rest of this paper is organized as follows. In order to compare the response time of clustering systems, lower bounds on the latency required for on-line category learning are developed for sequential, batch and reordered processing in Section 2. Practical issues are also discussed, and architectures are proposed. In Section 3, elements of reject option theory are briefly reviewed and developed for the detection of ambiguous decisions, as required by reordered processing. Finally, the experimental methodology, and proof-of-concept computer simulations are presented and discussed in Section 4. It is shown that the proportion of patterns that are declared ambiguous is a good indicator of poor clustering quality; and that a modification of the order in which input patterns are processed can indeed enhance clustering quality.

4.2 Latency in on-line category learning

On-line category learning of a continuous stream of patterns is performed by a clusterer (*e.g.*, an on-line k-means algorithm) which is exploited through one of several

data processing schemes. Let input patterns to be categorized arrive one at a time from some source that provides them at a rate that need not be constant. The *latency* L of the category learning is defined as the number of input patterns necessary before a final decision (category assignment) $\mathbf{y}(\mathbf{a})$ can be made for input $\mathbf{a}$.

The basic approach to learning a continuous data stream is through *sequential processing*. Upon observation of each input pattern $\mathbf{a}$, a category is assigned to $\mathbf{a}$ without waiting for subsequent patterns. *Batch processing* consists in categorizing the input patterns $\{\mathbf{a}\}$ in fixed-size batches of k patterns. Once k successive patterns have been collected and stored, the following patterns are buffered while the k patterns are being learned. Batches of k patterns are learned iteratively, over several epochs, until convergence is attained, that is, until the prototypes remain constant for two epochs (complete presentations of data in the batch). Categories are assigned once convergence has been detected, *i.e.*, after the last one of these epochs. *Reordered processing* consists in postponing the learning of input patterns for which category assignment is deemed ambiguous. Each postponed pattern is queued, and following a fixed latency of d patterns is categorized once and for all.

Assuming an infinitely fast clusterer, the minimum latency $L_{\min}$ for all three schemes must satisfy:

$$L_{\min} \geq 0 \ . \tag{4.1}$$

This lower bound corresponds to $\mathbf{a}$ being the last pattern in a batch with batch processing, or $\mathbf{a}$ not being postponed with reordered processing. The maximum latency

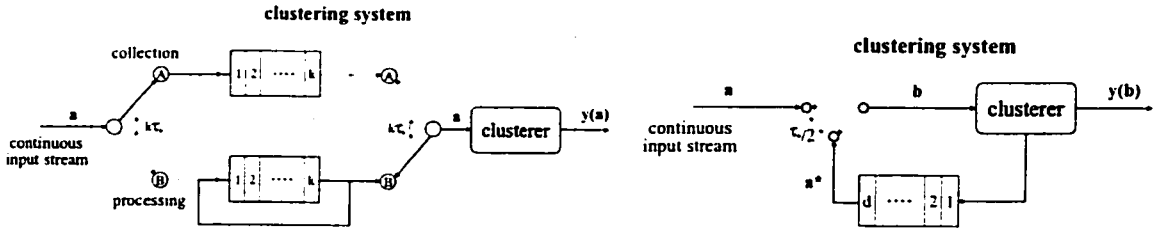
$L_{\max}$ must satisfy:

$$L_{\max} \geq \begin{cases} 0\ ; & \text{sequential processing,} \\ k-1\ ; & \text{batch processing,} \\ d\ ; & \text{reordered processing,} \end{cases} \tag{4.2}$$

where k is the number of patterns per batch, and d is the user-defined queue size. This bound corresponds to **a** being the first pattern in a batch with batch processing, or **a** being postponed with reordered processing. Lastly, it is straightforward to show that the average latency $\overline{L}$ must satisfy:

$$\overline{L} \geq \begin{cases} 0\ ; & \text{sequential processing,} \\ \frac{1}{k}\sum_{i=1}^{k}(k-i)\ ; & \text{batch processing,} \\ p_r(\mathbf{a}) \cdot d\ ; & \text{reordered processing,} \end{cases} \tag{4.3}$$

where $p_r(\mathbf{a})$ is the pattern rejection, or postponement rate. This rate depends on the data and the rejection criterion. Reordered processing constitutes a compromise between sequential processing and batch processing. Indeed if $d = (k-1)$ and $p_r(\mathbf{a}) = 50\%$, then the lower bounds on $L_{\max}$ and $\overline{L}$ are equal for batch and reordered processing. If, on the other hand, $d = 0$, then reordered processing reduces to sequential processing.



(a) Batch processing architecture. (b) Reordered processing architecture.

Figure 4.1: Clustering system architectures.

To characterize the latency of practical clustering systems, it is convenient to assume that inputs arrive at a regular interval τ_a. With batch processing, the number of batch epochs, δ, required to attain the state of convergence varies from one batch to the next according to the queue length k, and to the structure of the data in the batch. The computational effort is therefore variable and somewhat unpredictable. Detection of convergence requires at least one whole batch epoch of overhead and thus $\delta \geq 2$. To circumvent this delay, one can bound the number of batch epochs to a constant value across all batches, $\delta \geq 2$. A final decision $\mathbf{y}(\mathbf{a})$ can then be produced immediately after processing of $\mathbf{a}$ as part of the last epoch.

One possible batch processing architecture is shown in Figure 4.1(a). It consists of two identical fixed-length queues of k registers that operate concurrently, and a clusterer that can process each pattern within a fixed time τ_c. Each queue alternates between a collection (A) and a processing (B) mode. While one of the queues temporarily buffers k incoming patterns from the input stream (A), the other queue stores a batch of k patterns being learned by the clusterer (B). The processing rate must

satisfy $\tau_c \leq \tau_a/\delta$, which imposes a processing rate constraint on the clusterer that is proportional to δ, and prevents overflows of the queue operating in collection mode. Once a queue is switched to the collection mode (A), a pattern **a** from the input stream is stored every τ_a seconds inside its registers. After $k\tau_a$ seconds, the registers are full, and the queue is switched into processing mode (B). In this mode, patterns are shifted right through the clusterer every τ_c seconds. A feedback loop allows repeated presentations. After δ epochs, the registers are reset prior to switching back to collection mode. The latency of the batch processing architecture of Figure 4.1(a) is the sum of the delays incurred in both modes. In the best case, when **a** is the last of a batch,

$$L_{\min} = k \quad , \tag{4.4}$$

and in the worst case, when **a** is the first of a batch,

$$L_{\max} = (k-1) + \left\lceil \frac{(\delta-1)k+1}{\delta} \right\rceil \quad , \tag{4.5}$$

The average latency of this architecture is:

$$\overline{L} = \frac{1}{k}\sum_{i=1}^{k} \left\{ (k-i) + \left\lceil \frac{(\delta-1)k+i}{\delta} \right\rceil \right\} \quad , \tag{4.6}$$

It is easily verified that (4.4), (4.5) and (4.6) satisfy, but do not meet, the lower bounds (4.1), (4.2) and (4.3).

With reordered processing, a pattern having been postponed by d patterns is given

priority over a pattern from the input stream. This ensures a fixed worst-case response time of d patterns, but leads to the accumulation of up to d incoming patterns. Also, a clusterer embedded within this system requires additional functionality to detect ambiguity.

One possible reordered processing architecture is shown in Figure 4.1(b). It consists of a fixed-length queue composed of d registers, and a clusterer capable of detecting ambiguous category assignments. If an input pattern $\mathbf{a}$ is deemed ambiguous, it is diverted towards the queue and labeled $\mathbf{a}^*$. Shifting this pattern through the queue is equivalent to a fixed delay that changes the presentation order. Each pattern $\mathbf{b}$ that is processed by the clusterer is either a new input pattern, $\mathbf{b} = \mathbf{a}$, or a previously-rejected and therefore delayed pattern, $\mathbf{b} = \mathbf{a}^*$. A simple way to schedule processing is to assume that the clusterer's processing time is partitioned into interleaved time slices: odd slices reserved for input patterns $\{\mathbf{a}\}$, and even slices reserved for previously-rejected patterns $\{\mathbf{a}^*\}$. To alternate between the two sources, the augmented clusterer must be able to process a pattern $\mathbf{b}$ within a fixed time of τ_c that satisfies $\tau_c \leq \tau_a/2$. This speed constraint is similar to that of batch processing with $\delta = 2$. With this architecture, the lower bounds (4.1), (4.2) and (4.3) are met.

4.3 Ambiguity in competitive learning assignments

With reordered processing, ambiguity is employed as a criterion for postponing pattern categorization. In statistical pattern recognition, the *reject option* [33] [34] [49]

provides a framework for detecting ambiguous classifications. The Bayes decision procedure assigns one-of-N possible classes to input $\mathbf{a}$ using the maximum *a posteriori* probability decision rule, $J = \arg\max\{p(j \mid \mathbf{a}) : j = 1, 2, ..., N\}$, where $0 \leq p(j \mid \mathbf{a}) \leq 1$ and $\sum_{j=1}^{N} p(j \mid \mathbf{a}) = 1$. The *a posteriori* probability $p(j \mid \mathbf{a})$ that class j generated input $\mathbf{a}$ is computed according to the Bayes theorem [45] [50].

The degree of ambiguity regarding this decision rule can be measured in terms of the conditional error given input $\mathbf{a}$, $r_J(\mathbf{a}) = 1 - \max\{p(j \mid \mathbf{a}) : j = 1, 2, ..., N\}$ [33] [34] [44] [49]. Assignment of class J to input $\mathbf{a}$ is defined as *ambiguous* if $r_J(\mathbf{a})$ is greater than or equal to a *rejection threshold* γ, $\gamma \in (0, \frac{N-1}{N}]$. This criterion can be rewritten:

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{P_J \; p(\mathbf{a} \mid J)}{\sum_{j=1, j \neq J}^{N} P_j \; p(\mathbf{a} \mid j)} \; , \tag{4.7}$$

where P_j is the *a priori* probability of class j, with $0 \leq P_j \leq 1$ and $\sum_{j=1}^{N} P_j = 1$, and $p(\mathbf{a} \mid j)$ is the conditional probability density function (p.d.f.) of the input $\mathbf{a}$ given class j. Eq. (4.7) defines a region in the input space where class assignments are considered to be ambiguous. The size of this region grows as γ is decreased. For a given γ, the size of the region also depends on the shape of $r_J(\mathbf{a})$ at the decision boundary of class J, and hence on the class distribution in the input space.

In practice, the probabilities required to implement the reject option are often unknown. For on-line clustering applications, these probabilities must be estimated from the clusterer's response to input patterns. A clusterer may be subjected to different environments, with more or less prior information on the underlying data

structure. Eq. (4.7) is now developed for two possible environments.

In the *Gaussian rejection model*, the data are assumed to be generated by sources with the same Gaussian noise and with equal *a priori* probabilities P_j, and all variables are assumed to be statistically independent and to have equal variance σ^2. Then, data clusters are modeled explicitly as hyper-spherical normal distributions centered at mean vectors $\{\mu_j\}$. For this model, Eq. (4.7) becomes

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{\exp\left\{-\frac{||\mathbf{a}-\mu_J||^2}{2\sigma^2}\right\}}{\sum_{j=1, j\neq J}^{N} \exp\left\{-\frac{||\mathbf{a}-\mu_j||^2}{2\sigma^2}\right\}} \; . \tag{4.8}$$

When implemented as part of a clusterer, μ_j is equal to the prototype vector of category j. The variance σ^2 is required, either from prior knowledge or from on-line estimation. The Euclidean distance $||\mathbf{a} - \mu_j||$ is a core component of the prototype matching function commonly used by clusterers when prototypes represent mean patterns.

In the *softmax rejection model*, no prior assumption is made regarding the underlying data structure, and data clusters are modeled implicitly. During on-line category learning, the prototype matching function provides the response strength for each category with respect to $\mathbf{a}$. The match strength ϕ_j between input and prototype can be interpreted as an estimate of the a posteriori probability $p(j \mid \mathbf{a})$. To ensure that the ϕ_j values are valid probabilities (*i.e.*, sum up to 1 and range from 0 to 1), the *softmax activation function* [15], $y_j = \exp\{\phi_j\} / \sum_{k=1}^{N} \exp\{\phi_k\}$, from neural network literature can be applied. Assuming that y_j can be used as an estimate of $p(j \mid \mathbf{a})$,

the rejection rule $r_J(\mathbf{a}) \cong (1 - y_J) \geq \gamma$ can then be expressed in a form similar to that of Eq. (4.7):

$$\left(\frac{1-\gamma}{\gamma}\right) \geq \frac{\exp\{\phi_J\}}{\sum_{j=1, j\neq J}^{N} \exp\{\phi_j\}} \,. \tag{4.9}$$

The reader is referred to [59] for details of the mathematical derivation of Eqs. (4.7) - (4.9).

4.4 Simulation results and discussion

4.4.1 Experimental methodology

Unsupervised competitive learning (UCL) [65] [66] [86] [119] neural networks were used as clusterers for computer simulation of sequential, batch, and reordered processing. In order to compare the performance of the clustering systems, simulations were repeated over several independent trials. Prior to each trial, input patterns were shuffled into a random presentation order. The patterns were then learned by the clustering system under test. After every trial, clustering quality was measured.

The UCL neural networks selected for computer simulations are ART2A-E [48] and fuzzy ART [21]. ART2A-E is well suited for Gaussian type environments, where clusters are explicitly modeled as mean vectors. Fuzzy ART is appropriate for environments where clusters are modeled implicitly. Programs emulating these neural networks were written in the Matlab language for all three data processing schemes. For reordered processing, each network was granted the capacity to detect ambiguity.

The Gaussian rejection model (Eq. (4.8)) was used in the ART2A-E network, whereas the softmax rejection model (Eq. (4.9)) was used in the fuzzy ART network. In the second case, $\phi_j = T_j$ [21], for $j = 1, 2, ..., N$, such that the vigilance test is passed, was substituted in Eq. (4.9).

The data set employed for computer simulations was collected in the field by the Defence Research Establishment Ottawa. It consists of radar pulses from 12 shipborne navigation radars. Fifty pulses were collected from each radar, with the exception of radars #7 (100 pulses) and #8 (200 pulses). The pulses were preprocessed to yield 800 patterns with 16 real-valued features. Data clusters in this set set cannot be described by the same statistics — some are not Gaussian distributed, and they sometimes overlap one another [55]. This Radar data set is representative of a continuous pulse stream intercepted by a radar electronic support measures (ESM) system. Within these systems, automatic sorting of pulses according to emitter is very challenging due to the density and complexity of signals encountered in modern environments [40].

The Rand Adjusted similarity measure [76] was selected to assess clustering quality. This type of measure is known in pattern recognition literature as an external criterion index, and is used for evaluating the capacity to recover the true cluster structure [3] [43] [76]. A partition of n patterns into K groups defines a *clustering*. This can be represented as a set $A = \{a_1, a_2, ..., a_n\}$, where $a_h \in \{1, 2, ..., K\}$ is the category label assigned to pattern h. In our context, the correct classification results are known for the data sets used, and their patterns are all accompanied by cate-

gory labels. These labels provide a reference clustering, R, with which a clustering produced by computer simulation, A, may be compared. The Rand Adjusted measure now represents a *score*, $S_{RA}(A, R)$, that describes the quality of the clustering produced by the network. $S_{RA}(A, R)$ ranges from 0 to 1, where 0 denotes maximum dissimilarity (worst), and 1 denotes equivalence (best) [55].

4.4.2 Correlation of ambiguity with clustering quality

The clustering quality achieved can be examined as a function of the degree of ambiguity observed. Patterns from the Radar data set were categorized by the ART2A-E and fuzzy ART networks using reordered processing. Network parameters were fixed to values that produce a high Rand Adjusted score $S_{RA}(A, R)$ using fast learning and sequential processing. For a same randomly-selected presentation order, the rejection threshold γ was varied from trial to trial. Rejected patterns were counted, yet processed without postponement. After each trial, the score $S_{RA}(A, R)$, and the empirical rejection rate $\hat{p}_r$ were stored. The empirical rejection rate $\hat{p}_r$ is the ratio of the number of rejected patterns to the total number of patterns (the size of the data set). The linear correlation between score and respective rate was computed from a series of 1000 independent presentation orders, with γ ranging from 0 to 1.

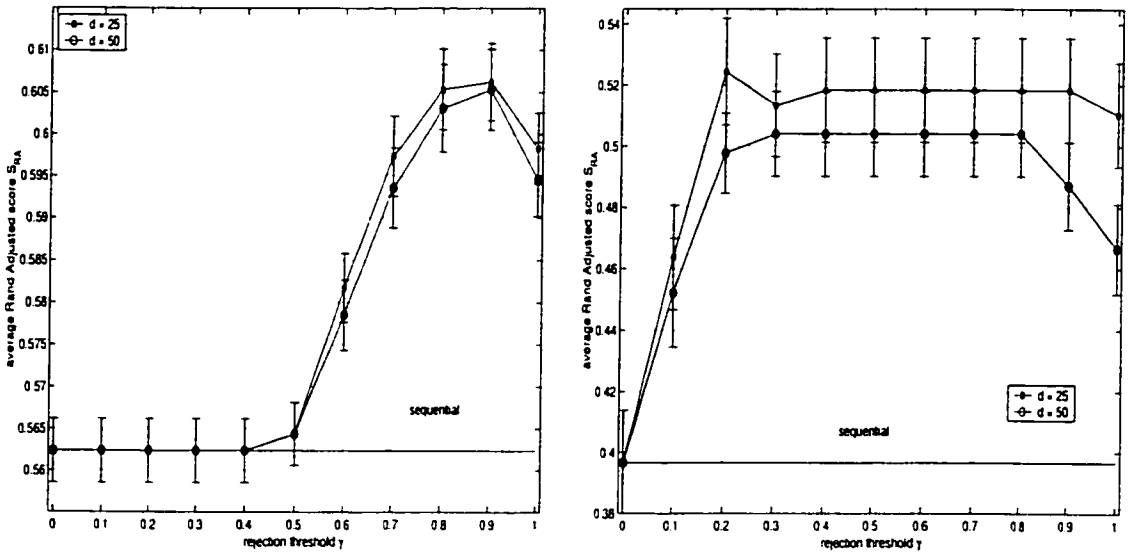
Results show that both clustering systems display a significant negative correlation between $\hat{p}_r$ and $S_{RA}(A, R)$, albeit in different ranges of values for the threshold γ. For a given value of γ, an increase in ambiguous category assignments corresponds to a

decline of the clustering score. For example, when using fuzzy ART and softmax rejection, the peak negative correlation coefficient of about -0.65 is obtained when $\gamma = 0.95$, which corresponds to $\hat{p}_r \cong 14\%$.

4.4.3 Clustering quality obtained using reordered processing

Having shown that the Gaussian and softmax rejection models detect ambiguous category assignments, and that these assignments lead to poor clustering performance, the effect of delaying ambiguous assignments is now examined. ART2A-E and fuzzy ART parameters were fixed to provide high Rand Adjusted scores $S_{RA}(R, A)$ with fast learning and sequential processing. Rejected patterns were delayed by d patterns. Two different delay values, $d = 25$ and 50, were tried. After each trial, $S_{RA}(A, R)$ was stored. Average $S_{RA}(A, R)$ values were obtained from 20 independent presentation orders, with γ ranging from 0 to 1.

Figure 4.2 shows the average $S_{RA}(A, R)$ as a function of γ. In both cases, reordered processing increases the clustering score over sequential processing alone. For example, if patterns are presented to the clustering system with fuzzy ART and $d = 25$, then reordered processing improves the score $S_{RA}(A, R)$ by about 30% over a wide range of γ values. In the example given above, scores of about $S_{RA}(A, R) = 0.52$ are obtained for $\gamma \in [0.2, 0.9]$, corresponding to $\hat{p}_r \cong 45\%$. For $0 \leq \gamma < 0.2$, the performance approaches that of sequential processing since $\hat{p}_r \rightarrow 0\%$. For γ values close to 1, the performance also declines because of the large number of rejections.



(a) ART2A-E on Radar data.

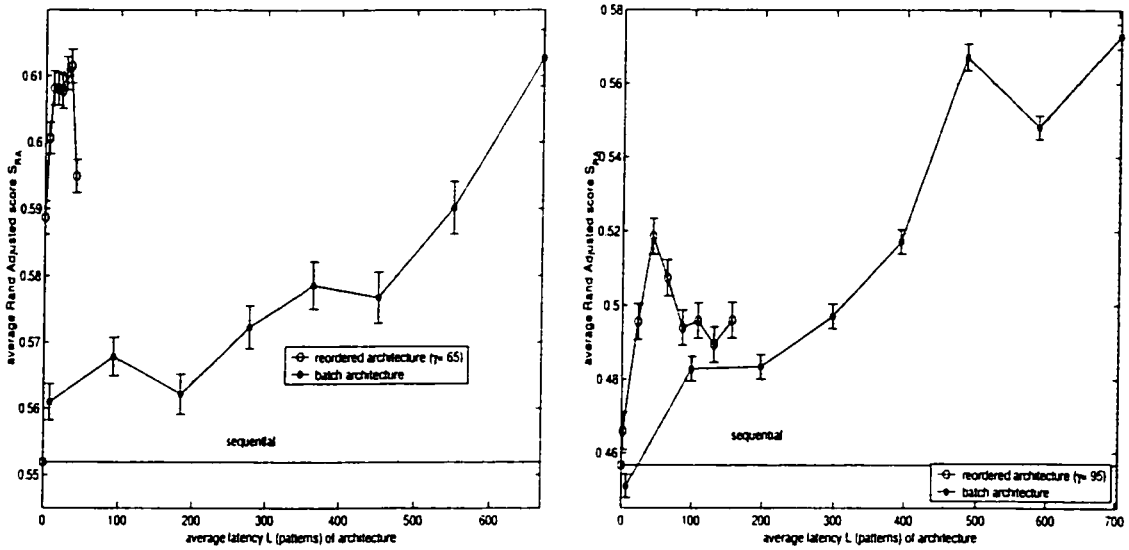
(b) Fuzzy ART on Radar data.

Figure 4.2: Average Rand Adjusted scores $S_{RA}(A, R)$ versus rejection threshold γ for simulations with the Radar data set. Error bars show standard error.

Fuzzy ART using softmax rejection shows strong negative correlation between $\hat{p}_r$ and $S_{RA}(A, R)$ across a wide range of γ values, and appears to benefit from reordered processing more than ART2A-E.

4.4.4 Clustering quality and latency

Sequential, batch and reordered processing are now compared in terms of clustering quality and latency. For a same randomly-selected presentation order, the queue lengths k and d were varied between 5 patterns and half the number of patterns in the data set, over successive trials. This range guarantees a minimum of two batches of k patterns for batch processing. With the batch architecture, the neural networks were



(a) ART2A-E on Radar data.

(b) Fuzzy ART on Radar data.

Figure 4.3: Average Rand Adjusted scores $S_{RA}(A, R)$ versus average latency $\overline{L}$ of the batch and reordered processing architectures for simulations with the Radar data set. Error bars show the standard error. The standard error for values of $\hat{p}_r$ always ranges from 0% to 2%. The batch architecture requires between 2 and 5 epochs for convergence on each batch of patterns.

left to converge for each batch of k patterns, until prototype weights were identical for two consecutive epochs. Network parameters were fixed *a priori* to provide high Rand Adjusted scores $S_{RA}(R, A)$ with batch processing when k is one half the data set size. With the reordered architecture, the network parameters were set *a priori* to provide high scores $S_{RA}(R, A)$ with sequential processing. The value of γ was set equal to 0.65 for ART2A-E and 0.95 for Fuzzy ART. After each trial, the score $S_{RA}(R, A)$ and the average latency $\overline{L}$ of the processing architectures were stored. The empirical rejection rate, $\hat{p}_r$, was used as a fixed estimate of the variable rejection rate $p_r(\mathbf{a})$, and was substituted into Eq. (4.3). Simulations were repeated 20 times for

every k and d value in order to yield representative results.

The average scores $S_{RA}(A, R)$ as a function of the average latency $\overline{L}$ for batch and reordered processing are shown in Figure 4.3. Both achieve significantly higher scores than sequential processing. However, reordered processing requires a much lower average latency than batch processing. For instance, using fuzzy ART and reordered processing with $\gamma = 0.95$ yields a score of $S_{RA}(A, R) \cong 0.52$ for $\hat{p}_r \cong 32\%$, $d = 118$ and an average latency of $\overline{L} \cong 40$ patterns. By comparison, batch processing yields a comparable level of performance for $k = 231$ and an average latency of $\overline{L} \cong 400$ patterns, ten times that of reordered processing. Results also reveal that the peak performance obtained with reordered processing often occurs for relatively small values of d, indicating that it is suitable for high speed applications. Beyond this peak, increasing d does not necessarily yield a higher score $S_{RA}(A, R)$. This may be due to our emulation of an infinite data stream with a fixed-size data set. Indeed, if an input a among the last d patterns of a data set is rejected, then it is postponed to the end of the data set, that is, for less than d patterns. The reader is referred to [59] for simulation results obtained on Gaussian distributed data and on the standard Iris data set.

4.5 Conclusions

A clustering system applied to on-line category learning would traditionally consist of a clusterer that processes input patterns sequentially. Despite the fast response

time obtained when using this data processing scheme, the quality and consistency of the results remain an issue. By contrast, batch processing yields higher clustering quality, but incurs longer delays. In this paper, an alternate approach has been proposed. It consists in postponing for a fixed time the learning of patterns for which category assignments are ambiguous. The reject option from statistical pattern recognition literature has been applied to the detection of ambiguous category assignments. Reordered processing alters the original pattern presentation order, and therefore introduces latency in the system. This latency has been defined and used to compare sequential, batch and reordered processing.

Computer simulations performed by presenting patterns from a radar pulse data set to ART2A-E and fuzzy ART neural networks with sequential, batch and reordered processing show that (1) the number of category assignments detected as ambiguous is correlated with poor clustering quality; (2) reordered processing improves clustering quality over sequential processing; and (3) it offers an attractive alternative to batch processing for trading off clustering quality versus response time.

Acknowledgements

This research was supported in part by the Defence Research Establishment Ottawa, the Fonds pour la Formation de Chercheurs et l'Aide à la Recherche du Québec, and the Natural Sciences and Engineering Research Council of Canada.

4.6 Synthèse et impact des résultats

En MSE radar, il est important de détecter les menaces aussi rapidement que possible. Le temps de réponse du triage métrique est alors un élément aussi critique que la qualité des catégorisations. Un système de catégorisation qui s'applique au triage métrique à débit élevé contient généralement d'un module de catégorisation en ligne qui utilise une technique de traitement appropriée. Dans l'article précédent, le traitement par ré-ordonnancement a été proposé pour améliorer la qualité des catégorisations obtenues avec le traitement séquentiel (technique par défaut), tout en permettant de contrôler le temps maximum de traitement d'un patron. Il s'agit de retarder l'apprentissage d'un patron pour un temps fixe quand on juge qu'une décision est ambiguë. C'est le cas lorsque le module de catégorisation ne détient pas assez d'informations pour assigner une seule catégorie à un patron. Le traitement par ré-ordonnancement s'applique à une grande famille d'algorithmes qui effectuent la catégorisation en ligne de séquences de patrons (*e.g.*, version en ligne de k-means).

Les résultats de simulations obtenus avec l'ensemble de données Radars et deux RNA qui utilisent le traitement sequentiel, par lot et par ré-ordonnancement ont permis de tirer les conclusions suivantes. Premièrement, il existe une corrélation négative forte entre le degré d'amiguïté qui est détecté (mesuré avec le taux de rejet empirique, $\hat{p}_r$), et la qualité des catégorisations (mesuré avec la métrique *Rand Adjusted*, $S_{RA}(A, R)$). Le nombre de patrons d'entré qui mène à une décision ambiguë est alors indicatif de la dégradation des résultats. Deuxièmement, un RNA qui utilise le

traitement par ré-ordonnancement peut produire une qualité de catégorisation plus élevée qu'un même RNA qui utilise le traitement séquentiel. Ce gain en qualité s'obtient avec un temps de réponse supplémentaire qui est peu coûteux. Finalement, le traitement par ré-ordonnancement offre une alternative intéressante au traitement par lot en termes du compromis entre la qualité des catégorisations et le temps de réponse.

L'ambiguïté dans le choix d'une catégorie par patron est un élément qui contribue à la dégradation des résultats d'un RNA comme fuzzy ART. Ce type d'ambiguïté dépend de l'ordre de d'apprentissage des patrons d'entrée et de la complexité de l'environnement à traiter (dispersion, chevauchement, etc., des données). Dans ce chapitre, l'ambiguïté a été utilisée comme critère pour modifier l'ordre d'apprentissage des patrons d'entrée. Un autre type de stratégie pour réduire l'impact des décisions ambiguës consiste à se servir de l'ambiguïté plus directement pour modifier la précision des bornes de décision. Il s'agirait d'explorer différentes représentations de catégories, fonctions de choix, lois d'apprentissages qui permettent de raffiner les bornes de décision selon l'ambiguïté. Le reste du chapitre décrit un exemple d'approche de ce type pour fuzzy ART.

Supposons que chaque catégorie j qui devient commise se voit allouée un *vecteur* de vigilance $\rho_j = (\rho_{j1}, \rho_{j2}, ..., \rho_{jM})$ qui est adaptatif. Un élément ρ_{ji} du vecteur est spécifique à la dimension $i = 1, 2, ..., M$ de l'espace d'entrée. Chacun est initialisé avec une valeur fixe égale au paramètre de vigilance original, $\rho_{ji} = \rho$. Lorsqu'un patron

d'entrée mène à une décision ambiguë, l'ensemble des catégories j qui participent au conflit sont retenues. Leurs paramètres ρ_{ji} sont alors augmentés localement selon une fonction de coût C_i qu'on cherche à minimiser:

$$\rho'_{ji} = \rho_{ji} - \eta \frac{\partial C_i}{\partial \rho_{ji}} , \tag{4.10}$$

où η règle la vitesse d'augmentation de ρ_{ji}. Cette fonction de coût peut dépendre de plusieurs facteurs, tels que la proximité de la dimension i de l'hyperrectangle j au patron d'entrée, le fait que ρ_{ji} correspond à la catégorie $j = J$, etc. Ensuite, le test de vigilance de fuzzy ART est modifié pour utiliser le vecteur de vigilance ρ_J. Puisque la sélection finale des catégories J est contrôlée par ce test, cette approche permet de représenter, dans l'espace d'entrée, des régions qui comportent de l'ambiguïté. Les catégories sont apprises avec différentes résolutions, selon l'ambiguïté qui est perçue dans l'environment. Il est à noter que cette stratégie permet aussi d'éliminer une catégorie si on observe trop d'ambiguïté.

Chapitre 5

Un réseau de neurones avec fusion "*what-and-where*" pour la reconnaissance d'émetteurs radars

Dans le deuxième volet de cette thèse, les RNA basés sur l'apprentissage supervisé (classificateurs) sont appliqués à l'identification des types de radar associés aux impulsions interceptées. On suppose l'existence d'informations a priori (sous la forme d'un lot de données) qui décrit le type d'émetteur qu'on est susceptible de rencontrer dans l'environment. Plus précisément, un RNA est requis pour effectuer la classification de patrons (*i.e.*, impulsions) selon leur type de source (*i.e.*, type de radar).

Un système de MSE radar qui exploite un RNA classificateur peut prendre plusieurs formes différentes. Dans ce chapitre, on suppose que le RNA classificateur se charge

de prédire le type de radar qui émet une impulsion quelconque. Il fait partie d'un système de reconnaissance qui assigne un numéro de piste aux PDW qui n'en n'ont pas reçu lors d'une phase préliminaire de désentrelacement. Les types d'émetteurs radars sont alors identifiées directement, sans triage, ni recherche dans une bibliothèque de MSE.

Évidemment, cette approche est complémentaire aux systèmes de MSE radar traditionnels, puisqu'elle s'applique seulement après que des plateformes de collection ont été déployées dans l'environment. Dans ce contexte, les données sont analysées et peuvent servir pour l'entraînement d'un RNA classificateur. Le fait d'entraîner un RNA avec des données extraites du théatre d'opération peut offrir une meilleure précision que les systèmes conventionels. De plus, il n'est plus nécessaire de construire, ni de maintenir une bibliothèque de MSE pour décrire explicitement les types de radar qu'on peut rencontrer dans l'environement. Cette tâche est généralement très complexe et coûteuse. Finalement, avec un RNA classificateur qui apprend de façon incrémentale, il est possible de raffiner les descriptions, et d'en ajouter de nouvelles, sans devoir refaire l'entrainement sur le lot de données au complet.

Ce chapitre résume la seule contribution[1] du deuxième volet — la proposition d'un RNA à fusion "what-and-where" pour la reconnaissance du type d'émetteur radar associé à chaque train d'impulsions. L'architecture de ce RNA est basée

[1]Les travaux pour cette contribution ont été entamés lors d'un stage au Department of Cognitive and Neural Systems de l'Université de Boston, avec la collaboration du Prof. Stephen Grossberg et du Dr. Mark Rubin.

sur l'agencement de trois sous-systèmes: un RNA classificateur, un sous-système de catégorisation en-ligne, et un sous-système d'accumulation de réponses.

La séquence de PDW traitée par le RNA à fusion "what-and-where" est partitionnée en deux séquences distinctes qu'on nomme "what" et "where"[2]. Les paramètres "what" forment l'entrée pour le RNA classificateur qui prédit les types de radar associés aux impulsions. Entre temps, les paramètres "where" forment l'entrée pour le sous-système de catégorisation en-ligne qui sépare les impulsions transmises par différents émetteurs. Ce dernier assigne alors un numéro de piste (*i.e.*, de catégorie) à chaque impulsion. Le sous-système d'accumulation de réponses permet de fusionner les réponses du RNA classificateur avec celles du sous-système de catégorisation. L'accumulation de réponses est réalisée par un ensemble de modules d'accumulation, un module par piste. Lorsqu'une piste est assignée à un PDW, le module d'accumulation correspondant est activé, et il accumule la prédiction du classificateur. L'accumulation de ces prédictions, selon chaque piste, permet d'identifier les émetteurs d'après une séquence d'impulsions, pour améliorer la précision.

Une mise-en-oeuvre particulière du RNA à fusion "what-and-where" a été choisie pour démontrer le concept. Elle combine une variante du RNA fuzzy ARTMAP [24] pour faire la classification, avec un algorithme qui exécute une association du type plus-proche-voisin et le filtrage de Kalman [12] pour faire la catégorisation en-ligne.

[2]Les paramètres de type "what" qu'on retrouve dans les PDWs sont les caractéristiques intrinsèques de modèles radars (*e.g.*, la fréquence). Ces paramètres peuvent servir à l'identification des types d'émetteurs. Les paramètres de type "where" sont liés à l'état d'un émetteur spécifique dans l'environnement (*e.g.*, angle d'arrivée), mais ils ne sont pas compilés dans un bibliothèque de MSE.

Afin de simuler le RNA à fusion "what-and-where," ces deux blocs ont été connectés avec le sous-système d'accumulation de réponses. Le lot de données radars utilisé lors de ces simulations a été collecté dans le champs et analysé par DREO. Il est constitué d'environ 52 000 impulsions qui appartiennent à 15 différents types de radar. Les paramètres "what" des PDW sont le PRI, PW et RF, tandis que les paramètres "where" sont Brg et PA.

Le fuzzy ARTMAP est un bon candidat pour le RNA classificateur car il permet d'effectuer un apprentissage incrémental. Cependant, quelques modifications ont due être incorporées afin de l'appliquer aux MSE. En effet, un problème de cohérence existe, car un lot d'entrainement peut contenir des impulsions quasi-identiques, mais qui appartiennent à différents types de radar. D'autre part, le RNA classificateur peut être confronté à des données incomplètes, lors de l'entrainement ou du test. Le RNA doit pouvoir traiter des impulsions avec des paramètres absents, et qui n'appartiennent pas à des types de radar représentés dans le lot d'entrainement.

L'article suivant contient plus de détails sur le RNA à fusion "what-and-where" pour la reconnaissance d'émetteurs radars:

GRANGER, E., RUBIN, M.A., GROSSBERG, S., et LAVOIE, P.,
"A What-and-Where fusion neural network for recognition and tracking of multiple radar emitters," *Neural Networks*, **14:3**, 325-344 (2001).

La suite de ce chapitre reproduit cet article. La dernière section du chapitre porte sur l'impact des résultats de cette contribution.

A What-and-Where fusion neural network for recognition and tracking of multiple radar emitters

Eric Granger[1,2], Mark A. Rubin[3,4], Stephen Grossberg[3] and Pierre Lavoie[1]

[1] Defence Research Establishment Ottawa, Department of National Defence, 3701 Carling Ave., Ottawa, Ontario, K1A 0Z4, Canada.

[2] Department of Electrical and Computer Engineering, École Polytechnique de Montréal, C. P. 6079, Station "centre-ville," Montreal, Quebec, H3C 3A7, Canada.

[3] Department of Cognitive and Neural Systems and Center for Adaptive Systems, Boston University, 677 Beacon St., Boston, MA 02215, USA.

Abstract

A neural network recognition and tracking system is proposed for classification of radar pulses in autonomous Electronic Support Measure systems. Radar type information is combined with position-specific information from active emitters in a scene. Type-specific parameters of the input pulse stream are fed to a neural network classifier trained on samples of data collected in the field. Meanwhile, a clustering

[4]**Current address:** Sensor Exploitation Group, M.I.T. Lincoln Laboratory, 244 Wood St., Lexington, MA 02420, USA

algorithm is used to separate pulses from different emitters according to position-specific parameters of the input pulse stream. Classifier responses corresponding to different emitters are separated into tracks, or trajectories, one per active emitter, allowing for more accurate identification of radar types based on multiple views of emitter data along each emitter trajectory. Such a What-and-Where fusion strategy is motivated by a similar subdivision of labor in the brain.

The fuzzy ARTMAP neural network is used to classify streams of pulses according to radar type using their functional parameters. Simulation results obtained with a radar pulse data set indicate that fuzzy ARTMAP compares favorably to several other approaches when performance is measured in terms of accuracy and computational complexity. Incorporation into fuzzy ARTMAP of negative match tracking (from ARTMAP-IC) facilitated convergence during training with this data set. Other modifications improved classification of data that include missing input pattern components and missing training classes. Fuzzy ARTMAP was combined with a bank of Kalman filters to group pulses transmitted from different emitters based on their position-specific parameters, and with a module to accumulate evidence from fuzzy ARTMAP responses corresponding to the track defined for each emitter. Simulation results demonstrate that the system provides a high level of performance on complex, incomplete and overlapping radar data.

5.1 Introduction

Radar Electronic Support Measures (ESM) involve the search for, interception, location, analysis and identification of radiated electromagnetic energy for military purposes. ESM hereby provide valuable information for real-time situation awareness, threat detection, threat avoidance, and for timely deployment of counter-measures [16] [40] [63] [121] [122] [123] [133] [139].

A critical function of radar ESM is the real-time identification of the radar type associated with each pulse train that is intercepted. Current approaches typically involve sorting incoming radar pulses into individual pulse trains, then comparing the pulse train characterizations with a library of parametric descriptions, which yields a list of likely radar types. This task is challenging owing to increases in environment density (*e.g.*, pulse Doppler radars that transmit hundreds of thousands of pulses per second); dynamically changing environments; multiplication and dispersion of the modes for military radars; agility in parameters like pulse repetition interval, radio frequency and scan; unknown and reserve modes for which no ESM library entry exists; overlaps between the parameters of different radar types in the ESM library; and noise and propagation effects that lead to erroneous or incomplete signal characterization. These aspects of the problem place severe stress on current ESM systems.

It this paper, an alternative approach is examined. A new recognition system combines diverse sources of information in order to predict the most likely radar type

for each intercepted pulse. Type-specific parameters of the input pulse stream are used to classify pulses according to radar type, while environment-specific parameters are used to separate pulses corresponding to active emitters. Such separation allows the system to accumulate the classifier's responses for each emitter, and therefore to predict an emitter's identity based on one or multiple responses.

A key component of the new recognition system is a neural network classifier that is trained to determine the types of radar emitters present in the environment. The system learns autonomously, directly from data collected in the field, to identify pulse parametric ranges corresponding to specific radar types. Aside from avoiding some of the pulse sorting, training on data from the actual environment to approximate an unknown mapping function may deliver greater predictive accuracy. Furthermore, the need for by-hand construction of an emitter library is obviated.

From an ESM standpoint, training a system directly on radar data is a radical departure from current practice. At present, data are collected, analyzed, combined with prior information, and distilled into ESM libraries off-line by skilled analysts. New libraries, containing explicit radar type descriptions, are disseminated to the field as needed. One inconvenience of the conventional approach is that it is very complex, time-consuming, and does not allow for rapid modifications of ESM libraries upon discovery of new radar modes in the field. Using a neural network able to learn incrementally offers a framework for refining familiar, or adding unfamiliar, radar type descriptions on the fly.

In a particular realization of the recognition system, fuzzy ARTMAP [21] [24] is considered for neural network classification of pulses from their type-specific parameters, whereas nearest-neighbor matching with a bank of Kalman filters [7] [12] is considered for separation of pulses from their environment-specific parameters. The features of the system include: (1) By virtue of the fast-learning capabilities of ARTMAP neural networks [21]- [24], new information from familiar or unfamiliar radar type classes can be learned incrementally without retraining on the whole data set. (2) Classification decisions can be made on the basis of single pulses or, for greater accuracy, on the basis of streams of pulses that have been determined to come from a given emitter. This determination is performed either by a time-of-arrival (TOA) deinterleaver or, when TOA deinterleaving is not practical, by a Kalman filter that tracks the bearing and amplitude of the pulses. The system is thus an example of a neural system combining temporal — When — and positional — Where — information with featural — What — information to arrive at its decision. It is well-known that the mammalian brain also divides What and Where computations into separate, but mutually interacting, cortical processing streams. Our What-and-Where model shows how this strategy can generate higher accuracy in identifying radar emitters. (3) The "familiarity discrimination" extension of fuzzy ARTMAP, called ARTMAP-FD [26] [27], allows the system not only to detect pulses from unfamiliar radar type classes (not presented during training), but to determine the threshold for rejection based on all of the training data, without the need for holding back a portion for a

validation set. This ability to determine the reject threshold on-line makes possible on-line learning of pulses from unfamiliar classes (LUC) [58]. (4) New extensions to fuzzy ARTMAP permit both training and testing on data with missing components, and the use of unlabeled training data [58].

Conventional approaches to, and challenges of, radar type identification in radar ESM systems are reviewed in the next section. A system-level overview of our novel neural network recognition system is provided in Section 5.3. A radar pulse data set used for proof-of-concept simulations is presented in Section 5.4. The three main components that form a specific implementation of the recognition system are described in Sections 5.5 through 5.7. In Section 5.5, the fuzzy ARTMAP neural network is applied to the classification of pulses according to radar type from functional, type-specific parameters. Then, aspects of this network for dealing with incomplete radar data are proposed and tested. In Section 5.6, a module for clustering incoming pulses by emitter based on environment-specific parameters is described. In Section 5.7, a module that accumulates evidence from fuzzy ARTMAP responses corresponding to the tracked emitters is proposed. Finally, these three components are connected, and global simulation results using this particular realization of the entire recognition system are presented and discussed.

5.2 Radar Electronic Support Measures

5.2.1 Overview

The basic functionality of current radar ESM approaches can be decomposed into three tasks: reception of radar signals, grouping of pulses according to emitter, and identification of corresponding radar types.

Radar signals are passively intercepted by the receiver portion of the ESM system. In typical theaters of operation, intercepted signals are a mixture of electromagnetic pulses transmitted from, typically, several sources. Simultaneous illumination by these sources causes overlap and interleaving of the received pulses. Upon detection of a radar pulse, most receivers measure the pulse amplitude (PA), pulse width (PW), radio frequency of the carrier wave (RF) and time-of-arrival (TOA). Direction-finding receivers also measure the bearing (Brg), while advanced receivers also measure the modulation on pulse (MOP). Once parameter values have been measured for a pulse, they are digitized and assembled into a data structure called a Pulse Descriptor Word (PDW). For the reader's convenience, a list of the radar ESM abbreviations used in this paper is given in Table 5.1.

The stream of successive PDWs is fed to a grouping module, which performs either TOA deinterleaving, or sorting, or both. In short, this module seeks to recover pulse trains and their inter-pulse structure prior to further analysis. This involves progressively grouping pulses that appear to have been transmitted from the same

Tableau 5.1: List of ESM abbreviations

Abbreviations	Definition
Brg	bearing
ESM	electronic support measures
EW	electronic warfare
MOP	modulation on pulse
PA	pulse amplitude
PDW	pulse descriptor word
PPI	pulse-to-pulse interval
PRI	pulse repetition interval
PW	pulse width
RF	radio frequency
TOA	time of arrival

emitter. An emitter is an instance of a radar type, and it is not uncommon to observe several emitters of a same type all being active in a theater of operation. A single type of radar can also operate under several different modes to perform various functions. To each group of pulses is associated a *track*. A track consists of statistical PDW parameters, plus other parameters that are derived from the sequence of grouped PDWs, like the pulse repetition interval (PRI).

Pulse grouping techniques either exploit the difference in TOA between pulses, or the actual parameters in the PDWs. Parametric ranges are associated with tracks, and updated to reflect changes in the emitter's characteristics over time. TOA deinterleaving attempts to discover consistent patterns in the TOA of pulses using techniques such as TOA difference histogramming [40] [100] [139]. If TOA consistencies are found, and these correlate with radar definitions compiled in an ESM library,

then the corresponding pulses are grouped based on PRI, and stripped away from the input stream of PDWs. Sorting attempts to group pulses based on the likeliness of their PDW parameters such as RF, PW and Brg. Gating [31] [40] [117] or clustering [3] [43] [140] techniques are commonly used to this end.

Identification makes use of an ESM library where are stored the parametric descriptions of known radar types, and attempts to assign a single radar type to each track. Incidentally, the parametric ranges of various types can overlap in the library, and multiple candidates can appear plausible for the same track, a situation known as an "ambiguity." Therefore, a list of likely radar types is often displayed and monitored over time for every track, along with a confidence rating, threat level, latest bearings, and so on. Further analysis can assist an ESM operator in revealing mode changes in emitters, links between emitters, and inferred platforms.

5.2.2 Challenges

Pulse grouping and radar type recognition keep evolving in response to the following defense trends: radar signals are more agile; power management and low probability of intercept waveforms in advanced threats reduce response time; ESM libraries are expensive to maintain; and unmanned platforms require autonomous ESM. These trends motivate this work, and call for more powerful ESM approaches.

The multiplication of radar modes is the result of computer control and the ease with which parameters such as RF and PRI can be changed. From an ESM stand-

point, this means libraries that grow larger and more complex. Agility in parameters like RF and PRI can make pulse grouping very difficult.

A shorter response time requires faster pulse grouping, as well as identification using fewer pulses. In addition, the occurrence of low power waveforms implies that pulses near the receiver detection threshold may be dropped, and hence that pulse grouping must work satisfactorily on sparse data. Response time is critical if threats are to be avoided, or self-protection measures such as chaff dispensing, maneuvering, or electronic jamming, are to be successful.

It is difficult and expensive to maintain comprehensive ESM libraries that accurately reflect each specific operational environment. Library construction requires explicit modeling of known radar systems, based on prior information and data that is not necessarily extracted from the local environment. This task is complex, tedious, and prone to error because some radar types are difficult to describe. Owing to the multiplication of modes, it is not uncommon for a library to be incomplete and to contain erroneous data. In addition, threats could deliberately reserve some of their modes for use during war time. Radar type identification must therefore be tolerant to such shortcomings. For instance, classical parametric approaches to pattern classification [45] [50] are generally less effective when the underlying radar type class distributions are incomplete and/or uncertain.

Personnel reduction in the Armed Forces, as well as the deployment of ESM on autonomous platforms such as unmanned aerial vehicles raises the expectations for

ESM. Without an operator to interpret ESM output and provide discernment, ESM systems must achieve enhanced accuracy and reliability. In light of these trends, alternative approaches are sought for pulse grouping and radar type identification.

5.3 A neural network for radar type identification

5.3.1 Adaptive learning and ESM

In this section, a new approach is described for radar type recognition. When collection platforms are brought into a theater of operations prior to military interventions, data from radars of interest can be collected and analyzed. Collection platforms include in-theater tactical aircraft and ships, unmanned aerial vehicles, and stand-off assets like electronic warfare (EW) aircraft. Data is collected prior to, or during, the conflict and analyzed either on-line (*e.g.*, on a ship) or off-line from electronic intelligence readings. Whereas the data are normally combined with prior information from other environments, and distilled by analysts into library entries, this paper explores their use for training an artificial neural network recognition system. Once trained on the data, the network can classify the pulses without the grouping process, thereby making use of a priori information early in the processing chain. As discussed in the following, this approach offers several potential advantages.

Firstly, training on real data gathered in the field may yield higher classification accuracy. Adaptive learning algorithms used to train neural network classifiers con-

stitute an interesting alternative to the explicit modeling currently employed in ESM libraries, since they can estimate unknown input-to-class mapping functions directly from the training set. Their supervised learning process involves a prescription to combine some prior assumptions (*i.e.*, a set of possible mapping functions) with the training data set, to approximate an unknown mapping function. This mapping is then used to generalize, that is, predict output classes (radar types) for unlabeled input patterns (pulses).

Secondly, an attractive feature of neural networks is the convenience of handling incomplete radar type descriptions and incomplete data. Since it is impossible to have an exhaustive data set to train a network, recognition of new radar types encountered during operations is important. Neural network classifiers that allow on-line incremental learning provide a consistent framework for automatically refining the description of familiar radar types, as well as for detecting unfamiliar radar types and learning their description as operations unfold.

Lastly, although it is not the focus of this paper, the massively parallel architecture of neural networks, when implemented on appropriate hardware, can provide extremely fast and fault tolerant processing of PDWs. Such a response would also be somewhat tolerant to incomplete and noisy data, yielding graceful performance degradation. The on-line nature of the networks also eliminates the need to collect batches of pulses prior to processing, as in current approaches.

Neural network techniques have previously been applied to several aspects of

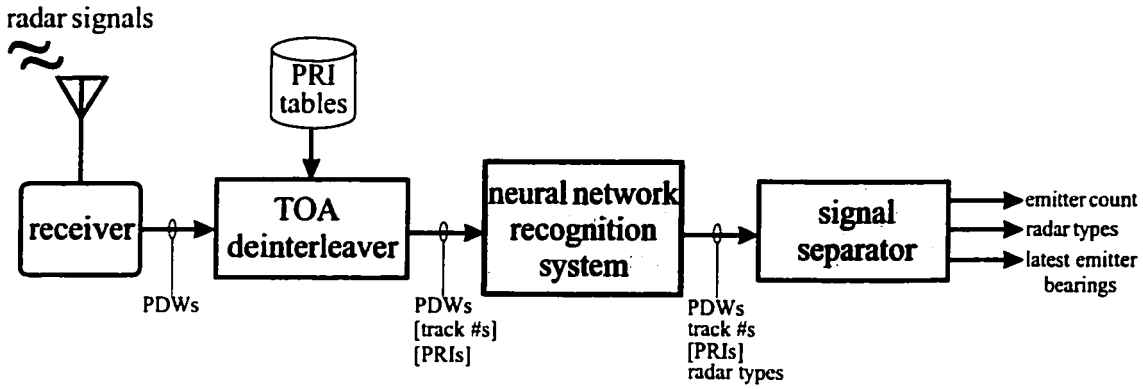


Figure 5.1: High level block diagram of a radar ESM system that uses a neural network recognition system. Brackets indicate that the corresponding field may be empty for some pulses.

radar ESM processing [123] [124], including parameter measurement [124], PDW sorting [4] [79] [101] [105] [135], and radar type recognition [96] [98] [116] [128]. A new neural network recognition and tracking system for classification of radar pulses is described next.

5.3.2 Overview of ESM model

One possible embodiment of a neural network recognition system into an ESM system is depicted in Figure 5.1. First a TOA deinterleaver uncovers periodicities in the TOA of input PDWs. Whenever grouping pulses is straightforward, it forms tracks and assigns a track number and a PRI to each grouped pulse. TOA deinterleaving continues to play an important role in ESM since it suffices to group the pulses of emitters having simple PRI patterns, like many high duty cycle Pulse Doppler radars. Besides, the PRI parameters themselves are useful for classification of pulses according to radar type.

The neural network recognition system receives all the PDWs, some of which have track numbers and PRI parameters. It has already been trained off-line on data from known radar types. The neural network weights replace the ESM library, and can be periodically updated by learning from radar data collected during operations. The neural network outputs a prediction of the radar type for every PDW, and assigns a track number to the PDWs that did not get one from the TOA deinterleaver. Track assignment is autonomous, and takes place regardless of the radar type classification. The radar type classification does, however, take into account track assignment.

The remaining module in Figure 5.1 is a signal separator, which receives all the PDWs along with their track numbers, radar types, and, whenever available, PRI parameters. The signal separator is responsible for the final track assignment and for distilling the stream of PDWs into emitter reports that are periodically updated. The final assignment takes into account the radar type recognition previously performed by the neural network. Emitter reports contain, for instance, the type of each emitter with its latest bearing.

5.3.3 What-and-Where model architecture

5.3.3.1 What and Where data streams

The PDW stream may be partitioned into two data streams called *What* and *Where*. This division is motivated by a similar subdivision of parallel processing in the primate cerebral cortex into a What stream for recognizing objects, and a Where

stream for localizing their position in space. See [68] for a theoretical discussion of these processing streams. Here the What data stream consists of parameters that characterize the functional aspects of radar systems. Such parameters include RF, PW and PRI. Since these parameters correspond to data typically compiled in ESM libraries, they are directly useful for radar type recognition. The Where data stream consists of context-specific parameters. This stream is defined by parameters, such as Brg and PA, that indicate the status (*e.g.*, position) of specific emitters in the environment. These parameters are less useful than What parameters for radar type recognition, but are important for grouping pulses into tracks, or trajectories. The definition of Where parameters can be extended to include emitter specific parameters that cannot be recorded *a priori* due to practical or physical considerations. Such parameters may prove effective for pulse grouping, irrespective of their value for radar type recognition. Some MOP parameters could, for example, be assigned to this processing stream.

5.3.3.2 Distinct What and Where data processing

The internal architecture of a neural network recognition system is shown in Figure 5.2. It is composed of three subsystems: neural network classification, clustering and evidence accumulation. These subsystems cooperate to predict the most likely radar type for each incoming pulse.

Prior to on-line operation, the neural network classification module is trained, via

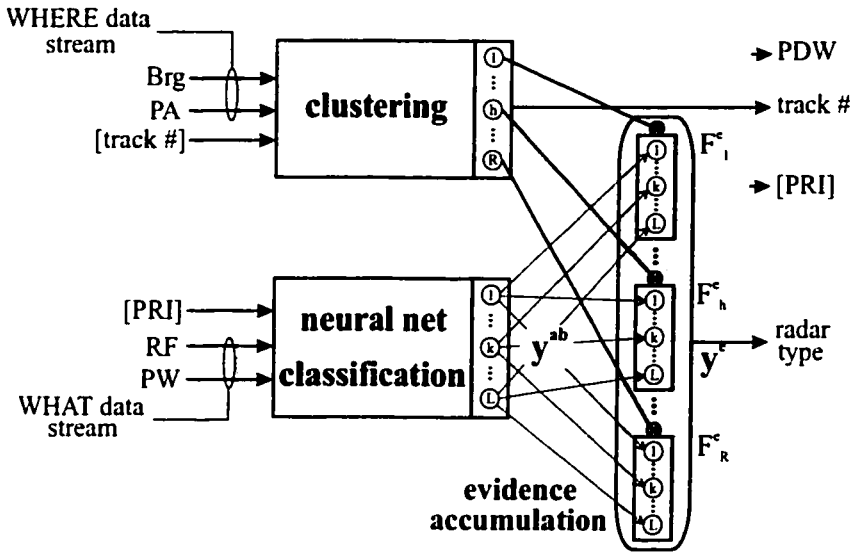


Figure 5.2: Internal architecture of the neural network recognition system.

supervised learning, using a data set of radar pulses collected in the field, and labeled with their respective radar type. Only What parameters are employed for training. PRI may be supplied if it is available.

During on-line operation, the recognition system accepts the stream of PDWs corresponding to intercepted radar pulses, as well as the track number and PRI of each pulse grouped by the TOA deinterleaver. Since each PDW is composed of predefined What and Where parameters, these can be automatically separated and fed to the neural network classification and clustering subsystems, respectively.

For each pulse, the neural network classification subsystem accepts What parameters, including PRI when available, and yields a prediction of the radar type. This prediction takes the form of a response pattern denoted by $\mathbf{y}^{ab}$ (refer to Figure 5.2). Meanwhile, the clustering subsystem attempts to group pulses into tracks based on

Where parameters. If a track number is supplied by the TOA deinterleaver, then the output of this subsystem is bypassed. Whether the subsystem produces the track number itself, or receives it from the TOA deinterleaver, it maintains an up-to-date picture of the number and activity of radar emitters illuminating the ESM system.

5.3.3.3 Evidence accumulation

In conventional radar ESM systems, Where information is employed early in the processing chain (*i.e.*, during track formation) to reduce data and subsequent computational costs. The neural network recognition system embodies an alternative approach that integrates both What and Where information streams for better recognition prior to the data reduction. Fusion of responses from the classification subsystem and the clustering subsystem is accomplished via evidence accumulation, which emulates the brain process of working memory; *e.g.*, [13], and [14]. Response patterns $\mathbf{y}^{ab}$ obtained from classification are hereby accumulated over time according to tracks, that is, groupings determined from Where data.

Track numbers obtained from the clustering module dictate the emitters to which PDWs are associated, and drive the evidence accumulation. This evidence accumulation is implemented as a set of evidence accumulation fields, with each field F_h^e corresponding to a track $h = 1, 2, ..., R$. Assignment of a track $h = H$ to a PDW activates an evidence accumulation field F_H^e that accumulates the classification module's response pattern $\mathbf{y}^{ab}$. Such accumulation produces a radar type response pattern

for each PDW, denoted by $\mathbf{y}^e$ (see Figure 5.2), which is obtained from one or more responses $\mathbf{y}^{ab}$. As discussed in Section 5.7, exploiting both What and Where information sources can thereby enhance the system's classification accuracy.

5.4 Radar pulse data

The data set used for the computer simulation contains approximately 100,000 consecutive radar pulses gathered over 16 seconds by the Defense Research Establishment Ottawa during a field trial. After the trial, an ESM analyst manually separated trains of pulses coming from different emitters. Each pulse was then tagged with two labels: a radar type number and a mode number. Since ESM trials are complex and never totally controlled, not all pulses could be tagged and a sizable residue was obtained. Residue pulses were discarded for this study.

The parameters used are Brg, PA, PRI, PW and RF (refer to Table 5.1). From this point on, a PDW is denoted by $(\mathbf{a};\mathbf{b})$. Patterns in the What and Where data streams are defined by $\mathbf{a}$ = (PRI, PW, RF) and $\mathbf{b}$ = (Brg, PA), respectively. The two Where parameters, Brg and PA, are specific to the environment, and thus are not employed for training the neural network classification module.

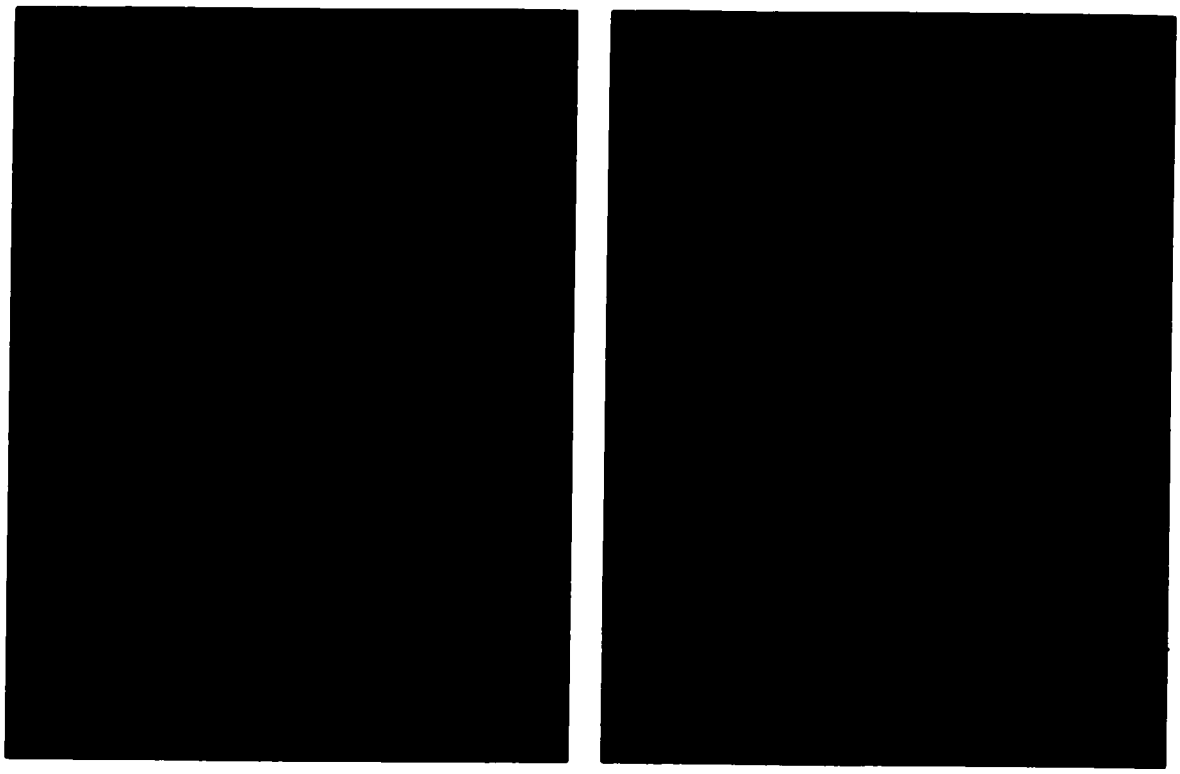
Brg, PA, PW and RF are automatically produced by the receiver on each individual pulse, whereas PRI is derived from the difference in time-of-arrival (TOA) between pulses from the same emitter. For simplicity, it is assumed that, as a part of the preprocessing, a simple TOA deinterleaver has grouped the pulses belonging

to each active emitter mode, and then computed their respective PRI values. Note that, since at least two successive pulses are required to compute a PRI value, the first pattern from each active emitter mode was omitted from the simulations. Also, owing to the circular scanning action of some radar emitters, pulses are recorded in bursts. The first pulse of each scan (or burst) was also omitted. Finally, the parameters were linearly normalized so that $a_i, b_j \in [0, 1]$, for $i = 1, 2, 3$ and $j = 1, 2$.

Once tagged and deinterleaved, the data used to train and test the neural network recognition system contain 52,192 radar pulses from 34 modes, each one belonging to one of 15 different radar types. The data feature bursts of high pulse densities, multiple emitters of the same type, modes with overlapping parametric ranges, radars transmitting at different pulse rates, and emitters switching modes. The sophistication of the radar types range from simple (constant RF and PRI) to fairly complex (pulse-to-pulse agility in RF and PRI). Figure 5.3 displays a 0.5 second sample of the radar pulse data set used for simulations. This particular example contains 1123 pulses from 8 emitters belonging to 7 different radar types, with agility and overlap of parameters, and an emitter switching modes.

5.5 An ARTMAP neural network for classification

An enhanced ARTMAP neural network is used to classify incoming radar pulses according to radar type from parameters in the What data stream. ARTMAP refers to a family of neural network architectures capable of fast, stable, on-line, unsu-



(a) What parameters (RF and PW) vs TOA. (b) Where parameters (Brg and PA) vs TOA.

Figure 5.3: A sample of the radar pulse data set used for simulations.

pervised or supervised, incremental learning, classification, and prediction [22] [24]. ARTMAP networks have several attractive features for applications such as electronic support measures (ESM). Because they can perform fast, stable, on-line, incremental learning, they can learn from novel events encountered in the field. Neural network classifiers such as the popular Multilayer Perceptron (MLP) [120] and Radial Basis Function (RBF) [32] require off-line retraining on the whole data set, through a lengthy iterative slow-learning procedure, to learn new patterns from existing radar

type classes or a new radar type class. ARTMAP networks can also perform familiarity discrimination to avoid meaningless guesses on patterns from unfamiliar radar types classes [26] [27] [56]. Furthermore, they can represent radar type classes using one or more prototypes, which appears desirable for handling radar types having several modes of operation. The k-Nearest-Neighbor (kNN) [38] and Probabilistic Neural Network (PNN) [129] classifiers would usually require greater computational resources to store all the training set patterns, and to yield on-line predictions. Finally, ARTMAP networks lend themselves well to high speed parallel processing, which is critical for real-time identification.

5.5.1 Fuzzy ARTMAP

ARTMAP is often applied using the simplified version shown in Figure 5.4. It is obtained by combining an ART unsupervised neural network [20] with a map field. Fuzzy ARTMAP [24] can process both analog and binary-valued input patterns by employing fuzzy ART [21] as the ART network.

The fuzzy ART neural network consists of two fully connected layers of nodes: an M node input layer, F_1, and an N node competitive layer, F_2. A set of real-valued weights $\mathbf{W} = \{w_{ij} \in [0,1] : i = 1,2,...,M;\ j = 1,2,...,N\}$ is associated with the F_1-to-F_2 layer connections. Each F_2 node j represents a recognition category that learns a prototype vector $\mathbf{w}_j = (w_{1j}, w_{2j}, ..., w_{Mj})$. The F_2 layer is connected, through learned associative links, to an L node map field F^{ab}, where L is the number

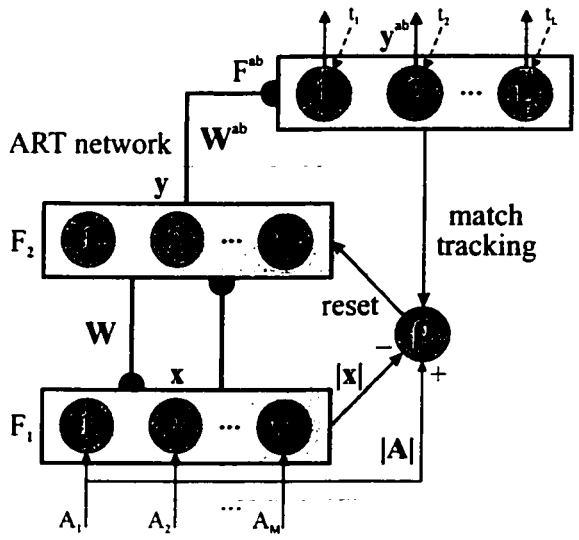


Figure 5.4: An ARTMAP neural network architecture specialized for pattern classification.

of classes in the output space. A set of binary weights $\mathbf{W}^{ab} = \{w_{jk}^{ab} \in \{0,1\} : j = 1,2,...,N;\ k = 1,2,...,L\}$ is associated with the F_2-to-F^{ab} connections. The vector $\mathbf{w}_j^{ab} = (w_{j1}^{ab}, w_{j2}^{ab}, ..., w_{jL}^{ab})$ links F_2 node j to one of the L output classes.

During training, ARTMAP classifiers perform supervised learning of the mapping between training set vectors $\mathbf{a} = (a_1, a_2, ..., a_m)$ and output labels $\mathbf{t} = (t_1, t_2, ..., t_L)$, where $t_K = 1$ if K is the target class label for $\mathbf{a}$, and zero elsewhere. The following algorithm describes fuzzy ARTMAP learning:

1. Initialization. Initially, all the F_2 nodes are uncommitted, all weight values w_{ij} are initialized to 1, and all weight values w_{jk}^{ab} are set to 0. An F_2 node becomes committed when it is selected to code an input vector $\mathbf{a}$, and is then linked to an F^{ab} node. Values of the learning rate $\beta \in [0,1]$, the choice $\alpha > 0$, and the baseline vigilance $\bar{\rho} \in [0,1]$ parameters are set.

2. Input pattern coding. When a training pair $(\mathbf{a}, \mathbf{t})$ is presented to the network, $\mathbf{a}$ undergoes a transformation called complement coding, which doubles its number of components. The complement-coded input pattern has $M = 2m$ dimensions and is defined by $\mathbf{A} = (\mathbf{a}, \mathbf{a}^c) = (a_1, a_2, ..., a_m, a_1^c, a_2^c, ..., a_m^c)$, where $a_i^c = (1 - a_i)$, and $a_i \in [0, 1]$. The vigilance parameter ρ is reset to its baseline value $\bar{\rho}$.

3. Prototype selection. Pattern $\mathbf{A}$ activates layer F_1 and is propagated through weighted connections $\mathbf{W}$ to layer F_2. Activation of each node j in the F_2 layer is determined by the *Weber law choice function*:

$$T_j(\mathbf{A}) = \frac{|\mathbf{A} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}\ , \tag{5.1}$$

where $|\cdot|$ is the norm operator, $|\mathbf{w}_j| \equiv \sum_{i=1}^{M} |w_{ij}|$, $\wedge$ is the fuzzy AND operator, $(\mathbf{A} \wedge \mathbf{w}_j)_i \equiv \min(A_i, w_{ij})$, and α is the user-defined *choice parameter*. The F_2 layer produces a binary, winner-take-all pattern of activity $\mathbf{y} = (y_1, y_2, ..., y_N)$ such that only the node $j = J$ with the greatest activation value $J = \arg\max\{T_j : j = 1, 2, ..., N\}$ remains active; thus $y_J = 1$ and $y_j = 0, j \neq J$. If more than one T_j is maximal, the node j with the smallest index is chosen. Node J propagates its top-down expectation, or prototype vector $\mathbf{w}_J$, back onto F_1 and the *vigilance test* is performed. This test compares the degree of match between $\mathbf{w}_J$ and $\mathbf{A}$ against the

dimensionless *vigilance parameter* ρ:

$$\frac{|\mathbf{A} \wedge \mathbf{w}_J|}{M} \geq \rho \, . \tag{5.2}$$

If the test is passed, then node J remains active and resonance is said to occur. Otherwise, the network inhibits the active F_2 node (*i.e.*, T_J is set to 0 until the network is presented with the next training pair $(\mathbf{a}, \mathbf{t})$) and searches for another node J that passes the vigilance test. If such a node does not exist, an uncommitted F_2 node becomes active and undergoes learning. The depth of search before an uncommitted node is selected is determined by the choice parameter α.

4. Class prediction. Pattern $\mathbf{t}$ is fed directly to the map field F^{ab}, while the F_2 category $\mathbf{y}$ learns to activate the map field via associative weights $\mathbf{W}^{ab}$. The F^{ab} layer produces a binary pattern of activity $\mathbf{y^{ab}} = (y_1^{ab}, y_2^{ab}, ..., y_L^{ab})$ in which the most active F^{ab} node K yields the class prediction $(K = k(J))$. If node K constitutes an incorrect class prediction, then a *match tracking* signal raises the vigilance parameter ρ just enough to induce another search among F_2 nodes in Step 3. This search continues until either an uncommitted F_2 node becomes active (and learning directly ensues in Step 5), or a node J that has previously learned the correct class prediction K becomes active.

5. Learning. Learning input $\mathbf{a}$ involves updating prototype vector $\mathbf{w}_J$, and, if J corresponds to a newly-committed node, creating an associative link to F^{ab}. The

prototype vector of F_2 node J is updated according to:

$$\mathbf{w}'_J = \beta(\mathbf{A} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J \ , \tag{5.3}$$

where β is a fixed *learning rate parameter*. The algorithm can be set to slow learning with $0 < \beta < 1$, or to fast learning with $\beta = 1$. With complement coding and fast learning, fuzzy ART represents category j as an m-dimensional hyperrectangle R_j that is just large enough to enclose the cluster of training set patterns $\mathbf{a}$ to which it has been assigned. A new association between F_2 node J and F^{ab} node K ($k(J) = K$) is learned by setting $w^{ab}_{Jk} = 1$ for $k = K$, where K is the target class label for $\mathbf{a}$, and 0 otherwise. Once the weights $\mathbf{W}$ have converged for the training set patterns, ARTMAP can predict a class label for an input pattern by performing Steps 2, 3 and 4 without any vigilance or match tests. During testing, a pattern $\mathbf{a}$ that activates node J is predicted to belong to class $K = k(J)$.

5.5.2 Comparative simulations

Fuzzy ARTMAP and three other ARTMAP neural networks — ART-EMAP (Stage 1) [25], ARTMAP-IC [28] and Gaussian ARTMAP [141] [142] - have been compared using computer simulations. The kNN and RBF classifiers were included for non-parametric, and semi-parametric [11] reference, respectively.

Prior to each simulation trial, the radar pulse data described in Section 5.4 was partitioned into training and test subsets. 50% of the data from each radar type was

selected at random to form the training subset. Then, the training vectors **a**, along with their radar types labels **t**, were repeatedly presented, until convergence. The same random order was used across presentations. Emitter mode labels were ignored since this paper concerns the classification of pulses according to radar type. Convergence was reached when the sum-squared-fractional-change (SSFC) of prototype weights **W** was less than 0.001 for two successive epochs. An epoch is defined as a presentation of the training subset to a classifier in TOA order. The RBF classifier used in this comparison selects training subset patterns one by one to encode hidden layer nodes [32]. Convergence was reached when the sum-squared-error between actual outputs (resulting from training set patterns) and target outputs fell below 0.01. After convergence, the test subset was presented to the trained classifier for prediction. Throughout this paper, average results are obtained from several independent simulation trials, each one with a different random selection of the training data.

The What patterns consist of 3 parameters: **a** = (PPI, PW, RF). It is assumed that a TOA deinterleaver has correctly grouped the n_k pulses belonging to each active emitter mode k, and then computed the pulse-to-pulse intervals: $\mathrm{PPI}_k(i) = \mathrm{TOA}_k(i) - \mathrm{TOA}_k(i-1)$ for $i = 2, 3, ..., n_k$. Using the PPI to estimate the PRI allows for simple training and testing of neural network classifiers without concern for the PRI agility of some emitters.

Tableau 5.2: Average classification results for the radar data. The "†" indicates that the classifier was unable to converge for the training set on each trial. (Numbers in parentheses are the standard error of the sample mean.)

Classifier	Evaluation criteria (std error)			
	Accuracy	Resources		
	Classification rate	Compression	Memory	Convergence time
k-Nearest-Neighbor ($k = 1$, $d_{\text{cityblock}}$)	99.64% (0.01%)	1.0 (0.0)	80311 (0)	N/A
k-Nearest-Neighbor ($k = 1$, $d_{\text{Euclidean}}$)	99.64% (0.01%)	1.0 (0.0)	80311 (0)	N/A
Radial Basis Function† (spread of RB kernel = 0.05)	99.68% (0.01%)	4.1 (0.1)	6367.9 (3.3)	2123.6 (1.1)
ART-EMAP† (Stage 1) ($\epsilon = 10^{-4}$, $\alpha = .001$, $\beta = 1$, $\bar{\rho} = 0$)	78.28% (1.65%)	222.9 (5.2)	709.2 (15.9)	stopped at 3.6 (0.1)
ARTMAP-IC ($\epsilon = 10^{-4}$, $\alpha = .001$, $\beta = 1$, $\bar{\rho} = 0$)	83.65% (1.90%)	214.0 (4.6)	737.4 (14.5)	3.8 (0.1)
fuzzy ARTMAP† ($\epsilon = 10^{-4}$, $\alpha = .001$, $\beta = 1$, $\bar{\rho} = 0$)	99.56% (0.02%)	217.5 (5.9)	729.6 (19.2)	stopped at 3.6 (0.1)
Gaussian ARTMAP† ($\epsilon = 10^{-3}$, $\gamma = .0025$, $\bar{\rho} = 0$)	99.69% (0.01%)	99.1 (1.2)	1583.4 (17.4)	stopped at 5.9 (0.2)
fuzzy ARTMAP with MT- ($\epsilon = 10^{-4}$, $\alpha = .001$, $\beta = 1$, $\bar{\rho} = 0$)	99.56% (0.01%)	213.2 (6.0)	744.3 (19.2)	3.8 (0.1)

The performance of each classifier was assessed in terms of both the amount of resources required and predictive accuracy. The amount of resources allocated during training is measured in the 3 following ways. *Compression* refers to the average ratio of training patterns to committed F_2 layer nodes. *Memory* is the number of normalized registers needed to store the set of learned prototype vectors, a normalized register being a fixed-size register whose number of bits suffices to store the classifier's real values such as a_i, w_{ji}, ρ, and so on. *Convergence time* is the number of epochs required for the classifier to converge. The predictive accuracy on the test subset is measured using the *classification rate* - the ratio of correctly classified patterns over all test patterns.

Average results from 20 simulation trials of fuzzy ARTMAP are given in Table 5.2, along with the standard error of the sample mean (in parentheses). Parameter settings were selected through trial and error to achieve the best classification rate for the least memory and convergence time during training. Results indicate that fuzzy ARTMAP and Gaussian ARTMAP consistently achieve the highest average classification rates, followed by ARTMAP-IC and ART-EMAP (Stage 1). The classification rates of fuzzy ARTMAP and Gaussian ARTMAP are comparable to those obtained using the kNN and RBF classifiers. ART-EMAP, ARTMAP-IC and fuzzy ARTMAP attain their classification rates with greater compression (and thus require less physical memory to store prototype vectors, and deliver faster fielded performance) than the other classifiers, and take fewer training epochs to converge than Gaussian ARTMAP and

RBF. Overall, fuzzy ARTMAP performs at least as well as the other classifiers in both accuracy and computational complexity, and better than each of them in at least one of these aspects of performance. The reader is referred to Appendix A and to Granger et al. [57] for further details of these simulations and the classifiers used.

5.5.3 Convergence and negative match tracking

A convergence problem occurs with the above fuzzy ARTMAP algorithm whenever the training subset contains identical patterns that belong to different classes. In the present application, this corresponds to radar pulses in a same resolution cell that belong to different radar types. The problem is aggravated because ARTMAP tends to segment the overlapping parts of classes into several tiny, often minimum-sized prototypes. The consequence is a proliferation of identical prototypes for certain training set patterns.

Consider the following example. Assume that in the first training epoch, fuzzy ARTMAP learns two completely overlapping, minimum-sized prototypes, $\mathbf{w}_{A.1}$ (linked to class A) and $\mathbf{w}_{B.1}$ (linked to class B), for two identical pulse patterns, $\mathbf{a}_1$ and $\mathbf{a}_2$. In a subsequent epoch, $\mathbf{w}_{A.1}$ is initially selected to learn $\mathbf{a}_2$, since $T_{A.1} = T_{B.1}$ and $\mathbf{w}_{A.1}$ was created prior to $\mathbf{w}_{B.1}$ (index $A.1$ is smaller than $B.1$). Since $\mathbf{w}_{A.1}$ is not linked to class B, mismatch raises the vigilance parameter ρ to $(|\mathbf{A}_2 \wedge \mathbf{w}_{A.1}|/M) + \epsilon$, where $|\mathbf{A}_2 \wedge \mathbf{w}_{A.1}| = |\mathbf{A}_2 \wedge \mathbf{w}_{B.1}|$. As a result, $\mathbf{w}_{B.1}$ can no longer pass the vigilance test required to become selected for $\mathbf{a}_2$, and fuzzy ARTMAP creates another minimum-

sized prototype $\mathbf{w}_{B.2} = \mathbf{w}_{B.1}$. From epoch to epoch, the same phenomenon repeats itself, yielding ever more identical prototypes $\mathbf{w}_{B.n} = \mathbf{w}_{B.1}$ for $n = 3, 4, ..., \infty$.

This phenomenon was observed while training fuzzy ARTMAP, ART-EMAP and Gaussian ARTMAP on the radar data set. Results in Table 5.2 were obtained through manual termination by: (1) detecting, from epoch to epoch, the repeated creation of identical prototypes for the same training patterns, (2) pruning non-unique prototypes from memory, and (3) defining the convergence time as the number of epochs leading to the creation of non-duplicate prototypes only.

ARTMAP-IC converged incrementally, by itself, on the radar data. The feature of ARTMAP-IC that circumvents the convergence problem is called *negative match tracking*, denoted MT-, which consists of using a negative ϵ value [28]. In the above example, mismatch raises ρ but $\mathbf{w}_{B.1}$ would still pass the vigilance test. This allows learning of fully overlapping non-unique prototypes for training set patterns that belong to different classes. As shown in Table 5.2, fuzzy ARTMAP with MT- performs as well, to within standard error, as without MT-, yet circumvents the convergence problem. As pointed out by Carpenter and Markuzon [28], MT- is a better algorithmic approximation to the continuous-time version of fuzzy ARTMAP.

5.5.4 Classification of incomplete data

A neural network classifier applied to radar ESM may be subjected to data, either during training or testing, that is incomplete in one or more of the following ways [58]:

1. Limited number of training cases. Collecting and analyzing ESM data from the field of operation to train a neural network can be a costly undertaking. Yet, if the number of training cases is insufficient, the classifier may not achieve good generalization during operations. It is therefore of interest to know how the performance of the classifier declines as the amount of training data is decreased, so that, *e.g.*, more training data may be gathered, if necessary, before the classifier is fielded. The effect on fuzzy ARTMAP performance of reducing the amount of training data from each radar type was characterized by Granger et al. [58]. Overall, fuzzy ARTMAP achieves a high level of accuracy when training with very few pulses from each radar type.

2. Missing components of the input patterns. The information in the different components of the PDW comes from a number of sources. Absence of components in radar ESM processing arises due to sensor limitations and/or delay in deriving parameters (*e.g.*, PRI). This implies that the classifier may encounter partial input patterns. A strategy is presented later in this section that allows fuzzy ARTMAP to effectively process partial input patterns during both training and testing.

3. Missing class labels during training. The task of analyzing radar ESM data collected in the field can be difficult owing to the complexity and lack of control of the environment. Expertise and experience are required for manual separation and labeling of pulse trains transmitted by different emitters. This problem raises the

question of whether the classifier may benefit from training on data with missing class labels. Training on such data is referred to as "semi-supervised learning" [41] or "partially supervised clustering" [8] [107]. To assess the effect on performance of training fuzzy ARTMAP using data with missing class labels, the network was trained in two phases [58]. During the first phase, involving supervised learning, the network was trained as usual until convergence with a fixed amount of labeled training data from each radar type. During the second phase, involving unsupervised learning, the network was presented with a varying percentage of unlabeled data from each radar type until the weights $\mathbf{W}$ converged once again. Using fuzzy ARTMAP without the class prediction (*i.e.,* without Step 4 of the fuzzy ARTMAP algorithm), plus other modifications discussed by Granger et al. [58]), the network associated each unlabeled training pattern with one of the already-existing F_2 category nodes and adjusted the corresponding prototype vectors through slow learning ($0 < \beta < 1$). In all simulations with the radar data, the classification rates observed were never greater than those achieved by simply discarding all unlabeled data [58]. Such an approach is most effective to the extent that clusters of data from different emitters are separable and well clustered, which is not necessarily the case for radar ESM data.

4. Missing classes during training. New radar types (not represented in the training set) may be encountered during operations. When the classifier receives a pattern transmitted by an new radar type, it would be desirable to "flag" the pattern as unfamiliar, rather than make a meaningless guess as to its class label. This may be

implemented by *familiarity discrimination* [26] [106]. The importance in radar ESM of familiarity discrimination during operations is evident: radar emitters can exhibit new modes at any time. The ability to learn unfamiliar classes is anticipated to be just as important. Presently, the task of providing training data to a neural network classifier requires time from an ESM analyst, so it cannot be expected that more than a small fraction of the large amount of available data would be labeled for training. Furthermore, it cannot be assumed that all of the unlabeled training data belongs to one of the radar types identified by the ESM analyst. (Although not explored here, this would involve performing familiarity discrimination during semi-supervised training on unlabeled data.) The modifications presented later in this section enable fuzzy ARTMAP to mitigate performance degradation due to missing class labels, while allowing it to benefit from learning information hidden in unlabeled data about as-yet unfamiliar classes.

Modifications to fuzzy ARTMAP with MT- are now introduced for dealing with missing components of the input pattern (Section 5.5), and missing classes during training (Sections 5.6 and 5.7). Performance obtained with these modifications is assessed via computer simulation using the methodology, evaluation criteria and radar pulse data described in Sections 5.4 and 5.5.1.

Tableau 5.3: Algorithmic modifications to fuzzy ARTMAP required for implementation of the indicator vector (IV) strategy. (Refer to fuzzy ARTMAP equations in Table 5.A.1.)

Algorithmic step	**fuzzy ARTMAP**	**fuzzy ARTMAP with IV**
Prototype selection:		
- choice function	$T_j(\mathbf{A}) = \frac{\lvert\mathbf{w}_j \wedge \mathbf{A}\rvert}{\alpha + \lvert\mathbf{w}_j\rvert}$	$T_j(\mathbf{A}, \delta) = \frac{\lvert\mathbf{w}_j \wedge \mathbf{A} \wedge \delta\rvert}{\alpha + \lvert\mathbf{w}_j \wedge \delta\rvert}$
- vigilance test	$\lvert\mathbf{w}_j \wedge \mathbf{A}\rvert \geq \rho\lvert\mathbf{A}\rvert$	$\lvert\mathbf{w}_j \wedge \mathbf{A} \wedge \delta\rvert \geq \rho\lvert\mathbf{A} \wedge \delta\rvert$
Learning:		
- prototype update	$\mathbf{w}'_J = \beta(\mathbf{A} \wedge \mathbf{w}_J)...$ $... + (1-\beta)\mathbf{w}_J$	$\mathbf{w}'_J = \beta((\mathbf{A} \vee \delta^c) \wedge \mathbf{w}_J)...$ $... + (1-\beta)\mathbf{w}_J$

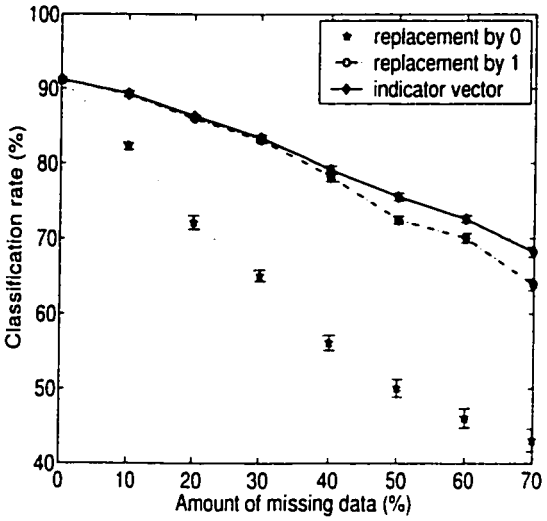
5.5.5 Indicator vector strategy for missing components

A strategy to address missing components in the input patterns consists in using *indicator vectors* [52] [58] [94]. An indicator vector $\delta = (\delta_1, \delta_2, ..., \delta_M)$ informs the fuzzy ARTMAP network about the presence or absence of each component in an input pattern: $\delta_i = 1$ if component i is present, and $\delta_i = 0$ if component i is missing, with $\delta_{i+m} \equiv \delta_i$ for $i = 1, \ldots, m$. Unlike strategies that involve replacement by "0" or by "1" [58], the indicator vector strategy modifies the prototype vectors as well as the input pattern in response to missing components. This approach circumvents a bias in the order of prototype selections by F_2 nodes. The adjustments to the fuzzy ARTMAP algorithm that realized the indicator vector strategy are summarized in Table 5.3.

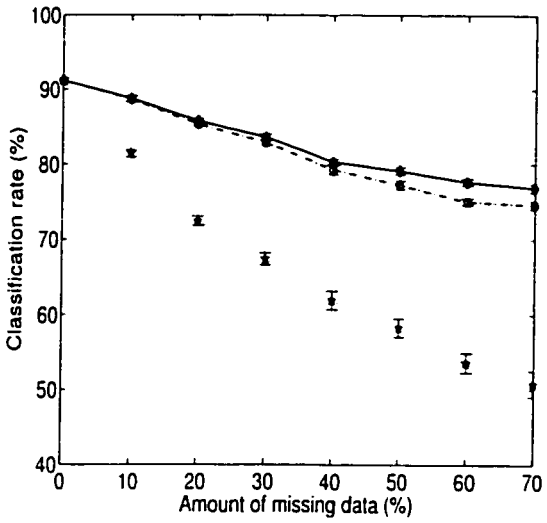
To verify the effectiveness of this strategy, fuzzy ARTMAP with MT- and indicator vector was trained using a randomly-selected 0.5% of the available training set from

each radar type class. (Recall from Section 5.5.2 that the available training subset consists of a random selection of 50% of the pulses from each class. The remaining data forms the test set.) For each class in the training set, a variable percentage, between 0% and 70%, of the components were randomly removed from the patterns and declared to be "missing" (although, if a particular choice of missing components would have left the pattern with *no* components, another random choice was made). The classification rate and compression are shown in Figure 5.5 for the indicator vector strategy, as well as for replacement by "1" and replacement by "0." Also shown are the results obtained when components are removed from the test set. Whether components are missing during training or testing, the indicator vector strategy provides a simple and effective means of handling the absence of components, as its performance degrades gracefully with the percentage of missing data.

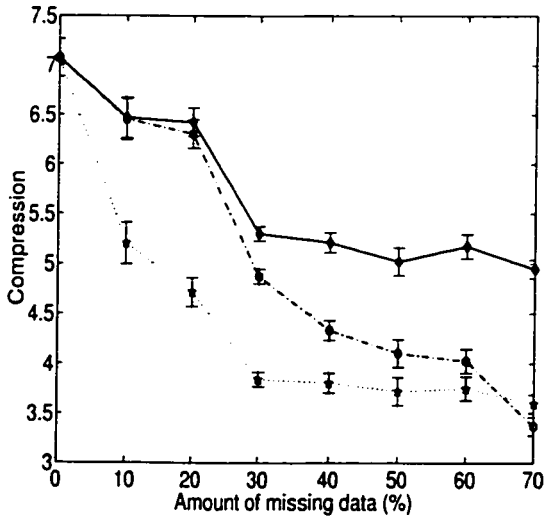
Also noteworthy are the results obtained when there are no missing components. With a randomly-selected 0.5% of the available training data from each class (about 130 pulses total), the classification rate on the test set is 91.4%, compared to 99.6% when all the training data (about 26000 pulses total) are used. The slow decline in accuracy is in part due to the uneven distribution of pulses among radar type classes in the data set.



(a) Accuracy for missing components during testing only.



(b) Accuracy for missing components during training only.



(c) Compression associated with Figure 5.5(b).

Figure 5.5: Average performance of fuzzy ARTMAP with MT-, over 20 simulation trials, using strategies to manage input patterns with missing input components. (Error bars are standard error of the sample mean.)

5.5.6 Familiarity discrimination

An extension to fuzzy ARTMAP that allows for detection of patterns from unfamiliar classes is called ARTMAP-FD. The ARTMAP-FD algorithm has been shown to effectively perform *familiarity discrimination* on simulated radar range profiles [26], and radar pulse data [56]. In addition to the classification rate on patterns from familiar classes, the performance of the classifier can be measured in terms of a *hit rate* (H) — fraction of familiar-class test patterns correctly predicted to belong to one of the familiar classes — and a *false alarm rate* (F) — fraction of unfamiliar-class test patterns incorrectly predicted as familiar by the classifier.

5.5.6.1 ARTMAP-FD algorithm

With complement coding and fast learning ($\beta = 1$), the M-dimensional prototype vector $\mathbf{w}_j = (w_{j1}, w_{j2}, ..., w_{jM})$ associated with F_2 layer category node j of fuzzy ARTMAP defines a hyperrectangle in the m-dimensional space of input pattern components, with edges parallel to the coordinate axes. Such a hyperrectangle records the largest and smallest component value of the training set patterns assigned category j. A pattern that is associated with an F_2 node during testing is "completely familiar" if it falls within the hyperrectangle, and "less-than-completely familiar" to the extent that it falls outside. This notion can be quantified by the *familiarity measure*:

$$\phi(\mathbf{A}) = \frac{T_J(\mathbf{A})}{T_J^{\max}} = \frac{|\ \mathbf{A} \wedge \mathbf{w}_J\ |}{|\ \mathbf{w}_J\ |}\ , \tag{5.4}$$

where $T_J^{\max} = \mid \mathbf{w}_J \mid / (\alpha + \mid \mathbf{w}_J \mid)$. The maximal value of $T_J = T_J^{\max}$ is attained for an input $\mathbf{a}$ if it lies inside the hyperrectangle associated with node J, for then $\mid \mathbf{A} \wedge \mathbf{w}_J \mid = \mid \mathbf{w}_J \mid$. In other words, an input $\mathbf{a}$ that is assigned category J during testing has the maximum familiarity value $\phi(\mathbf{A}) = 1$ if and only if $\mathbf{a}$ lies within hyperrectangle R_J.

ARTMAP-FD is identical to fuzzy ARTMAP during training. During testing, $\phi(\mathbf{A})$ is computed for each input pattern $\mathbf{a}$ after fuzzy ARTMAP has selected node J, and tentatively predicted class $K = k(J)$. An input $\mathbf{a}$ is declared to belong to a *familiar* class if the value of the familiarity measure $\phi(\mathbf{A})$ is greater than a decision threshold γ. In this case, ARTMAP-FD outputs the prediction of class K for $\mathbf{a}$. Otherwise ($\phi(\mathbf{A}) \leq \gamma$), input $\mathbf{a}$ is flagged as belonging to an unfamiliar class, and ARTMAP-FD makes no prediction.

5.5.6.2 Familiarity threshold selection

The choice of a particular familiarity threshold $\gamma = \Gamma$ for use during operations depends upon the relative cost of errors due to *misses* (patterns belonging to familiar classes that the network flags as unfamiliar) and *false alarms*. Since familiarity discrimination involves placing an input into one of two sets, familiar or unfamiliar, the Receiver Operating Characteristic (ROC) formalism [74] can be used to measure the effectiveness of ARTMAP-FD. Optimizing Γ corresponds to choosing a point on the parameterized ROC curve that is close to the upper left-hand corner of the unit

square. This maximizes correct selection of familiar patterns (H) while minimizing incorrect selection of unfamiliar patterns (F). Two methods for predicting a familiarity threshold Γ value are described in [27]. A variant of the "on-line threshold determination" method has been chosen for this work and is now described.

During the first ARTMAP-FD training epoch, every time a category node J wins the competition for a pattern $\mathbf{a}$, fast learning expands R_J just enough to enclose $\mathbf{a}$. Before learning takes place, $\phi(\mathbf{A})$ is be computed, and can have a value less than one. The degree to which $\phi(\mathbf{A})$ is less than 1 reflects the distance from the training pattern to R_J. A training pattern successfully coded by a category node (without reset) is taken to be representative of familiar test-set patterns. The corresponding familiarity measure $\phi(\mathbf{A})$ contributes to the generation of a training hit rate curve, where $H(\gamma)$ equals the fraction of training inputs with $\phi(\mathbf{A}) > \gamma$. In contrast, a reset event during the first training epoch resembles the arrival of an unfamiliar pattern during testing, where reset occurs when a category node J that predicts class K wins the competition for a pattern that actually belongs to a different class k, $k \neq K$. The set of $\phi(\mathbf{A})$ values corresponding to these events are used to generate a training false-alarm rate curve, where $F(\gamma)$ equals the fraction of match-tracking inputs with $\phi(\mathbf{A}) > \gamma$.

After training, the predicted familiarity threshold is given by $\Gamma = \arg\max_\gamma \{H(\gamma) - F(\gamma)\}$. Predictive accuracy is improved by use of a reduced set of $\phi(\mathbf{A})$ values in the training-set ROC curve construction process; namely, training patterns that fall inside

a hyperrectangle ($\phi(\mathbf{A}) = 1$), are not used because these exemplars tend to distort the miss-rate curve. In addition, the *first* incorrect response to a training input is the best predictor of the network's response to an unfamiliar testing input, since sequential search will not be available during testing. Finally, giving more weight to events occurring later in the training process improves accuracy. In this paper, this is accomplished by computing the training curves $H(\gamma)$ and $F(\gamma)$, and predicting the threshold Γ, from the data presented only after the system has created a number of category nodes equal to L (the number of training set classes). Note that this threshold determination method requires storage of all of the training patterns, so as to obtain $H(\gamma)$ and $F(\gamma)$ and thereby predict Γ. For the sake of computational efficiency, it should be possible to approximate $H(\gamma)$ and $F(\gamma)$ from a reduced set of $\phi(\mathbf{A})$ values which would be updated incrementally as new data are obtained.

5.5.7 Learning of unfamiliar classes

With Learning of Unfamiliar Classes (LUC), a classifier continues during the testing phase to adjust its weights via semi-supervised learning. The criteria for familiarity discrimination is also adjusted on-line, and when a test pattern is flagged as unfamiliar, the classifier defines a new class. Subsequent test patterns may be declared by the classifier to be "familiar" and classified as belonging either to classes encountered during training (*i.e.*, training-set classes) *or* to the "newly-minted" classes; or they may be declared to be "unfamiliar," in which case another new class is defined.

The hit rate for an LUC classifier (H^*) can be defined as the fraction of test patterns from training-set classes that are correctly declared to belong to one of the training-set classes. The false alarm rate for an LUC classifier (F^*) is the fraction of unfamiliar-class (*i.e.*, not encountered during the training phase) test patterns not either flagged as unfamiliar nor assigned to a "new" class defined during testing. An additional figure of merit for an LUC classifier is a "purity measure," such as the Rand clustering score [76], which rewards the classifier for learning the right number of unfamiliar classes, and correctly assigning them to unfamiliar patterns.

The algorithm for fuzzy ARTMAP was modified as follows to incorporate LUC. First, in order to focus on the effects of LUC (as opposed to learning with missing class labels), the weights associated with F_2 category nodes allocated during the training phase (training classes) are kept at fixed values during testing. Only "new" F_2 category nodes created during the test phase are allowed to change through slow learning ($0 < \beta < 1$).

In addition, to prevent the generation of an excessive number of new F_2 nodes, patterns that are declared to be unfamiliar are given a "second chance" to be associated with an already-existing new F_2 node before defining a new class. Specifically, a pattern **a** declared unfamiliar by the network is subjected to a vigilance test at each of the new nodes. If it passes the test for one or more of these nodes, it is associated with the node j_{new} among these new nodes which has the strongest activation $T_{j_{\text{new}}}$. If the pattern cannot in this way be associated with an already-existing new node,

and the node J to which the pattern was tentatively assigned is not a new node, then a *coactivation test* is performed. If the activation level T_J of node J is not stronger than the activation level $T_{j_{new}}$ for all of the new nodes j_{new} by a sufficient amount — that is, if $T_J - T_{j_{new}} < \epsilon_{co}$ — then the pattern is associated with the node j_{new} for which $T_J - T_{j_{new}}$ is smallest. If either of these two tests is passed, then no weight adjustment takes place for $\mathbf{w}_{j_{new}}$. Only if neither of these options for association with an already-existing new node succeeds is a new class defined.

When a new class is defined, a new F_2 category node J_{new} is allocated to encode the input ($\mathbf{w}_{J_{new}} \leftarrow \mathbf{A}$), and linked to a new F^{ab} map node K_{new} through $\mathbf{w}^{ab}_{J_{new}}$. Slow semi-supervised learning subsequently adjusts the prototype weights of new F_2 nodes upon their assignment to familiar patterns. Notice that newly-defined classes are interpreted differently: each new F_2 or F^{ab} node represents a fragment of data from an emitter mode that belongs to an unfamiliar radar type.

Finally, fuzzy ARTMAP-LUC extends the on-line method to allow for enhancement of the familiarity threshold Γ from test-set patterns. If pattern $\mathbf{a}$ is declared familiar, $\phi(\mathbf{A})$ is added to the $\{\phi\}$ values used to generate the training set hit rate curve $H^*(\gamma)$; otherwise $\phi(\mathbf{A})$ is added to the $\{\phi\}$ values used to generate the false alarm rate curve $F^*(\gamma)$. The threshold $\Gamma^* = \arg\max_\gamma\{H^*(\gamma) - F^*(\gamma)\}$ is recomputed following each input pattern assignment. For faster execution, it is possible to adjusts Γ every time a pattern is assigned a new F_2 node; that is, when $\mathbf{W}$ is modified.

Tableau 5.4: Average results, over the same 20 simulation trials, using ARTMAP-FD with and without LUC. (Numbers in parentheses are the standard error of the sample mean.)

Evaluation criteria	**fuzzy ARTMAP with FD**	**fuzzy ARTMAP with FD and LUC**
Hit rate	99.60% (0.07%)	99.63% (0.05%)
False alarm rate	14.33% (8.89%)	7.46% (4.25%)
Classification rate	99.51% (0.05%)	99.49% (0.05%)
Memory	806.2 (40.1)	931.1 (41.4)
Rand clustering score	N/A	0.7640 (0.0597)

5.5.8 Simulations with familiarity discrimination and unfamiliar classes

In computer simulations, 13 out of the 15 radar types were declared to be familiar (thus training classes). Familiar class selection was performed at random, with the restriction that an insufficient number of unfamiliar-class data patterns (less than a thousand) was not allowed. A randomly-selected 50% of the data from each of the 13 training classes were presented to the network during the training phase. The familiarity threshold Γ was determined during the training phase using the on-line method described in Section 5.6.2. Patterns remaining from the 13 training classes, plus all the patterns from the 2 unfamiliar radar type classes formed the test set.

Average results obtained for fuzzy ARTMAP with FD (pure ARTMAP-FD) and for fuzzy ARTMAP with FD and LUC are shown in Table 5.4. Overall results indicate that fuzzy ARTMAP with FD has a high hit rate (99.60%), but only marginal

performance with regards to avoiding false alarms: 14.33% of patterns belonging to unfamiliar classes are mistakenly assigned familiar training classes. This is a consequence of the overlap, scattering, and uneven mixture of pulses from different radar types. By defining new classes and assigning them to unfamiliar-class patterns, LUC reduces this false alarm rate by about half, to 7.46%, without loss of accuracy.

On average, fuzzy ARTMAP with FD and LUC requires an additional 124.9 registers to store the prototype vectors, and thus about 21 new F_2 (plus F^{ab}) nodes to represent the different emitter modes from the 2 unfamiliar classes. Despite the creation of these additional nodes, the network's predictive accuracy on data from familiar classes is not significantly degraded. The moderate Rand score, 0.7640, indicates the ability of LUC to recover some of the true cluster structure in data from the 2 unfamiliar classes.

5.6 Pattern clustering

The objective of pattern clustering in the What-and-Where recognition architecture shown in Figure 5.2 is to group patterns from the Where data stream into tracks. Impinging signals contain information about emitter status, which may change in time. Desirable features for such on-line clustering include the ability to initialize new tracks whenever new emitters are detected, to adjust tracks in response to emitter maneuvers, and to delete tracks as emitters leave or stop transmitting.

Several techniques can perform on-line sequential clustering. For instance, adap-

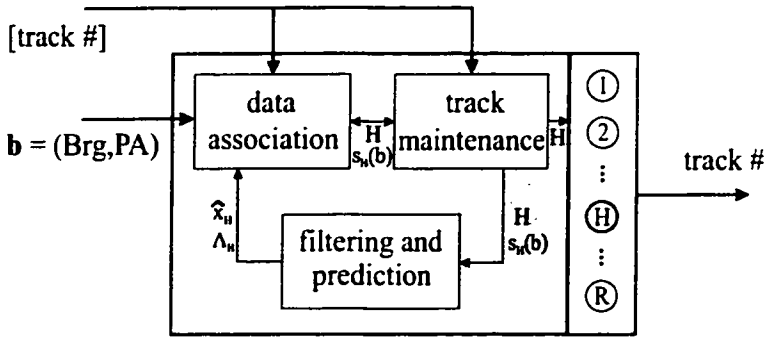


Figure 5.6: Pattern clustering system based on nearest-neighbor matching and Kalman filtering.

tive vector quantization [64] algorithms may be used if parameter values vary slowly. More sophisticated tracking algorithms [7] [12] are needed to update tracks for parameters that exhibit rapid linear or nonlinear variations. In this section, on-line clustering of Brg and PA parameters is implemented by combining nearest-neighbor matching with linear Kalman filtering. A breakdown of the recursive processing required for on-line clustering is given in Figure 5.6. The three basic functions — data association, track maintenance, and filtering and prediction — are now examined.

5.6.1 Data association

An incoming pattern **b** from the Where data stream is initially considered for association with existing tracks. This association involves computing a *match* $s_h(\mathbf{b})$ between input **b** and the *next predicted* position of every track ($h = 1, 2, ..., R$) in the Where environment. Assume that track positions are drawn independently and identically from a mixture of Gaussian distributions, in which one distribution is associated with each track. Then, the match can be taken to be a probability, and

written:

$$s_h(\mathbf{b}) = \frac{1}{(2\pi)^{\frac{M}{2}}|\Lambda_h|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(\mathbf{b} - \hat{\mathbf{x}}_h)^T \Lambda_h^{-1}(\mathbf{b} - \hat{\mathbf{x}}_h)\} \tag{5.5}$$

where M is the number of dimensions in the Where space, and $\hat{\mathbf{x}}_h$ and Λ_h are, respectively, the predicted position and covariance matrix for track h. Assuming that all tracks have equal prior probabilities, the track $h = H$ that maximizes Equation 5.5 is associated with $\mathbf{b}$:

$$H = \arg\max_h \{s_h(\mathbf{b}) : h = 1, 2, ..., R\}. \tag{5.6}$$

Kalman filtering [7] [12] is employed to predict the next position $\hat{\mathbf{x}}_h$, and covariance matrix Λ_h of each track h. Recall from Section 5.3 that the usual clustering is bypassed when $\mathbf{b}$ corresponds to a PDW that has been assigned a previously-established track through TOA deinterleaving. In this case, $\mathbf{b}$ retains its track, and does not perform data association, nor track maintenance. Kalman filtering and prediction are however still performed in order to sustain a consistent description of all active emitters in the environment.

5.6.2 Track maintenance

Once associated with pattern $\mathbf{b}$, track H undergoes two tests. In the first, the match $s_H(\mathbf{b})$ is compared to a threshold δ_c that regulates the creation of new tracks, $\delta_c \in [0, 1]$. If $s_H(\mathbf{b}) \geq \delta_c$, then the test is passed. In the second test, the cumulative

average of match value S_H is computed:

$$S_H = \frac{\sum s_H(\mathbf{b})}{Q_H} , \tag{5.7}$$

where Q_H is the number of patterns to which track H was assigned. S_H is compared to another threshold δ_d, that regulates the overall quality of existing tracks, $\delta_d \in [\delta_c, 1]$. If $S_H \geq \delta_d$, then the test is passed. If both tests are passed, then track H is *assigned* to $\mathbf{b}$.

If either test is failed, then a new track is initiated for pattern $\mathbf{b}$. When a new track H is initiated, $s_H(\mathbf{b}) = 1$, $\hat{\mathbf{x}}_H$ is set equal to $\mathbf{b}$ and Λ_H is set equal to $\sigma^2 I_M$, where I_M is the identity matrix, and $\sigma = (\sigma_1, \sigma_2, ..., \sigma_M)$ represents the resolution of Where parameter measurements. Furthermore, if the second test is failed (*i.e.*, $S_H < \delta_d$), then the previously-established track H is deleted.

After assignment of H to either an existing or newly-initiated track, any track h that has not been assigned to an input pattern for a time greater than an ageout parameter $\tau > 0$ is deleted. That is, a track is deleted if:

$$\text{TOA}(\mathbf{b}) - \text{TOA}_h > \tau , \tag{5.8}$$

where TOA_h is the time at which track h was last assigned to an input. Deleting a track frees up resources, and reduces the chances of future miss-assignments. The track number H is the output from track maintenance.

A high quality track is one that is assigned pulses transmitted, to a large extent, from the same emitter. To ensure high quality tracks, new tracks are initiated rapidly by setting δ_c close to 1, whereas poor quality tracks are deleted more slowly (as S_H progressively declines). This reduces ambiguity during track assignment, but may lead to the initiation of a greater number of tracks, and thus predictions K^e based on the accumulation of short sequences of pulses.

5.6.3 Kalman filtering and prediction

Kalman filtering and prediction is implemented with a bank of standard Kalman filters, one per track. Every track h is associated with a Kalman filter, and is represented by a unimodal Gaussian distribution. Upon the assignment of a track H to $\mathbf{b}$, the filter of H is employed to predict its next position and covariance matrix.

5.7 Sequential evidence accumulation

Sequential evidence accumulation exploits Where information by combining the responses of pattern clustering with neural network classification. In short, the classifier's responses are accumulated according to track, thus offering predictions from multiple views of an emitter.

5.7.1 Fusion of What and Where information

As mentioned in Section 5.6, clustering produces a track number $h = H$ for each pattern $\mathbf{b}$ from the Where data stream. The track number indicates the specific emitter assigned to $\mathbf{b}$.

Sequential evidence accumulation is implemented by means of identical evidence accumulation fields $F_1^e, F_2^e, ..., F_R^e$, where each field F_h^e is connected to a track h, and replicates the neural network classifier's output field, that is, contains L nodes, one per radar type class. The classifier's output nodes are linked to their respective nodes in all fields F_h^e, $h = 1, 2, ..., R$. Each field F_h^e incorporates a short-term memory capable of accumulating its input patterns. The memory for F_h^e is characterized by a field accumulation pattern $\mathbf{T}_h^e = (T_{h1}^e, T_{h2}^e, ..., T_{hL}^e)$.

Upon initiation of a track h, $\mathbf{T}_h^e$ is set equal to $\mathbf{0}$. When track $h = H$ is assigned to pattern $\mathbf{b}$, F_H^e becomes active. The activity pattern $\mathbf{y}^{ab}$ output by the classifier accumulates onto F_H^e according to:

$$(\mathbf{T}_H^e)' = \mathbf{T}_H^e + \mathbf{y}^{ab} \ . \tag{5.9}$$

Accumulation of activity patterns in F_h^e continues until track h is deleted.

For a given input PDW, the activity pattern $\mathbf{y}^e$ output from evidence accumulation

is equal to $\mathbf{T}^e_H$. The radar type is predicted to be:

$$K^e = \arg\max_{k^e}\{T^e_{Hk^e} : k^e = 1, 2, ..., L\} \tag{5.10}$$

Besides discrete prediction, evidence accumulation fields can be used to feed an emitter table in the signal separator of Figure 5.1. The fields can also describe multiple radar types associated with same Where features, for instance the same location, which can assist in linking emitters to platforms.

5.7.2 Prediction from multiple views

Sequential evidence accumulation may improve the overall classification rate of the recognition system, since the accumulated prediction K^e is tolerant to errors committed by the neural network classifier (prediction K). The concept of predicting classes from multiple "views" of a source, that are accumulated through time, has been successfully developed in a number of neural network architectures [6] [14] [25]. In the present case, information on the origin of input patterns is provided through clustering of the Where data. The effectiveness of evidence accumulation here depends on the quality of the tracks that are computed as in Section 5.6.

5.7.3 Simulations with evidence accumulation

This section summarizes simulations of how the entire What-and-Where system performs on the radar pulse data set. Software was written in the Matlab language

to implement the neural network classification, clustering and evidence accumulation modules, as well as their integration. The parameter settings used for clustering were $\delta_c = 0.98$, $\delta_d = \delta_c + 0.01$, and $\tau = 10$ ms. The neural network classifier was fuzzy ARTMAP with negative match tracking (MT−), indicator vector strategy (IVS), familiarity discrimination (FD) and learning of unfamiliar classes (LUC), as described in Section 5.5. The vigilance and coactivation parameters used for associating unfamiliar patterns with already-existing new nodes in LUC were $\rho = 0.8$ and $\epsilon_{co} = 0.05$. A learning rate of $\beta = 0.5$ was used for semi-supervised adjustment of prototype weights for new nodes.

Patterns presented to the fuzzy ARTMAP classifier contained 3 What parameters — namely PRI, PW and RF — whereas patterns presented to the clustering module contained 2 Where parameters — namely Brg and PA. The system normally receives the track number and PRI value for PDWs to which a track was assigned by the TOA deinterleaver. TOA deinterleaving is a difficult task due to the agility (jitter, stagger, etc.) of the PRI in modern radar systems. For simulation purposes, it was assumed that TOA deinterleaving can, during on-line operation, be reliably achieved only for constant PRIs. Approximately 30,000 pulses from the data set belong to constant PRI emitters, and the rest (about 22,000 pulses) belong to complex PRI emitters.

During fuzzy ARTMAP training, all What patterns had a PRI component. For emitters with complex PRI patterns, the PRI values were computed as in Section 5.5. During testing, the PRI components were declared missing from test-set patterns that

belonged to emitters with complex PRI patterns. In addition, a randomly-selected 10% of the What components from each emitter mode were also declared missing. The IVS was used by fuzzy ARTMAP to deal with missing components.

For emitters with constant PRI, the PRI values used for both training and testing were mean PRI values, estimated using a moving average that accounts for dropped pulses. Assume that the TOA deinterleaver has correctly sorted the n_k pulses belonging to emitter mode k, and computed the $\mathrm{PPI}_k(i)$ and $q_k(i)$ values for $i = 2, 3, ..., n_k$, where $q_k(i)$ is the closest multiple of the nominal PRI_k value (taken from a PRI table for emitter mode k) to the $\mathrm{PPI}_k(i)$ value; namely $q_k(i) = \arg\min_q\{q : |q \cdot \mathrm{PRI}_k - \mathrm{PPI}_k(i)| : q = 1, 2, 3 \ldots\}$. The moving average is:

$$\overline{\mathrm{PRI}}_k(i) = \frac{1}{\Delta} \sum_{m=i-(\Delta+1)}^{i} \frac{\mathrm{PPI}_k(m)}{q_k(m)} , \tag{5.11}$$

where Δ is the number of pulses in the moving average window. Observing the PPI of pulses inside a window defined by the last 5 consecutive PRIs is consistent with the clustering parameter $\tau = 10$ms, which corresponds to 5 times the longest nominal PRI value that is expected. If the PPI elapsed for a pulse goes beyond this window, the pulse is assumed to belong to the next burst of pulses from the emitter.

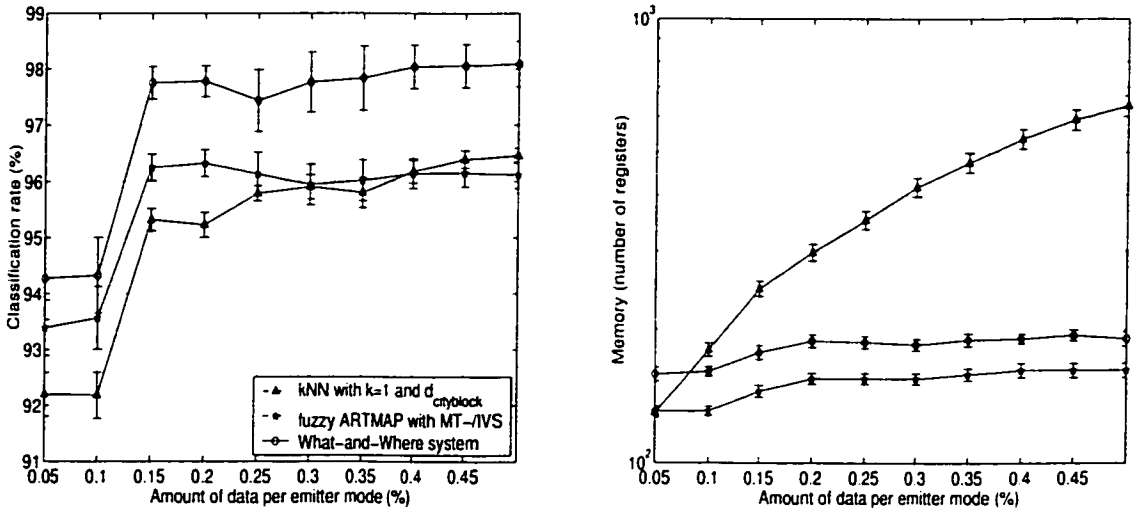
To account for the chronological evolution of the environment, the training set was formed by selecting the first 0.5% of data encountered from each emitter mode in TOA order. Furthermore, this training data was taken only from the emitter modes corresponding to 13 of the 15 radar types selected at random, and declared to be

familiar. In practice, this could correspond to training the neural network classifier on the first few bursts of pulses intercepted from each familiar-class emitter. After training, all the remaining data from the 13 familiar classes (99.5% from each emitter mode), plus all the patterns from the 2 unfamiliar classes, were presented to the recognition system in TOA order for prediction. To deal with missing classes during training, FD and LUC (as described in Section 5.5) were used. When applied with the IVS, the familiarity measure of Equation 5.4 becomes:

$$\phi(\mathbf{A}) = \frac{|\ \mathbf{A} \wedge \mathbf{w}_J \wedge \delta\ |}{|\ \mathbf{w}_J \wedge \delta\ |}\ . \tag{5.12}$$

When using fuzzy ARTMAP with FD alone, evidence accumulation of responses $\mathbf{y}^{ab}$ onto field F_H^e occurs only if the input $\mathbf{a}$ is declared familiar. Using fuzzy ARTMAP with FD and LUC, evidence accumulation always occurs since new classes are defined for unfamiliar-class patterns. To support the ability to accumulate unfamiliar-class patterns, whenever a new class is defined, a new node is initialized within each evidence accumulation field F_h^e, $h = 1, 2, ..., R$, as well as in the F_2 and F^{ab} layers.

The performance of the What-and-Where system as a function of the amount of data used for training is summarized in Figure 5.7. The amount of data is a randomly-selected percentage of patterns from each emitter mode in the training subset. This percentage was varied between 10% and 100% of the training data (between 0.05% and 0.5% of the entire data set). Results obtained with the fuzzy ARTMAP with MT- and IVS, and with kNN classifiers, are included for comparison. When components



(a) Classification rate on familiar-class patterns.

(b) Memory requirements during training.

Figure 5.7: Average performance, over 20 simulation trials, of the What-and-Where system. (Error bars are standard error of the sample mean.)

are missing, test-set patterns are classified with kNN on the basis of parameters that are present.

Figure 5.7(a) indicates that the What-and-Where system, and thus the fusion of What and Where information, significantly improves the classification rate of fuzzy ARTMAP with MT-/IVS on familiar-class patterns by about 2%. The system achieves a classification rate of about 98% with a training set consisting of as little as 0.15% of the whole data set, or about 80 pulses. This level of performance is attained along with a capability for detecting and learning patterns from unfamiliar classes. When trained on just 0.5% of the data per familiar class, the recognition system yields an average hit rate of $H^* = 96.9\%$ and a false alarm rate of $F^* = 12.9\%$ on the test subset.

The notion that additional training examples beyond a certain point become "redundant" is borne out of Figure 5.7(b), which shows memory growing as the number of training patterns is increased. The figure also shows memory cost due to defining new classes during testing. When trained on 0.5% of the data per familiar-class emitter modes, the system creates on average 5 new nodes on F_2, F^{ab}, and all F_h^e layers. The Rand clustering quality score of the unfamiliar-class patterns assigned to these nodes is on average 0.73. This is a slight improvement to the score of 0.70 obtained with fuzzy ARTMAP with MT-/FD/LUC.

5.8 Conclusions

A novel What-and-Where architecture has been proposed for recognition and tracking of radar emitters for Electronic Support Measures (ESM). This architecture combines a neural network classifier, an on-line clustering algorithm, and an evidence accumulation module. Once trained on samples of data gathered in the field of operation, the neural network classifier can predict the radar type of intercepted pulses based on their What parameters. Meanwhile, the clustering algorithm separates these pulses according to emitter based on their Where parameters. The evidence accumulation module permits fusion of the classifier's What responses with the clustering algorithm's Where estimates, and thus allows prediction of the radar type from classifications along an entire emitter trajectory. For proof-of-concept computer simulations using a radar data set, a particular realization of the recognition

system has been considered. It consists in using a variant of fuzzy ARTMAP for classification, and nearest-neighbor matching with a bank of Kalman filters for on-line clustering.

Simulations results show that fuzzy ARTMAP with negative match tracking, MT-, a core concept of the ARTMAP-IC algorithm [28], consistently delivers a high level of accuracy and compression on the radar pulse data set, even when the amount of training data is limited. Compared to several other ARTMAP variants, as well as the reference RBF and kNN classifiers, it yields one of the best classification rates, yet requires among the least resources (shortest convergence time and least storage for prototypes) and computational complexity for on-line predictions. The MT- feature allows fuzzy ARTMAP to converge naturally, by circumventing a node proliferation problem that can arise when identical or nearly-identical input patterns in the training data correspond to different classes. Modifications of fuzzy ARTMAP with MT- have also been introduced for dealing with missing components of the input pattern, and missing classes during training. The indicator vector strategy (IVS) provides an effective means of processing partial input patterns, whenever components of the data set are missing during training or testing. Familiarity discrimination (FD) allows fuzzy ARTMAP to detect patterns belonging to unfamiliar classes during training and testing, and enables learning of unfamiliar classes (LUC) to take place during testing.

Computer simulations that test the entire What-and-Where system improve accuracy significantly over those obtained with fuzzy ARTMAP with MT- and IVS, and

with kNN, while also detecting and learning patterns from unfamiliar classes. These results support the general approach of integrating What and Where information via evidence accumulation, and offer promise for application in autonomous ESM systems which may be subjected to complex, incomplete, and overlapping radar data.

5.9 Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency and the Office of Naval Research ONR N00014-95-1-0409 (S. G. and M. A. R.), the National Science Foundation NSF IRI-97-20333 (S. G.), the Natural Sciences and Engineering Research Council of Canada (E. G.), and the Office of Naval Research ONR N00014-95-1-0657 (S. G.).

5.A Appendix A

5.A.1 ARTMAP neural network classifiers

ART-EMAP (Stage 1) and ARTMAP-IC extend fuzzy ARTMAP to produce a distributed activation of coded F_2 nodes during testing. Furthermore, ARTMAP-IC biases distributed test set predictions according to the number of times F_2 nodes are assigned to training set patterns. It also uses negative match tracking (*i.e.*, negative ϵ values), to address the problem of inconsistent cases, whereby identical training set patterns correspond to different classes labels. After an incorrect prediction during

training, the vigilance parameter is raised just enough to induce a search for another internal category node J, and then lowered by a small amount $\epsilon > 0$.

Gaussian ARTMAP represents each category j as a separable Gaussian density function, defined by its mean $\mu_j = (\mu_{j1}, \mu_{j2}, ..., \mu_{jM})$ and its standard deviation $\sigma_j = (\sigma_{j1}, \sigma_{j2}, ..., \sigma_{jM})$ vectors. During training, the number of committed F_2 nodes, N_c grows. All committed F_2 nodes that pass the vigilance test for pattern **a** activate, and distribute a pattern of activity $\mathbf{y} = (y_1, y_2, ..., y_{N_c})$. Match tracking and learning are performed according to the relative activation over the "ensemble" E_K of F_2 nodes linked to the predicted F^{ab} node K. The relative activation over E_K is defined by the distributed pattern $\mathbf{y}^* = (y_1^*, y_2^*, ..., y_{N_c}^*)$, where $y_j^* = y_j / \sum_{l \in E_K} y_l$ only if $j \in E_K$, and $y_j^* = 0$ otherwise. Finally, the learning rate of category j is gradually decreased according to y_j^*/n_j. Table 5.5 highlights the main algorithmic differences between fuzzy ARTMAP and Gaussian ARTMAP.

5.A.2 Distributing and biasing test set activation

Simulation trials showed that the Q-max rule [28] for distributing F_2 layer activation in the ART-EMAP (Stage 1) and ARTMAP-IC classifiers gives better results than either power or threshold rules [25] with the radar data. The following choice of Q was found to give good results: $Q = \min\{\lceil N_c/2L \rceil, 2L\}$, where L is the number of classes (15 in this study) and N_c is the number of committed F_2 nodes. In particular, bounding Q to be below $2L$ reduces performance fluctuations.

Regardless of distributed activation, fuzzy ARTMAP performs better than its two extensions, ART-EMAP (Stage 1) and ARTMAP-IC, on the radar data; see Table 5.5. Distributed activation of the test patterns (in ART-EMAP, for example) yields more prediction errors because radar types in the data set can be dispersed, fragmented, and overlap one another. These data properties work against class predictions that are based on the distribution of strongly activated F_2 nodes among radar types, rather than on the most active F_2 node. Indeed, an F^{ab} class node k that receives a very strong activation from one F_2 node may have weaker overall activation than an F^{ab} class node h that receives a moderately strong activation from several F_2 nodes. Perhaps this explains why kNN gives its best performance for $k = 1$, and degrades slowly as k is increased. For example, its classification rate is 0.992 for $k = 9$ and $d_{\text{cityblock}}$.

Gaussian ARTMAP (Table 5.5) involves training and testing with a distributed pattern of activity. F_2 layer activation is distributed among nodes that pass the vigilance test. When training, each category $j \in E_K$ learns according to its relative activation for **a**. The learning rate of each category j is also gradually decreased as a function of n_j. F_2 nodes learn a Gaussian mixture model of the input space. Although computationally intensive, this learning strategy allows Gaussian ARTMAP to achieve high classification rates.

Tableau 5.5: Distinctive equations used by the ARTMAP neural networks in the comparison. ART-EMAP (Stage 1), ARTMAP-IC are extensions of fuzzy ARTMAP. With Gaussian ARTMAP, γ is the initial standard deviation assigned to newly-committed $F2$ nodes, and the scalar n_j accumulates the amount of relative activation obtained by F_2 node j on training set patterns.

Algorithmic step	**ARTMAP classifier**	
	fuzzy ARTMAP	Gaussian ARTMAP
1. Initialization:	$\alpha > 0$, $\beta \in [0,1]$	$\mu_J = \mathbf{A}$, $\sigma_{Ji} = \gamma$ $(\gamma > 0)$, $w_{JK}^{ab} = 1$, $n_J = 1$
2. Input pattern coding:	$\mathbf{A} = (\mathbf{a}, \mathbf{a}^c)$	$\mathbf{A} = \mathbf{a}$
3. Prototype selection:		
– choice function	$T_j(\mathbf{A}) = \lvert\mathbf{A} \wedge \mathbf{w}_j\rvert / (\alpha + \lvert\mathbf{w}_j\rvert)$	$g_j(\mathbf{A}) = \begin{cases} \frac{n_j}{\prod_{i=1}^{M} \sigma_{ji}} G_j(\mathbf{A}) & \text{if } G_j(\mathbf{A}) > \rho \\ 0 & \text{otherwise} \end{cases}$
– vigilance test	$\lvert\mathbf{A} \wedge \mathbf{w}_J\rvert \geq \rho m$	$G_j(\mathbf{A}) = \exp\left\{-\frac{1}{2}\sum_{i=1}^{M} \frac{(A_i - \mu_{ji})^2}{\sigma_{ji}^2}\right\} > \rho$
– $F2$ activation	$y_j = 1$ only if $j = J$ (winning node)	$y_j = g_j / (0.01 + \sum_{l=1}^{N_c} g_l)$
4. Class prediction:		
– match tracking $(\epsilon = 0^+)$	$\rho = (\lvert\mathbf{A} \wedge \mathbf{w}_J\rvert / m) + \epsilon$	$\rho = \exp\left\{-\frac{1}{2}\sum_{j \in E_K} y_j^* \sum_{i=1}^{M} \frac{(A_i - \mu_{ji})^2}{\sigma_{ji}^2}\right\} + \epsilon$
5. Learning:		$n_j' = n_j + y_j^*$
– prototype update	$\mathbf{w}_J' = \beta(\mathbf{A} \wedge \mathbf{w}_J) + (1 - \beta)\mathbf{w}_J$	$\mu_j' = \frac{y_j^*}{n_j}\mathbf{A} + (1 - \frac{y_j^*}{n_j})\mu_j$
– standard deviation update	N/A	$\sigma_j' = \sqrt{\frac{y_j^*}{n_j}(\mathbf{A} - \mu_j)^2 + (1 - \frac{y_j^*}{n_j})\sigma_j^2}$

ARTMAP-IC and Gaussian ARTMAP accumulate weighting factors that depend on the quantity of training subset patterns assigned to each F_2 node. This frequency information is used to bias predictions towards classes assigned the most training patterns. This is a problem in radar ESM since some critical radars transmit very few pulses, while others transmit hundreds of thousands of pulses per second. Biasing prototype choices according to patterns in the TOA of pulses would, for instance, be more appropriate.

Fuzzy ARTMAP with MT- breaks ties (during prototype selection) by choosing the F_2 node with the smallest index. Even though it is not necessarily appropriate in our context, it may be useful on other data sets for fuzzy ARTMAP with MT- to use instance-weighted outputs only in the case where winning nodes are "inconsistent-case siblings," since in this case a basis on which to choose one of the winning nodes over another may be the frequency with which they were winning nodes during training. When using fuzzy ARTMAP with "limited instance counting," instances are still counted for all F_2 nodes. However, it distributes activation weighted by the instance counts if and only if nodes J are a set that code for the same test pattern but map to different classes. Limited IC does not harm accuracy on the radar data set [57].

5.A.3 Prototype representations

Given the quantization of parameter measurements, intercepted radar pulses fall into resolution cells. The measurement uncertainty of the three parameters used (RF,

PRI and PW) is uncorrelated, therefore the radar type definitions are essentially rectangular. ART-EMAP, ARTMAP-IC and fuzzy ARTMAP use hyperrectangles to represent prototypes in the input space, and appear to be a better match for this type of data. Gaussian ARTMAP and RBF, on the other hand, represent prototypes with Gaussian density functions. This results in substantial fragmentation of radar type classes, and in low compression for these two classifiers.

5.10 Synthèse et impact des résultats:

Dans l'article précédent, un RNA à fusion "what-and-where" a été proposé pour l'identification des types de radar associés aux impulsions interceptées. Ce RNA fusionne deux sources d'information d'après les séquences d'impulsions. Les paramètres de type "what" dans les PDWs servent à classifier les impulsions radars selon leur type, tandis que les paramètres "where" servent à séparer les impulsions correspondant aux émetteurs actifs. Cette séparation permet d'accumuler les réponses du classificateur pour chaque émetteur, et donc de prédire le type de radar d'un émetteur actif selon plusieurs réponses.

Des simulations ont été effectuées pour une mise-en-oeuvre particulière du RNA à fusion "what-and-where" et pour un ensemble de données radars. Avec cette mise-en-oeuvre, le RNA classificateur est réalisé par une variante du fuzzy ARTMAP, tandis que le sous-système de catégorisation est réalisé par un algorithme qui exécute l'association du type plus-proche-voisin et le filtrage de Kalman.

La performance du RNA fuzzy ARTMAP a été comparée avec celle de quelques autres variantes de ARTMAP, ainsi qu'avec le kNN et le RBF. Les résultats de simulations ont révélé que le fuzzy ARTMAP permet d'obtenir un des meilleurs taux de classification parmi ces approches. De plus, fuzzy ARTMAP est parmi les classificateurs qui sont les plus efficaces en termes du temps de convergence, de la mémoire pour stocker les prototypes et de la complexité des calculs. Son niveau élevé de performance a été obtenu avec les données radars, même quand la taille du lot d'entrainement est

limitée. Lors de simulations, l'utilisation de la partie MT- (pour "negative match tracking") de ARTMAP-IC [28] a facilité la convergence du RNA fuzzy ARTMAP quand le lot d'entrainement contenait des impulsions dont les paramètres "what" sont quasi-identiques, mais qui sont liées à différents types de radar. Les simulations ont aussi montré que l'utilisation des *vecteurs indicateurs* est une stratégie très efficace pour traiter des PDWs quand des paramètres "what" sont absents. Finalement, une extension du ARTMAP-FD [26] a permi non seulement de détecter des impulsions émises par des types de radar non-familiers, mais d'apprendre les classes correspondantes en temps réel. Il est à noter que ce dernier RNA peut aussi servir au triage par cellules avec information *a priori* (sur les émetteurs qu'on est susceptible de rencontrer) [47].

Les résultats de simulations pour le RNA à fusion "what-and-where" en entier ont démontré une amélioration significative des performances par rapport au fuzzy ARTMAP modifié seul. Le système obtient un taux de classification d'environ 98% avec un lot d'entrainement formé d'aussi peu que 0.15% de l'ensemble au complet des données. On peut anticiper des performances encore meilleures si le système proposé exploitait des paramètres "where" plus puissants, comme le MOP.

Il existe plusieurs aspects qui seraient intéressant de développer pour le RNA à fusion "what-and-where." En ce qui concerne la détection de nouveauté, il serait souhaitable de comparer les performance de l'approche qui est proposée avec plusieurs autres méthodes comme le "Near-Enough-Neighbor" [56]. La performance de cette

composante du RNA devient de moins en moins bonne avec le chevauchement des classes. (Dans ce cas, les erreurs de classification donnent souvent lieu à des valeurs de nouveauté très élevées.) Il serait souhaitable d'améliorer la mesure qui détecte les nouveautés pour tenir compte du chevauchement entre les classes qui sont apprises.

Une considération importante pour les systèmes de MSE radar est la classification de signaux qui sont émis par des émetteurs inconnus. Le RNA à fusion "what-and-where" permet de raffiner les connaissances du réseau dans le champs face à ces émetteurs. Malgré des résultats prometteurs, l'approche proposée pour modifier les poids synaptiques et le seuil pour la détection de nouveauté mérite plus de recherche. En fait, il serait pertinent d'entreprendre une étude approfondie des différents algorithmes d'apprentissage qui permettent d'apprendre de nouvelles informations de façon incrémentale.

Finalement, il serait utile d'observer les performances de ce RNA pour d'autres données radars. De plus, afin d'isoler les cas problématiques, il serait utile d'observer la distribution des erreurs de classification dans le temps et selon les différentes classes.

Conclusion et discussion générale

Discussion générale

Les systèmes de Mesures de Soutien Électronique (MSE) radars sont employés dans un contexte de guerre électronique, pour détecter et identifier les émetteurs dans un environement électromagnétique. Dans les environements modernes, ces systèmes peuvent s'attendre à traiter une densité et une complexité croissante de signaux radars. On cherche alors des technologies plus rapides, plus précises et plus fiables pour les concevoir.

Les travaux abordés dans cette thèse s'inscrivent dans le cadre d'une étude sur le potentiel des techniques de réseaux de neurones artificiels (RNA) pour effectuer deux fonctions critiques en MSE radar: le triage métrique d'impulsions radar et l'identification des types d'émetteurs radars. Cet ouvrage comporte quatres contributions qui sont organisées en deux volets. Le premier volet est lié aux trois premières contributions et traite des RNA pour le triage métrique rapide d'impulsions radars. Le deuxième volet est lié à la dernière contribution et traite des RNA pour l'identification

des types de radar.

En MSE radar, il est important de détecter les menaces aussi rapidement que possible. Le temps de réponse du triage métrique est alors un élément aussi critique que la qualité des catégorisations. Dans la première contribution, quatre RNA auto-organisateurs de type apprentissage compétitif ont été comparés en termes de leur qualité de catégorisation et de leur effort de calcul. Les résultats des simulations avec un ensemble de données radar et des estimations de complexité ont permi de conclure que deux de ces réseaux, le Self-Organizing Feature Mapping (SOFM) et le fuzzy Adaptive Resonance Theory (ART), sont d'excellents candidats pour le triage rapide de séquences d'impulsions, sans connaissance a priori. Le SOFM produit des catégorisations très précises, mais peut exiger un long délai pour converger et est caractérisé par une complexité de calcul élevée. Alors, celui-ci serait plus approprié pour des systèmes de surveillance, d'intelligence et de ciblage à longue portée, où la précision est plus critique. En revanche, fuzzy ART produit des catégorisations un peu moins précises que le Self-Organizing Feature Mapping, mais il possède un grand potentiel pour le traitement à débit élevé. Celui-ci serait plutôt désirable pour des systèmes d'alerte contre les menaces, où le temps de réaction et/ou la compacité du système est plus critique. Les contributions deux et trois élaborent des concepts liés au triage métrique rapide avec le RNA fuzzy ART.

Dans la seconde contribution, la mise en oeuvre VLSI du RNA fuzzy ART a été étudiée pour des applications de triage rapide. L'algorithme fuzzy ART a été refor-

mulé pour faciliter sa mise en oeuvre avec la technologie VLSI numérique. Ensuite, une architecture de système VLSI dédiée a été proposée pour partitionner la fonctionalité de cet algorithme sur plusieurs ASIC. Cette architecture est modulaire et cascadable selon les besoins de l'application. Elle comprend un comparateur global, ainsi qu'un ensemble de modules élémentaires identiques, qui permettent chacun d'émuler un certain nombre de neurones. Le système a été ciblé au triage métrique d'impulsions radars à débit élevé en MSE. Un modèle d'estimation AT pour cette architecture a permi d'isoler un ensemble de configurations qui peuvent accomoder plus de 250 catégories et traiter bien au-delà de 10^5 patrons par seconde, tout en occupant une surface acceptable. Une mise-en-oeuvre VLSI d'un module élémentaire réalisée par des étudiants du Groupe de Recherche en Micro-électronique à démontré les performances potentielles d'une telle architecture.

Un système de catégorisation qui s'applique au triage métrique à débit élevé comprend généralement un algorithme de catégorisation en ligne, qui utilise une technique de traitement appropriée. Dans la troisième contribution, le traitement par reordonnancement a été proposé pour gérer la manière dont les patrons d'un séquence d'entrée sont appris par l'algorithme de catégorisation. Avec cette technique, chaque patron d'entrée, qui mène à une décision ambiguë, n'est appris qu'après un temps fixe. La théorie sur l'option de rejet a permi de dériver deux modèles pratiques pour faire la détection des cas ambigus. La latence requise pour effectuer des catégorisations en ligne a été dérivée pour un système de catégorisation qui utilise le traitement

séquentiel (le cas par défaut), par lot et par ré-ordonnancement.

Les résultats de simulation obtenus avec un ensemble de données radars et deux RNA (ART2A-E et fuzzy ART) qui utilisent le traitement séquentiel, par lot et par ré-ordonnancement, ont permis de tirer les conclusions suivantes. Premièrement, le nombre de patrons d'entrée qui mène à une décision ambiguë est indicatif de la dégradation des résultats. Deuxièmement, un RNA qui utilise le traitement par ré-ordonnancement produit une qualité de catégorisation plus élevée qu'un même RNA qui utilise le traitement séquentiel. Ce gain en qualité s'obtient avec un temps de réponse supplémentaire qui est modeste. Finalement, le traitement par ré-ordonnancement offre une alternative intéressante au traitement par lot en termes du compromis entre la qualité des catégorisations et le temps de réponse. Ce traitement permet alors d'améliorer la qualité des catégorisations, tout en permettant de contrôler le temps maximum de traitement d'un patron.

Dans la quatrième contribution, un RNA à fusion "what-and-where" a été proposé pour l'identification rapide des types de radar associés aux impulsions interceptées. Les paramètres "what" forment l'entrée pour le RNA classificateur, qui prédit les types de radar associés aux impulsions, tandis que les paramètres "where" forment l'entrée pour le sous-système de catégorisation en-ligne, qui sépare les impulsions transmises par différents émetteurs. Cette séparation permet d'accumuler les réponses du classificateur pour chaque émetteur, et donc de prédire le type de radar d'un émetteur actif d'après une séquence d'impulsions (pour améliorer la précision).

Des simulations ont été effectuées pour une mise-en-oeuvre particulière du RNA à fusion "what-and-where" et pour un ensemble de données radars. Avec cette mise-en-oeuvre, le RNA classificateur est réalisé par une variante du fuzzy ARTMAP, tandis que le sous-système de catégorisation est réalisé par un algorithme qui exécute l'association du type plus-proche-voisin et le filtrage de Kalman. Les résultats de simulation ont démontré une amélioration significative des performances par rapport au fuzzy ARTMAP modifié seul. Le système obtient un taux de classification d'environ 98% avec un lot d'entrainement formé d'aussi peu que 0.15% de l'ensemble de données complet. On peut anticiper des performances encore meilleures si le système proposé exploite des paramètres "where" plus puissants, comme le MOP.

La performance du RNA fuzzy ARTMAP a été comparée avec celle de plusieurs autres classificateurs. Les résultats de simulation ont révélé que le fuzzy ARTMAP permet d'obtenir un des meilleurs taux de classification parmi ces approches. De plus, fuzzy ARTMAP est parmi les classificateurs qui sont les plus efficaces en termes du temps de convergence, de la mémoire pour stocker les prototypes et de la complexité des calculs. Des extensions au fuzzy ARTMAP original lui ont permi (1) de converger quand le lot d'entrainement contenait des impulsions dont les paramètres "what" sont quasi-identiques, mais qui sont liées à différents types de radar, (2) de traiter des PDWs dont les paramètres "what" sont absents, (3) de détecter des impulsions émises par des types de radar non-familiers, et (4) d'apprendre les classes correspondantes en temps réel.

Suggestions de travaux futurs

Les RNA SOFM et fuzzy ART comparés dans la première contribution ne sont pas pratiques, comme tels, pour le triage métrique en MSE radar. Par exemple, des modifications sont requises pour permettre au SOFM de supporter l'apprentissage continu de nouvelles informations dans un environnement dynamique. De plus, un mécanisme est nécessaire pour faciliter l'interprétation de ses résultats. Par contre, le fuzzy ART est plus directement applicable au triage métrique. Cependant, ses résultats manquent de précision et varient selon l'ordre de présentation des données. Il serait pertinent de mener une étude plus approfondie concernant l'effet de l'ordre de présentation des données sur les résultats de fuzzy ART. Cette étude peut indiquer des ordres de présentation qui mènent généralement à de bonnes performances. Finalement, il serait utile de développer des critères pour pouvoir re-initialiser les neurones correspondant aux émetteurs qui ne sont plus actifs.

Il existe plusieurs variantes architecturales qui peuvent influencer la performance et la surface de l'architecture VLSI présentée dans la deuxième contribution. Par exemple, il serait possible avec cette architecture de re-initialiser les neurones s'il ne sont pas actifs pour un certain temps. Il serait aussi possible de remplacer le diviseur global par un multiplieur et de la mémoire supplémentaire, ou bien de distribuer des diviseurs bit-sériels localement. Finalement, il serait possible d'effectuer la phase d'apprentissage de façon préemptive.

Dans le cadre de la troisième contribution, il est certain que l'ambiguïté dans le

choix d'une catégorie a un impact sur la performance d'un RNA comme fuzzy ART. Il serait intéressant d'observer l'effet du traitement par re-ordonnancement sur d'autres algorithmes qui effectuent la catégorisation en ligne de séquences de patrons (*e.g.*, version en ligne de k-means). L'ambiguïté dans le choix d'une catégorie dépend du contexte, *e.g.*, de l'ordre d'apprentissage des patrons d'entrée et de la complexité de l'environement à traiter. Une stratégie qui peut réduire l'impact des décisions ambiguës consiste à utiliser l'ambiguïté comme critère pour ajuster la précision des bornes de décision. Il serait intéressant d'explorer différentes représentations de catégories, fonctions de choix, et lois d'apprentissages, qui permettent de raffiner les bornes de décision en fonction de l'ambiguïté.

Il existe plusieurs aspects qui seraient intéressant de développer pour le RNA à fusion "what-and-where." En ce qui concerne la détection de nouveauté, il serait souhaitable de comparer les performance de l'approche qui est proposée avec plusieurs autres méthodes. Il serait aussi souhaitable d'améliorer la mesure qui détecte les nouveautés pour tenir compte du chevauchement entre les classes qui sont apprises. Une considération importante pour les systèmes de MSE radar est la classification de signaux qui sont émis par des émetteurs inconnus. Le RNA à fusion "what-and-where" offre la possibilité de raffiner les connaissances du réseau dans le champs face à ces émetteurs. Malgré des résultats prometteurs, l'approche proposée pour modifier les poids synaptiques et le seuil pour la détection de nouveauté mérite plus de recherche. En fait, il serait pertinent d'entreprendre une étude approfondie des différents al-

gorithmes d'apprentissage qui permettent d'apprendre de nouvelles informations de façon incrémentale.

Le regroupement et l'identification de signaux radars sont présentement des tâches très exigeants pour les systèmes de MSE modernes. (Malheureusement, il est impossible de présenter et de comparer les performances obtenues avec ces systèmes conventiennels.) Cet ouvrage démontre néanmoins les avantages du traitement neuronique, comme supplément aux approches conventionnelles, pour effectuer ces deux tâches. Les RNA comme ceux de la famille ART peuvent effectivement jouer un rôle important dans les systèmes MSE autonomes et adaptatifs du futur.

Bibliographie

[1] AHALT, S.C., KRISHNAMURTHY, A.K., CHEN, P., et MELTON, D.E., "Competitive learning algorithms for vector quantization," *Neural Networks*, **3**, 277-290 (1990).

[2] ALTOFT, J., et FORD, B., "Applications of the Dempster-Shafer belief theory to CANEWS2 (U)," *DREO Technical Report TR 1243*, Defence Research Establishment Ottawa (octobre 1994).

[3] ANDERBERG, M.R., *Cluster Analysis for Applications*, Academic Press (1973).

[4] ANDERSON, J.A., GATELY, M.T., PENZ, P.A., et COLLINS, D.R., "Radar signal categorization using a neural network," *Proc. IEEE*, **78:10**, 1646-1656 (1990).

[5] BALL, G.H., et HALL, D.J., "ISODATA, an iterative method of multivariate analysis and pattern classification", *Proc. IFIPS Congress* (1965).

[6] BALOCH, A.A., et WAXMAN, A.M., "Visual learning, adaptive expectation, and behavioral conditioning of the mobile robot MAVIN," *Neural Networks*, **4**, 271-302 (1991).

[7] BAR-SHALOM, Y., et LI, R.A., *Estimation and Tracking: Principles, Techniques and Software*. Artech House (1993).

[8] BENSAID, A.M., HALL, L.O., BEZDEK, J.C., et CLARKE, L.P., "Partially supervised clustering for image segmentation," *Pattern Recognition*, **29**, 859-871 (1996).

[9] BEZDEK, J.C., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1987.

[10] BEZDEK, J.C., "Some non-standard clustering algorithms", *NATO ASI Series, Vol. G14, Developments in Numerical Ecology*, P. Legendre and L. Legendre, Eds., Springer-Verlag (1987).

[11] BISHOP, C., *Neural Networks for Pattern Recognition*, Clarendon Press (1995).

[12] BLACKMAN, S.S., *Multiple-Target Tracking with Radar Applications*, Artech House (1986).

[13] BRADSKI, G., et GROSSBERG, S., "STORE working memory networks for storage and recall of arbitrary temporal sequences," *Biological Cybernetics*, **71**, 469-480 (1994).

[14] BRADSKI, G., et GROSSBERG, S., "Fast-learning VIEWNET architectures for recognizing three-dimensional objects from multiple two-dimensional views," *Neural Networks*, **8:7**, 1053-1080 (1995).

[15] BRIDLE, J.S., "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing – Algorithms, Architectures and Applications*, eds., Fogelman-Soulié, F., Hérault, J., NATO ASI Series F68, Springer-Verlag, 227-236 (1989).

[16] BROWNS, J.P.R., et THURBON, M.T., *Electronic Warfare*, Brassey's (1998).

[17] CANTIN, M.-A., "Implantation du réseau de neurones Fuzzy ART," Thèse de maîtrise, Université du Québec à Montréal (octobre 1998).

[18] CANTIN, M.-A., SAVARIA, Y., BLAQUIÈRE, Y., GRANGER E., et LAVOIE, P., "Four implementations of the fuzzy adaptive resonance theory (ART) neural network for high data throughput applications," soumis pour publication *IEEE Trans. on Very Large Scale Integration* (mai 2000).

[19] CANTIN, M.-A., "Integration Support for the ICDPMFA3: Fuzzy ART Neural Network ASIC," manuel d'usager (janvier 2001).

[20] CARPENTER, G.A., et GROSSBERG, S., "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer, Vision, Graphics and Image Processing*, **37**, 54-115 (1987).

[21] CARPENTER, G.A., GROSSBERG, S., et ROSEN, D.B., "Fuzzy ART: fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, **4:6**, 759-771 (1991).

[22] CARPENTER, G.A., GROSSBERG, S., et REYNOLDS, J.H., "ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network," *Neural Networks*, **4**, 565-588 (1991).

[23] CARPENTER, G.A., et GROSSBERG, S., eds., *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, 1991.

[24] CARPENTER, G.A., GROSSBERG, S., MARKUZON, N., REYNOLDS, J.H., et ROSEN, D.B., "Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, **3:5**, 698-713 (1992).

[25] CARPENTER, G.A., et ROSS, W.D., "ART-EMAP: a neural network architecture for object recognition by evidence accumulation," *IEEE Trans. on Neural Networks*, **6:4**, 805-818 (1995).

[26] CARPENTER, G.A, RUBIN, M.A., et STREILEIN, W.W., "ARTMAP-FD: familiarity discrimination applied to radar target recognition," *Proc. 1997 IEEE Int'l Conference on Neural Networks*, **3**, 1459-1464, Houston, USA (juin 1997).

[27] CARPENTER, G.A, RUBIN, M.A., et STREILEIN, W.W., "Threshold determination for ARTMAP-FD familiarity discrimination," C. H. Dagli *et al.*, eds.,

Intelligent Engineering Systems Through Artificial Neural Networks 7, ASME Press, 23-28 (1997).

[28] CARPENTER, G.A., et MARKUZON, N., "ARTMAP-IC and medical diagnosis: instance counting and inconsistent cases," *Neural Networks*, **11:2**, 323-336 (1998).

[29] CAUDELL, T.P., "Hybrid optoelectronic adaptive resonance theory neural processor, ART1," *Applied Optics*, **31:29**, 6220-6229 (1992).

[30] CAVANAGH, J.F. *Digital Computer Arithmetic, Design and Implementation*, McGraw-Hill, 289-301 (1984).

[31] CHANDRA, V., JYOTISHI, B.K., et BAJPAI, R.C., "Some new algorithms for ESM data processing," *Proc. 20th Southeastern Symposium on System Theory*, 108-112, Charlotte, USA (mai 1988).

[32] CHEN, S., COWAN, C.F.N., et GRANT, P.M., "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. on Neural Networks*, **2:2**, 302-309 (1991).

[33] CHOW, C.K., "An optimum character recognition system using decision functions," *IRE Trans. on Electronic Computers*, **6**, 247-254 (1957).

[34] CHOW, C.K., "On optimum recognition error and reject tradeoff," *IEEE Trans. on Information Theory*, **16**, 41-46 (1970).

[35] CANADIAN MICROELECTRONICS CORPORATION, *BiCMOS Edge Support, Version 1.0* (juin 1993).

[36] CANADIAN MICROELECTRONICS CORPORATION, *cmc128 × 32 (128 word × 32 bit) BiCMOS 1-Port SRAM* (août 1995).

[37] CORMAN, T.H., LEISERSON C.E., et RIVEST, R.L., *Introduction to Algorithms*, MIT Press (1990).

[38] COVER, T.M., et HART, P.E., "Nearest neighbor patterns classification," *IEEE Trans. on Information Theory*, **13:1**, 21-27 (1967).

[39] CRESPO, J.F., LAVOIE P., et SAVARIA, Y., "Fast convergence with low precision weights in ART1 networks," *Proc. 1994 IEEE Int'l Symp. Circuits and Systems*, **6**, 237-204, London, UK (1994).

[40] DAVIES, C.L., et HOLLANDS, P., "Automatic processing for ESM," *Proc. IEE*, **129:3 (F)**, 164-171 (juin 1982).

[41] DEMIRIZ, A., BENNETT, K.P., et EMBRECHTS, M.J., "Semi-supervised clustering using genetic algorithms," Dagli, C. H. et al., eds., *Intelligent Engineering Systems Through Artificial Neural Networks 9*, ASME Press, 809-814 (1999).

[42] DESIENO, D., "Adding a conscience to competitive learning," *Proc. IEEE Int'l Conf. Neural Networks*, 1117-1124, San Diego, USA (juin 1988).

[43] DUBES, R.C., et JAIN A.K., *Algorithms for Clustering Data*, Prentice Hall (1988).

[44] DUBUISSON, B., et MASSON, M., "A statistical decison rule with incomplete knowledge about classes," *Pattern Recognition*, **26:1**, 155-165 (1993).

[45] DUDA, R.O., et HART, P.E., *Pattern Classification and Scene Analysis*, John Weiley & Sons (1973).

[46] FISHER, R.A., "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, **7** (1936).

[47] FORD B., MICKEAL J., et GRANGER, E., "A comparative analysis of selected ESM deinterleavers (U)," DREO Technical Report TR 2000-097, Defence Research Establishment Ottawa (novembre 2000).

[48] FRANK, T., KRAISS, K.-F., et KUHLAN, T., "Comparative analysis of Fuzzy ART and ART-2A network clustering performance," *IEEE Trans. on Neural Networks*, **9:3**, 544-559 (1998).

[49] FUKUNAGA, K., et KESSELL, D.L., "Application of optimal error-reject functions," *IEEE Trans. on Information Theory*, 814-817 (novembre 1972).

[50] FUKUNAGA, K., *Introduction to Statistical Pattern Recognition*, Academic Press (1990).

[51] GERSHO. A., "On the structure of vector quantizers," *IEEE Trans. on Information Theory*, **28:2**, 157-166 (1982).

[52] GHAHRAMANI, Z., et JORDAN, M.I., "Learning from incomplete data," *A. I. Memo No. 1509, C. B. C. L. Paper No. 108*, M. I. T. Artificial Intellignece Laboratory and Center for Biological and Computational Learning, Dept. of Brain and Cognitive Sciences (1994).

[53] GRANGER, E., BLAQUIÈRE, Y., SAVARIA, Y., CANTIN, M.-A., et LAVOIE, P., "A VLSI architecture for fast clustering with the fuzzy ART neural network," *Proc. Int'l Workshop on Neural Networks for Identification, Control, Robotics, and Signal Processing (NICROSP'96)*, 117-125, Venice, Italy (août 1996).

[54] GRANGER, E., BLAQUIÈRE, Y., SAVARIA, Y., CANTIN, M.-A., et LAVOIE, P., "A VLSI architecture for fast clustering with the fuzzy ART neural network," *Journal of Microelectronic Systems Integration*, **5:1**, 3-18 (1997).

[55] GRANGER, E., SAVARIA, Y., LAVOIE P., et CANTIN, M.-A., "A comparison of self-organizing neural networks for fast clustering of radar pulses," *Signal Processing*, **64:3**, 249-269 (1998).

[56] GRANGER, E., GROSSBERG, S., RUBIN, M.A., et STREILEIN, W.W., "Familiarity discrimination of radar pulses," Kearns, M. S., et al., eds., *Advances in Neural Information Processing Systems 11*, MIT Press, 875-881 (1999).

[57] GRANGER, E., GROSSBERG, S., LAVOIE, P., et RUBIN, M. A., "A comparison of classifiers for radar emitter type identification," Dagli, C. H. et al., eds., *Intelligent Engineering Systems Through Artificial Neural Networks 9*, ASME Press, 3-11 (1999).

[58] GRANGER, E., RUBIN, M.A., GROSSBERG, S., et LAVOIE, P., "Classification of incomplete data using the fuzzy ARTMAP neural network," *Proc. 2000 Int'l Joint Conference on Neural Networks*, **4**, 35-40, Como, Italy (juillet 2000).

[59] GRANGER, E., RUBIN, M.A., GROSSBERG, S., et LAVOIE, P., "Radar ESM with a What-and-Where fusion neural network," Miller, D. J. et al., eds., *Proc. 2001 IEEE Workshop on Neural Neural Networks for Signal Processing XI*, IEEE Press, 539-548 (2001).

[60] GRANGER, E., RUBIN, M.A., GROSSBERG, S., et LAVOIE, P., "A What-and-Where fusion neural network for recognition and tracking of multiple radar emitters," *Neural Networks*, **14:3**, 325-344 (2001).

[61] GRANGER, E., SAVARIA, Y., et LAVOIE, P., "A pattern reordering approach based on ambiguity detection for on-line category learning," *Département de génie électrique, École Polytechnique de Montréal, EPM/RT-01/02*, 40 pages (septembre 2001).

[62] GRANGER, E., SAVARIA, Y., et LAVOIE, P., "A pattern reordering approach based on ambiguity detection for on-line category learning," *IEEE Trans. on*

Pattern Analysis and Machine Intelligence, soumis pour publication, , l'article a été resoumis pour une 2e phase de revision (novembre 2001).

[63] GRANT, P.M., et COLLINS, J.H., "Introduction to electronic warfare," *Proc. IEE*, **129:3 (F)**, 621-625 (juin 1982).

[64] GRAY, R.M., "Vector quantization," *IEEE ASSP Magazine*, **1:2**, 4-29 (1984).

[65] GROSSBERG, S., "Adaptive pattern classification and universal recoding: I. parallel development and coding of neural detectors," *Biological Cybernetics*, **23**, 121-134 (1976).

[66] GROSSBERG, S., "Adaptive pattern classification and universal recoding: II. feedback, oscillation, olfaction, and illusions," *Biological Cybernetics*, **23**, 187-207 (1976).

[67] GROSSBERG, S., "Competitive learning: from interactive activation to adaptive resonance," *Cognitive Science*, **11**, 23-63 (1987).

[68] GROSSBERG, S., "The complementary brain: A unifying view of brain specialization and modularity," *Trends in Cognitive Sciences*, in press (2000).

[69] HA, T. M., "The optimal class-selective rejection rule," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **19:6**, 608-615 (1997).

[70] HADAWAY, R., KEMPF, P., SCHVAN, P., ROWLANDSON, M., HO, V., KOLK, J., TAIT, B., SUTHERLAND, D., et JOLLY, G., "A sub-micron BiC-

MOS technology for telecommunications," *Proc. 21st European Solid State Device Research Conference*, 513-516 (1991).

[71] HARTIGAN, J.A., *Clustering Algorithms*, Wiley (1975).

[72] HAYKIN, S., *Neural Networks: A Comprehensive Foundation*, Macmillan College Publishing Co. (1994).

[73] HELLMAN, M.E., "The nearest neighbor classification rule with reject option," *IEEE Trans. on Systems Science and Cybernetics*, **6:3**, 179-185 (1970).

[74] HELSTROM, C.W., *Elements of Signal Detection and Estimation*, Prentice Hall (1995).

[75] HECHT-NIELSEN, R., "Applications of counterpropagation networks," *Neural Networks*, **1**, 131-141 (1988).

[76] HUBERT, L., et ARABIE, P., "Comparing partitions," *Journal of Classification*, **2**, 193-218 (1985).

[77] JONES S.R., SAMMUT K.M., NEILSON C.H., et STRAUNTRUP J., "Toroidal neural networks: architectures and processor granularity issues," Ramacher, U., and Ruckert, U. et al., eds., *VLSI Design of Neural Networks*, Klewer Academic Publisher, 229-254 (1991).

[78] JONES S.R., SAMMUT, K.M. et HUNTER, J., "Learning in linear systolic neural network engines: analysis and implementation," *IEEE Transactions on Neural Networks*, **5:4**, 584-593 (1994).

[79] KAMGAR-PARSI, BEHROOZ, KAMGAR-PARSI, BEHZAD, et SCIORTINO, J.C., "Automatic data sorting using neural network techniques," *Naval Research Laboratory Report NRL/FR/5720-96-9803*, (février 1996).

[80] KANE, J.S., et PAQUIN, M.J., "POPART: partial optical implementation of adaptive resonance theory 2," *IEEE Trans. on Neural Networks*, **4:4**, 695-702 (1993).

[81] KEYVAN, S., RABELO, L.C., et MALKANI, A., "Nuclear reactor condition monitoring by adaptive resonance theory", *Proc. 1992 Int'l Joint Conference on Neural Networks*, Baltimore, USA, **3**, 321-28 (1992).

[82] KIM, Y.S., et MITRA, S., "A comparative study of the performance of fuzzy ART-type clustering algorithms in pattern recognition", *Proc. SPIE: Intelligent Robots and Computer Vision XI*, 335-341, Boston, USA (novembre 1992).

[83] KIM, Y.S., et MITRA, S., "Integrated adaptive fuzzy clustering (IAFC) algorithm", *Proc. 1993 Fuzz-IEEE*, 1264-1268 (juin 1993).

[84] KIM, Y.S., et MITRA, S., "An adaptive integrated fuzzy clustering model for pattern recognition", *Fuzzy Sets and Systems*, **65**, 297-310 (1994).

[85] KOHONEN, T., "Self-organizing formation of topologically correct feature maps", *Biological Cybernetics*, **43**, 59-69 (1982).

[86] KOHONEN, T., *Self-organization and associative memory*, Springer-Verlag (1989).

[87] KOSKO, B., *Neural Networks and Fuzzy Systems: A Dynamical Approach to Machine Intelligence*, Prentice-Hall (1992).

[88] KUNG, S.Y., *Digital Neural Networks*, Prentice-Hall, 337-406 (1993).

[89] LAWRENCE, S., GILES, C.L., TSOI, A.C., et BACK, A.D., "Face recognition: a convolutional neural-network approach", *IEEE Trans. on Neural Networks*, **8:1**, 98-113 (1997).

[90] LEE, J.P.Y., "Wideband I/Q demodulators: measurement technique and matching characteristics", *Proc. IEE: Radar, Sonar, Navigation*, **143:5**, 300-306, (octobre 1996).

[91] LIN, C.-T., et GEORGE LEE, C.S., *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems*, Prentice-Hall (1995).

[92] LINDE, Y., BUZO, A., et GRAY, R.M., "An algorithm for vector quantizer design," *IEEE Trans. on Communications*, **28:1**, 84-95 (1980).

[93] HO, C.S., LIOU, J.J., GEORGIOPOULOS, M., et CHRISTODOULOU, C., "A mixed analog/digital vlsi design and simulation of an adaptive resonance theory (ART) neural network architecture," *Simulation*, **66:1**, 31-42 (1996).

[94] LITTLE, R.J.A., et RUBIN, D.B., *Statistical Analysis with Missing Data*, Wiley (1987).

[95] LLOYD, S.P., "Least-square quantization in PCM," *IEEE Trans. on Information Theory*, **28:2**, 129-137 (1982).

[96] MACEDO FILHO, A.D., et GRIFFITHS, H.D., "Radar signal clustering and identification by means of microwave neual networks," *Proc. 1994 Int'l Radar Conference*, 470-475, Paris, France (May 1994).

[97] MACQUEEN, J., "Some methods for classification analysis of multivariate observations," *Proc. 5th Berkley Symposium on Mathematical Statistics and Probability*, 281-297 (1967).

[98] MALONEY, P. S., et SPECHT, D. F., "The use of probabilistic neural networks to improve solution times for emitter-to-hull correlation problems," *Proc. 1989 Int'l Joint Conference on Neural Networks*, **1**, 289-294, Washington, USA (juin 1989).

[99] VON DER MALSBERG, C., "Self-organization of orientation sensitive cells of the striate cortex", *Kybernetik*, **14**, 85-100 (1973).

[100] MARDIA, H.K., "New techniques for the deinterleaving of repetitive sequences," *Proc. IEE*, **136:4 (F)**. 149-154 (août 1989).

[101] MEILER, P.P., et VAN WEZENBEEK, A. M., "BSB radar pulse classification," *TNO Physics and Electronics Laboratory Report FEL-90-B023* (août 1990).

[102] MILLIGANG. W., SOON S.C., et SOKOL, L.M., "The effect of cluster size, dimensionality, and the number of clusters on recovery of true cluster structure", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **5**, 40-47 (1983).

[103] MOORE, B., "ART1 and pattern clustering", Touretski, XXX., et al., eds., *Proc. 1988 Connectionist Models Summer School*, Morgan Kaufmann Publishers, 174-185 (1988).

[104] MURTAGH, F., "Interpreting the Kohonen self-organizing feature map using contiguity-constrained clustering", *Pattern Recognition Letters*, **16**, 399-408 (1995).

[105] PAPE, D.R., ANDERSON, J.A., CARTER, J.A., et WASILOUSKY, P.A, "Advanced signal waveform classifier," *Proc. SPIE - The Int'l Society of Optical Engineers*, **3160**, 162-169, San Diego, USA (juillet 1997).

[106] PARRA, L., DECO, G., et MIESBACH, S., "Statistical independance and novelty detection with information preserving non-linear maps," *Neural Computation*, **8**, 260-269 (1996).

[107] PEDRYCZ, W., "Algorithms for fuzzy clustering with partial supervision," *Pattern Recognition Letters*, **3**, 13-20 (1985).

[108] PEPER, R., ZHANG, B., et NODA, H., "A comparative study of the ART2-A and the self-organizing feature map", *Proc. 1993 Int'l Joint Conf. on Neural Networks*, 1425-1429 (1993).

[109] PESONEN, E., ESKELINEN, M., et JUHOLA, M., "Comparison of different neural network algorithms in the diagnosis of acute appendicitis", *Int'l Journal of Bio-Medical Computing*, **40:3**, 227-233 (1996).

[110] POIRÉ, P., SAVARIA, Y., CANTIN, M.-A., DANIEL, H., et BLAQUIÈRE, Y., "Hardware/software codesign of a fuzzy ART neural clusterer: The Benefits of Reconfigurable Computing," *Proc. of SPIE*, vol. 3526, 90-96, 1998.

[111] RAMACHER, U., "Guidelines to VLSI design of neural nets," Ramacher, U., and Ruckert, U. et al., eds., *VLSI Design of Neural Networks*, Klewer Academic Publisher, 2-34 (1991).

[112] RAO, A., WALKER, M.R., CLARK, L.T., et AKERS, L.A., "Integrated circuit emulation of ART1 Networks," *Proc. 1989 IEEE Conference on Artificial Neural Networks*, 37-41, (1989).

[113] RAO, A., WALKER, M.R., CLARK, L.T., AKERS, L.A., et GRONDIN, R.O., "VLSI implementation of neural classifiers," *Neural Computation*, **2**, 35-43 (1990).

[114] RATANAPAN, K., et DAGLI, C.H., "Implementation of ART1 architecture on CNAPS neuro-computer," *Proc. of the SPIE - The Int'l Society of Optical Engineers*, **2492:1**, 104-110 (1995).

[115] ROE, J., CUSSONS, S., et FELTHAM, A., "Knowledge-based signal processing for radar ESM systems," *Proc. IEE (UK)*, **137:5(F)**, 293-301 (octobre 1990).

[116] ROE, A.L., et ROE, J., "The application of artificial intelligence techniques to naval ESM radar identification," *Proc. 1994 Int'l Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-94)*, 565-572, Austin, USA (juin 1994).

[117] ROGERS, J.A.V, "ESM processor system for high pulse density radar environments," *Proc. IEE*, **132:7 (F)**, 113-129 (december 1985).

[118] ROGERS, J.A.V, "SADIE – improvements for in-sevice tuning," *Proc. IEE Colloquium on Signal Processing Techniques for EW*, **8** 46, London, UK (1992).

[119] RUMELHART, D.E., et ZIPSER, D., "Feature discovery by competitive learning," *Cognitive Science*, **9**, 75-112 (1985).

[120] RUMELHART, D.E., HINTON, G., et WILLIAMS, R.J., "Learning internal representations by error propagation," Rumelhart, D.E., et McClelland, J.L., eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, **1**, 318-362 (1986).

[121] SCHLEHER, D.C., *Introduction to Electronic Warfare*, Artech House (1986).

[122] SCHLEHER, D.C., *Electronic Warfare in the Information Age*, Artech House (1999).

[123] SCIORTINO, J.C., "Autonomous ESM systems," *Naval Engineers Journal*, 73-84 (novembre 1997).

[124] SELF, A., et BOURASSA, G., "Future ESM systems and the potential for neural processing," *AGARD'91*, **9**, 1-10, (novembre 1991).

[125] SERRANO-GOTARREDONA, T., LINARES-BARRANCO, B., et HUERTAS, J.L., "A real time clustering CMOS neural engine", Tesauro, G., *et al.*, eds., *Advances in Neural Information Processing Systems 7*, MIT Press, 755-760 (1994).

[126] SIMPSON, P., "Fuzzy min-max neural networks - part 2: clustering", *IEEE Trans. on Fuzzy Systems*, **1:1**, 32-45 (1993).

[127] SKOLNIK, M.I., *Introduction to Radar Systems*, McGraw-Hill (1980).

[128] SPECHT, D.F., "Probabilistic neural networks (a one-pass learning method) and potential applications," *WESCON'89*, 780-785, (mai 1989).

[129] SPECHT, D.F., "Probabilistic neural networks," *Neural Networks*, **3**, 109-118 (1990).

[130] TOU J.T., et GONZALEZ, R.C., *Pattern Recognition Principles*, Addison-Wesley (1974).

[131] TSAY, S.W., EL-LEITHY, N., et NEWCOMB, R.W., "CMOS realization of a class of hartline neural pools," *Proc. 1990 IEEE Int'l Symp. Circuits and Systems*, 2417-2420, New Orleans, USA (mai 1990).

[132] TSAY, S.W., et NEWCOMB, R.W., "VLSI implementation of ART1 memories," *IEEE Trans. on Neural Networks*, **2:2**, 214-221 (1991).

[133] TSUI, J.B., *Microwave Receivers with Electronic Warfare Applications*, John Wiley and Sons (1986).

[134] VALIN, P., COUTURE, J., et SIMARD, M.-A., "Position and attribute fusion of radar, ESM, IFF and datalink for AAW missions of the canadian patrol frigate", *Proc. 1996 IEEE/SICE/RSJ Int't Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 63-71, Washington, USA (mai 1996).

[135] WANG, D.C., et THOMPSON, J., "An adaptive data sorter based on probabilistic neural networks," *Proc. 1991 IEEE NAECOM*, **3**, 1-12, Dayton, USA (mai 1991).

[136] WANG, Z., GUERRIERO, A., et DE SARIO, M., "Comparison of several approaches for the segmentation of texture images", *Proc. SPIE - The Int'l Society of Optical Engineering*, **2424**, 580-591 (1995).

[137] WANG, G., HE, Y., GAO, Z., et LIU, Y., "Improved association of ESM measurements with radar tracks," *Proc. 1997 Int'l Radar Conf.*, 648-652, Edinburg, UK (juillet 1997).

[138] WANN C.-D., et THOMOPOULOS, C.A., "Comparative study of self-organizing neural networks", *Proc. IWANN'93*, 316-21, Sitges, Spain (juin 1993).

[139] WILEY, R.G., *Electronic Intelligence: The Analysis of Radar Signals*, Artech House, 1993.

[140] WILKINSON, M.A., et WATSON, M.A., "Use of metric techniques in ESM data processing," *Proc. IEE*, **132:4 (F)**, 229-232 (juillet 1985).

[141] WILLIAMSON, J.R., "Gaussian ARTMAP: a neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, **9:5**, 881-897 (1996).

[142] WILLIAMSON, J.R., "A constructive, incremental-learning neural network for mixture modeling and classification," *Neural Computation*, **9:7**, 1517-1543 (1997).

[143] WITTALL, N.J., "Signal sorting in ESM systems," *Proc. IEE (UK)*, **132:4 (F)**, 226-228 (juillet 1985).

[144] WUNSCH, D.C., CAUDELL, T.P., CAPPS, C.D., MARKS, R.J., et FALK, R.A., "An Optoelectronic implementation of the adaptive resonance neural networks," *IEEE Trans. on Neural Networks*, **4:4**, 673-684 (1993).

[145] ZURADA, J.M., *Introduction to Artificial Neural Systems*, West Publishing Co., 1992.