Titre: Title:	Emotion Recognition with Deep Neural Networks
Auteur: Author:	Samira Ebrahimi Kahou
Date:	2016
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Ebrahimi Kahou, S. (2016). Emotion Recognition with Deep Neural Networks [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/2290/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/2290/
Directeurs de recherche: Advisors:	Christopher J. Pal
Programme: Program:	Génie informatique

UNIVERSITÉ DE MONTRÉAL

EMOTION RECOGNITION WITH DEEP NEURAL NETWORKS

SAMIRA EBRAHIMI KAHOU
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(GÉNIE INFORMATIQUE)
JUILLET 2016

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

EMOTION RECOGNITION WITH DEEP NEURAL NETWORKS

présentée par: EBRAHIMI KAHOU Samira
en vue de l'obtention du diplôme de: Philosophiæ Doctor
a été dûment acceptée par le jury d'examen constitué de:

Mme CHERIET Farida, Ph. D., présidente
M. PAL Christopher J., Ph. D., membre et directeur de recherche
M. HURTUT Thomas, Ph. D., membre
M. COHN Jeffrey F., Ph. D., membre externe

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Professor Christopher Pal for introducing me to the field of computer vision and machine learning and guiding me patiently throughout my Ph.D. studies. I am grateful for being given the chance to be part of the Montreal Institute for Learning Algorithms (MILA) and collaborate with brilliant deep learning experts at Université de Montréal, Professors Yoshua Bengio, Roland Memisevic, Pascal Vincent and Aaron Courville.

I thank the members of my committee, Professors Farida Cheriet, Marc Laforest, Thomas Hurtut and Jeffrey F. Cohn, one of the pioneers in automatic facial analysis, for their valuable time.

I appreciate all the help from MILA members in coding-related questions, especially Frédéric Bastien, Pascal Lamblin, Vincent Michalski, Guillaume Alain and Mehdi Mirza. I thank all my co-authors without whom this work would not have been possible, especially Pierre Froumenty, Vincent Michalski, Kishore Konda, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre and Adriana Romero for all their hard work until the last minute before deadlines.

I thank Roland Memisevic and Yoshua Bengio for regular meetings and their valuable insights in my research. Thanks also to Guillaume Alain for help with the French abstract, Rosemary Ke and Anirudh Goyal for proofreading and helpful comments.

I have met many great people during my studies. To name a few of my friends at Polytechnique and MILA, whom I have not mentioned yet, I would like to thank, Fannie Puech, Louis-Philippe Bourret, Samira Shabanian, Soroush Mehri, Dmitrii Serdiuk, Dzmitry Bahdanau, David Krueger, Philemon Brakel, Jörg Bornschein, Faruk Ahmed, Mélanie Ducoffe and Dustin Webb.

Most of all, I am thankful to my family for their exceptional support through all challenges and Vincent whose love and encouragement motivated me to complete this thesis.

Finally, I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and Ubisoft for funding my research.

RÉSUMÉ

La reconnaissance automatique des émotions humaines a été étudiée pendant des décennies. Il est l'un des éléments clés de l'interaction homme-ordinateur dans les domaines des soins de santé, de l'éducation, du divertissement et de la publicité. La reconnaissance des émotions est une tâche difficile car elle repose sur la prédiction des états émotionnels abstraits à partir de données d'entrée multimodales. Ces modalités comprennent la vidéo, l'audio et des signaux physiologiques. La modalité visuelle est l'un des canaux les plus informatifs. Notons en particulier les expressions du visage qui sont un très fort indicateur de l'état émotionnel d'un sujet. Un système automatisé commun de reconnaissance d'émotion comprend plusieurs étapes de traitement, dont chacune doit être réglée et intégrée dans un pipeline. Ces pipelines sont souvent ajustés à la main, et ce processus peut introduire des hypothèses fortes sur les propriétés de la tâche et des données. Limiter ces hypothèses et utiliser un apprentissage automatique du pipeline de traitement de données donne souvent des solutions plus générales.

Au cours des dernières années, il a été démontré que les méthodes d'apprentissage profond mènent à de bonnes représentations pour diverses modalités. Pour de nombreux benchmarks, l'écart diminue rapidement entre les algorithmes de pointe basés sur des réseaux neuronaux profonds et la performance humaine. Ces réseaux apprennent hiérarchies de caractéristiques. Avec la profondeur croissante, ces hiérarchies peuvent décrire des concepts plus abstraits. Cette progrès suggèrent d'explorer les applications de ces méthodes d'apprentissage à l'analyse du visage et de la reconnaissance des émotions. Cette thèse repose sur une étude préliminaire et trois articles, qui contribuent au domaine de la reconnaissance des émotions.

L'étude préliminaire présente une nouvelle variante de Patterns Binaires Locales (PBL), qui est utilisé comme une représentation binaire de haute dimension des images faciales. Il est commun de créer des histogrammes de caractéristiques de PBL dans les régions d'images d'entrée. Toutefois, dans ce travail, ils sont utilisés en tant que vecteurs binaires de haute dimension qui sont extraits à des échelles multiples autour les points clés faciales détectées. Nous examinons un pipeline constitué de la réduction de la dimensionnalité non supervisé et supervisé, en utilisant l'Analyse en Composantes Principales (ACP) et l'Analyse Discriminante Fisher Locale (ADFL), suivi d'une Machine à Vecteurs de Support (MVS) comme classificateur pour la prédiction des expressions faciales. Les expériences montrent que les étapes de réduction de dimensionnalité fournissent de la robustesse en présence de bruit dans points clés. Cette approche atteint, lors de sa publication, des performances de l'état de l'art dans la reconnaissance de l'expression du visage sur l'ensemble de données Extended Cohn-Kanade (CK+) (Lucey *et al.*, 2010) et sur la détection de sourire sur l'ensemble de données GENKI (GENKI-4K, 2008). Pour la tâche de détection de sourire, un profond Réseau Neuronal Convolutif (RNC) a été utilisé pour référence fiable.

La reconnaissance de l'émotion dans les vidéos semblable à ceux de la vie de tous les jours, tels que les clips de films d'Hollywood dans l'Emotion Recognition in the Wild (EmotiW) challenge (Dhall *et al.*, 2013), est beaucoup plus difficile que dans des environnements de laboratoire contrôlées. Le premier article est une analyse en profondeur de la entrée gagnante de l'EmotiW 2013 challenge (Kahou *et al.*, 2013) avec des expériments supplémentaires sur l'ensemble de données du défi de l'an 2014. Le pipeline est constitué d'une combinaison de modèles d'apprentissage en profondeur, chacun spécialisé dans une modalité. Ces modèles comprennent une nouvelle technique d'agrégation de caractéristiques d'images individuelles pour permettre de transférer les caractéristiques apprises par réseaux convolutionnels (CNN) sur un grand ensemble de données d'expressions faciales, et de les application au domaine de l'analyse de contenu vidéo. On y trouve aussi un "deep belief net" (DBN) pour les caractéristiques audio, un pipeline de reconnaissance d'activité pour capturer les caractéristiques spatio-temporelles, ainsi qu'modèle de type "bag-of-mouths" basé sur k-means pour extraire les caractéristiques propres à la bouche. Plusieurs approches pour la fusion des prédictions des modèles spécifiques à la modalité sont comparés. La performance après un nouvel entraînement basé sur les données de 2014, établis avec quelques adaptations, est toujours comparable à l'état de l'art actuel.

Un inconvénient de la méthode décrite dans le premier article est l'approche de l'agrégation de la modalité visuelle qui implique la mise en commun par image requiert un vecteur de longueur fixe. Cela ne tient pas compte de l'ordre temporel à l'intérieur des segments groupés. Les Réseau de Neurones Récurrents (RNR) sont des réseaux neuronaux construits pour le traitement séquentiel des données. Ils peuvent résoudre ce problème en résumant les images dans un vecteur de valeurs réelles qui est mis à jour à chaque pas de temps. En général, les RNR fournissent une façon d'apprendre une approche d'agrégation d'une manière axée sur les données. Le deuxième article analyse l'application d'un RNR sur les caractéristiques issues d'un réseau neuronal de convolution utilisé pour la reconnaissance des émotions dans la vidéo. Une comparaison de la RNR avec l'approche fondée sur pooling montre une amélioration significative des performances de classification. Il comprend également une fusion au niveau de la caractéristiques et au niveau de décision de modèles pour différentes modalités. En plus d'utiliser RNR comme dans les travaux antérieurs, il utilise aussi un modèle audio basé sur MVS, ainsi que l'ancien modèle d'agrégation qui sont fusionnées pour améliorer les performances sur l'ensemble de données de défi EmotiW 2015. Cette approche a terminé en troisième position dans le concours, avec une différence de seulement 1% dans la précision de classification par rapport au modèle gagnant.

Le dernier article se concentre sur un problème de vision par ordinateur plus général, à savoir le suivi visuel. Un RNR est augmenté avec un mécanisme d'attention neuronal qui lui permet de se concentrer sur l'information liée à une tâche, ignorant les distractions potentielles dans la trame vidéo d'entrée. L'approche est formulée dans un cadre neuronal modulaire constitué de trois composantes : un module d'attention récurrente qui détermine *où* chercher, un module d'extraction de caractéristiques fournissant une représentation de *quel objet* est vu, et un module objectif qui indique *pourquoi* un comportement attentionnel est appris. Chaque module est entièrement différentiables, ce qui permet une optimisation simple à base de gradient. Un tel cadre pourrait être utilisé pour concevoir une solution de bout en bout pour la reconnaissance de l'émotion dans la vision, ne nécessitant pas les étapes initiales de détection de visage ou de localisation d'endroits d'intérêt. L'approche est présentée dans trois ensembles de données de suivi, y compris un ensemble de données du monde réel.

En résumé, cette thèse explore et développe une multitude de techniques d'apprentissage en profondeur, complétant des étapes importantes en vue de l'objectif à long terme de la construction d'un système entraînable de bout en bout pour la reconnaissance des émotions.

ABSTRACT

Automatic recognition of human emotion has been studied for decades. It is one of the key components in human computer interaction with applications in health care, education, entertainment and advertisement. Emotion recognition is a challenging task as it involves predicting abstract emotional states from multi-modal input data. These modalities include video, audio and physiological signals. The visual modality is one of the most informative channels; especially facial expressions, which have been shown to be strong cues for the emotional state of a subject. A common automated emotion recognition system includes several processing steps, each of which has to be tuned and integrated into a pipeline. Such pipelines are often hand-engineered which can introduce strong assumptions about the properties of the task and data. Limiting assumptions and learning the processing pipeline from data often yields more general solutions.

In recent years, deep learning methods have been shown to be able to learn good representations for various modalities. For many computer vision benchmarks, the gap between state-of-the-art algorithms based on deep neural networks and human performance is shrinking rapidly. These networks learn hierarchies of features. With increasing depth, these hierarchies can describe increasingly abstract concepts. This development suggests exploring the applications of such learning methods to facial analysis and emotion recognition. This thesis is based on a preliminary study and three articles, which contribute to the field of emotion recognition.

The preliminary study introduces a new variant of Local Binary Patterns (LBPs), which is used as a high dimensional binary representation of facial images. It is common to create histograms of LBP features within regions of input images. However, in this work, they are used as high dimensional binary vectors that are extracted at multiple scales around detected facial keypoints. We examine a pipeline consisting of unsupervised and supervised dimensionality reduction, using Principal Component Analysis (PCA) and Local Fisher Discriminant Analysis (LFDA), followed by a Support Vector Machine (SVM) classifier for prediction of facial expressions. The experiments show that the dimensionality reduction steps provide robustness in the presence of noisy keypoints. This approach achieved state-of-the-art performance in facial expression recognition on the Extended Cohn-Kanade (CK+) data set (Lucey *et al.*, 2010) and smile detection on the GENKI data set (GENKI-4K, 2008) at the time. For the smile detection task, a deep Convolutional Neural Network (CNN) was used as a strong baseline.

Emotion recognition in close-to-real-world videos, such as the Hollywood film clips in the Emotion Recognition in the Wild (EmotiW) challenge (Dhall *et al.*, 2013), is much harder than in controlled lab environments. The first article is an in-depth analysis of the EmotiW 2013 challenge winning entry (Kahou *et al.*, 2013) with additional experiments on the data set of the 2014 challenge. The pipeline consists of a combination of deep learning models, each specializing on one modality. The models include the following: a novel aggregation of per-frame features helps to transfer powerful CNN features learned on a large pooled data set of facial expression images to the video domain, a Deep Belief Network (DBN) learns audio features, an activity recognition pipeline captures spatio-temporal motion features and a k-means based bag-of-mouths model extracts features around the mouth region. Several approaches for fusing the predictions of modality-specific models are compared. The performance after re-training on the 2014 data set with a few adaptions is still competitive to the new state-of-the-art.

One drawback of the method described in the first article is the aggregation approach of the visual modality which involves pooling per-frame features into a fixed-length vector. This ignores the temporal order inside the pooled segments. Recurrent Neural Networks (RNNs) are neural networks built for sequential processing of data, which can address this issue by summarizing frames in a real-valued state vector that is updated at each time-step. In general, RNNs provide a way of learning an aggregation approach in a data-driven manner. The second article analyzes the application of an RNN on CNN features for emotion recognition in video. A comparison of the RNN with the pooling-based approach shows a significant improvement in classification performance. It also includes a feature-level fusion and decision-level fusion of models for different modalities. In addition to the RNN, the same activity pipeline as previous work, an SVM-based audio model and the old aggregation model are fused to boost performance on the EmotiW 2015 challenge data set. This approach was the second runner-up in the challenge with a small margin of 1% in classification accuracy to the challenge winner.

The last article focuses on a more general computer vision problem, namely visual tracking. An RNN is augmented with a neural attention mechanism that allows it to focus on task-related information, ignoring potential distractors in input frames. The approach is formulated in a modular neural framework consisting of three components: a recurrent attention module controlling *where* to look, a feature-extraction module providing a representation of *what* is seen and an objective module which indicates *why* an attentional behaviour is learned. Each module is fully differentiable allowing simple gradient-based optimization. Such a framework could be used to design an end-to-end solution for emotion recognition in vision, potentially not requiring initial steps of face detection or keypoint localization. The

approach is tested on three tracking data sets including one real-world data set.

In summary, this thesis explores and develops a multitude of deep learning techniques, making significant steps towards a long-term goal of building an end-to-end trainable systems for emotion recognition.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DRAW	Deep Recurrent Attentive Writer
EmotiW	Emotion Recognition in the Wild
FACS	Facial Action Coding System
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
ILSVRC	Large Scale Visual Recognition Challenge
IoU	Intersection-over-Union
LBP	Local Binary Pattern
LFDA	Local Fisher Discriminant Analysis
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
PCA	Principal Component Analysis
RATM	Recurrent Attentional Tracking Model
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

This thesis by article approaches the problem of *emotion recognition* by exploring and developing various *machine learning* and *computer vision* techniques. Studies by Ekman (1992) suggest that there are six basic emotions which are universal among different cultures, namely happiness, surprise, fear, sadness, anger and disgust. These emotions are associated with specific facial expressions (Ekman and Keltner, 1970; Ekman, 1992). Facial expressions also play a major role in communication of feelings and attitudes, among other important cues such as postures, gestures, verbal and vocal expressions (Mehrabian, 1971).

There have been many attempts to automate the analysis of facial expressions, one of the earliest being Suwa *et al.* (1978), who tracked the motion of facial landmarks for comparison with prototypical motion patterns for different facial expressions (Samal and Iyengar, 1992). Since the 1990s, the field has become very active (Samal and Iyengar, 1992; Cohn *et al.*, 1997; Lien *et al.*, 1997, 1998; Bartlett *et al.*, 1999) and many data sets containing facial images have been released (e.g. Kanade *et al.*, 2000; Whitehill *et al.*, 2009; Lucey *et al.*, 2010; Dhall *et al.*, 2012). Improvements in this field enable systems to assist human experts by extracting task-relevant information from data (Gratch *et al.*, 2014). More specifically, automated emotion recognition systems have applications in health care, e.g. detection of psychological distress (Scherer *et al.*, 2013), in education by estimating student engagement (Shen *et al.*, 2009), and in gaming for improvement of the players' experience (Shaker *et al.*, 2011).

One of the main challenges in emotion recognition as in most computer vision tasks is to deal with the complexity of real-world scenarios. This includes large variations of illumination, appearance of subjects and sensor properties, among many other parameters. With the success of *deep-learning* methods (Krizhevsky *et al.*, 2012; Bahdanau *et al.*, 2014) that learn hierarchical representations from data, the demand for large-scale data sets (Deng *et al.*, 2009) is increased. One very important challenge is to explore these techniques in detail for emotion recognition tasks where the amount of annotated close-to-real-world data is limited compared to object recognition tasks. Another aspect to study is how to benefit from complementary information from different modalities, such as the visual and auditory streams of videos. From the artificial intelligence perspective, it would be interesting to see if building an integrated system, such as a neural network architecture that can be trained as one system from input to predictions, is feasible.

Each of the presented articles is a step towards building an end-to-end trainable system for facial analysis and emotion recognition. These article are preceded by an initial study which presents a hand-engineered feature-extraction pipeline for facial expression recognition. In contrast, the first article employs feature-learning methods. The second article further introduces a data-driven approach for aggregation of sequential information, which is shown to be useful for video processing. In the last article, attention mechanisms are used to automate the initial step of detecting a region of interest.

This chapter provides background on the basic concepts this work is based on. This is followed by Chapter 2, containing an additional literature review and Chapter 3, presenting a guide to the contributions. Chapter 4 provides an initial study of hand-engineered features for emotion recognition. The three articles are then presented in Chapters 5, 6 and 7. Finally, Chapters 8 and 9 discuss the results of the presented work as a whole and draw conclusions.

1.1 Features in Computer Vision

Computer vision deals with analyzing and understanding images or sequences of images. Images are typically stored in 2D arrays of pixel values. In the context of this thesis pixel values are either a single scalar for grayscale or a triple of red, green and blue intensities. Representing an image with pixels is not efficient as the dimensionality is very high and pixel values are highly redundant. In most images, there is a high probability of neighboring pixels having similar values (Attneave, 1954; Hyvärinen *et al.*, 2009).

The goal in computer vision tasks is to define a mapping from images to a description of image content. Defining such a mapping is not trivial and consists of different steps. As a first step, it is common to represent an image in terms of *features*. A feature is a function of an image that explicitly and compactly exploits the structure to simplify further processing (Hyvärinen *et al.*, 2009). An example of features is a set of local edges extracted from an image. Computer vision research invented many types of feature descriptors for different tasks. Some examples are the Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), Gabor filters (Fogel and Sagi, 1989), Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005), Local Binary Patterns (LBPs) (Ojala *et al.*, 1996) and Haar-like features (Viola and Jones, 2001). Haar-like features and LBPs have been successfully applied to face-related tasks such as face detection (Viola and Jones, 2001), facial expression analysis (Shan *et al.*, 2009; Kahou *et al.*, 2014) and face verification (Chen *et al.*, 2013). A Gabor filter is a combination of a sinusoidal wave pattern with a Gaussian window, resulting in a local edge detector. They have been shown to be a good model for the receptive fields of simple cells in the visual cortex of mammalian brains (Jones and Palmer, 1987).

The above mentioned features are hand-crafted and often have to be selected for specific tasks. As this can require a lot of time and effort, automated learning of features from data has received increased interest. Since Krizhevsky *et al.* (2012) won the Large Scale Visual Recognition Challenge (ILSVRC) (Deng *et al.*, 2009) by a large margin using deep Convolutional Neural Networks (CNNs), these architectures are widely investigated in all areas of computer vision research. The general idea behind *deep learning* is to learn a hierarchy of features where high-level features are composed of lower-level features. For instance, the lowest-level features could be edge detectors, the second level could detect groups of edges and higher in hierarchy, filters might resemble facial features (Khorrami *et al.*, 2015). Interestingly, the lowest-level features in CNNs usually look Gabor-like when trained on natural image data. An example of CNN low-level features learned on facial expression images is shown in Figure 1.1. Details on CNN methods are described in Section 1.2.10.

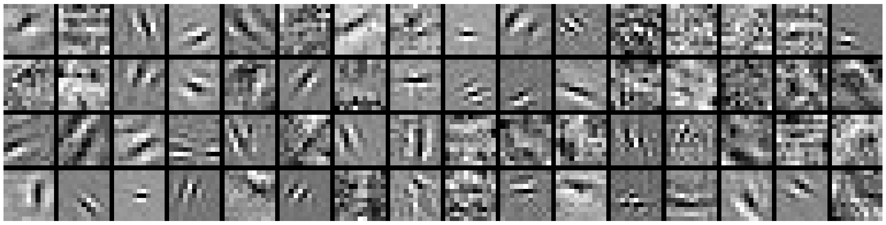


Figure 1.1 Filters learned on a data set of facial expression images.

1.2 Machine Learning

It is not always straightforward to explicitly design a computer program that can solve a complicated task. However, there are families of algorithms, that can adapt a model automatically given data samples. The term machine learning refers to the study of such algorithms for recognizing patterns in data and using them for making decisions and/or predictions. There are three types of learning approaches, namely, *supervised*, *unsupervised* and *reinforcement learning* (Murphy, 2012).

Methods applied in this thesis fall into the supervised and unsupervised categories. For an introduction to reinforcement learning, see Sutton and Barto (1998).

1.2.1 Basic concepts

Supervised learning

In supervised learning the objective is to estimate a function $f : \mathbf{x} \to y$ that maps inputs $\mathbf{x}$ to target values y, given a set of N training examples $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. If the target values are categorical, the task is called classification. One example of classification is emotion recognition, where we map an input image to one of the seven basic emotion labels. Supervised learning with continuous target values is referred to as regression. An example of regression is to predict the bounding box coordinates in an object detection task. In both cases the target values can be multi-dimensional. Supervised learning methods that are used in this thesis are Support Vector Machines (SVMs) (Cortes and Vapnik, 1995), Local Fisher Discriminant Analysis (LFDA) (Sugiyama, 2007), Logistic Regression and Neural Networks. The basics of Logistic Regression, SVMs and Neural Networks are covered in this chapter (See Sections 1.2.5, 1.2.6 and 1.2.7). LFDA is described in Section 4.3.1.

Unsupervised learning

Unsupervised learning is a family of machine learning methods that discover patterns in unannotated data. The tasks in unsupervised learning can usually be interpreted as density estimation where the objective is to find a good model of the data distribution $p(\mathbf{x})$ (Murphy, 2012). Applications of unsupervised learning are dimensionality reduction, clustering and feature learning. Throughout this thesis, the following methods are used:

- Principal Component Analysis (PCA) is used for dimensionality and noise reduction,
- k-means for feature learning by clustering the data,
- auto-encoders for feature learning.

Short introductions to these methods are given in Sections 1.2.3, 1.2.4 and 1.2.9, respectively.

Optimization

In machine learning, the aim is usually to find optimal parameters θ^* of a function or *model* f_θ that minimize a cost function $J(\theta)$. Formally, we are looking for an optimal parameter setting

$$\theta^* = \operatorname*{argmin}_\theta J(\theta). \tag{1.1}$$

The cost function $J(\theta)$ is often defined as the *empirical risk*. For supervised learning, the *empirical risk* takes the form

$$\hat{R}(f_\theta, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, f_\theta(\mathbf{x}_i)), \tag{1.2}$$

which is the mean per-sample value of loss function L, where $(x_i, y_i) \in \mathcal{D}$ is an input-label pair of given data set $\mathcal{D}$. Common loss functions for supervised learning are the 0-1 loss

$$L(y, f_\theta(\mathbf{x})) = \begin{cases} 1, & \text{if } y \neq f_\theta(\mathbf{x}) \\ 0, & \text{otherwise} \end{cases} \tag{1.3}$$

and the squared error

$$L(y, f_\theta(\mathbf{x})) = (y - f_\theta(\mathbf{x}))^2. \tag{1.4}$$

In unsupervised learning, there is no output and y is replaced with x in the loss function. For instance, in unsupervised dimensionality reduction, the objective is to minimize the reconstruction loss (Murphy, 2012):

$$L(x, f_\theta(\mathbf{x})) = (x - f_\theta(\mathbf{x}))^2 \tag{1.5}$$

It is also common to maximize the likelihood of data when dealing with probabilistic models. In this case, the optimization problem can be written as

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmax}} \left[P(x_{1,\ldots,n} | \theta) \right] \\ &= \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{n} P(x_i | \theta) \right], \end{aligned} \tag{1.6}$$

where P is the probability of x given the vector of parameters θ and we assume the data samples are independent (Prince, 2012). Alternatively, one can minimize the *cross entropy* between two distributions which corresponds to the negative of the log of the likelihood:

$$H(P_{\mathit{true}}, P_{\mathit{estimated}}) = -\sum_{x} P_{\mathit{true}}(x) \log P_{\mathit{estimated}}(x). \tag{1.7}$$

It is more convenient to optimize this log-likelihood, as one can compute the derivative of each part of the sum separately. As an example, in binary classification problems, the cross entropy can be written as:

$$H(P_{\mathit{true}}, P_{\mathit{estimated}}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}). \tag{1.8}$$

where y is the ground truth label (0 or 1) and $\hat{y}$ is the predicted probability.

If the objective function has no closed-form solution, one has to resort to iterative methods such as gradient descent (see Section 1.2.2).

Generalization

Optimization methods allow us to learn a model given data samples. However, a complex model can achieve perfect performance on these samples by simply memorizing them. The term *generalization* refers to a model's capability to perform well on unseen data. It is common that a model has high performance on the data used in optimization and low performance on unseen data. This is known as *overfitting* problem.

To estimate the true error of a model, the data set is divided into three disjoint subsets, the training, validation and test sets. The training set is used in optimization, the validation set is for *hyperparameter* selection and the test set is for reporting the algorithm's error. A predefined test set allows us to benchmark different algorithms.

Hyperparameters control the representational power of a model and have to be tuned carefully to avoid overfitting. A typical learning curve is shown in Figure 1.2. It can be seen that for good generalization, the learning should be stopped at the iteration, for which the validation error is minimal as the model is likely overfitting after that. This method is called *early stopping*, where the number of training iterations is a hyperparameter of the model.

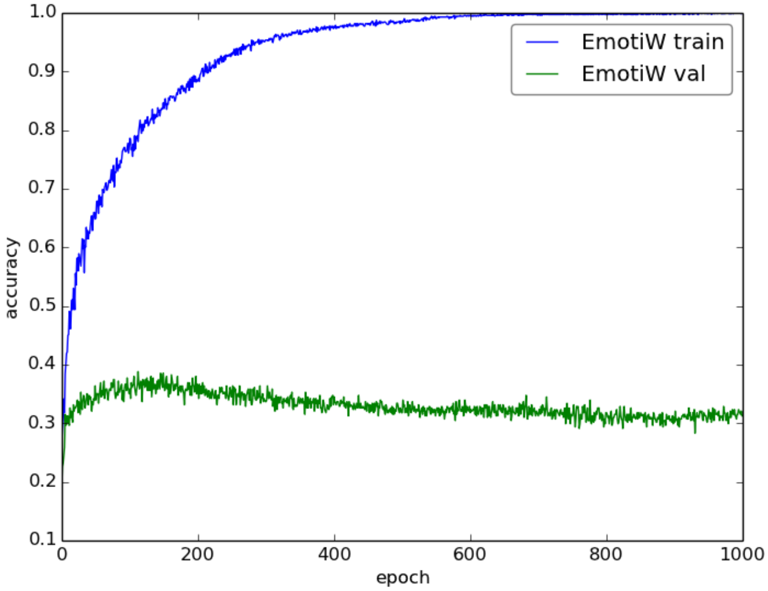


Figure 1.2 A learning curve showing training and validation classification accuracies over iterations (epochs). While the training accuracy consistently increases up to perfect classification, the validation accuracy drops at some point. The model is a neural network trained on emotion recognition videos.

For hyperparameter tuning, *cross validation* is used. It consists of the following steps:

1. Train the model with different hyperparameter settings.
2. Compute the model's validation error for each setting.
3. Report the test set error for the setting with the lowest validation error.

This process can be repeated multiple times with different training-validation splits to get a better estimate of the performance.

1.2.2 Gradient-based optimization

The cost $J(\theta)$ defines a surface by assigning a scalar cost to each value of θ. An example of such a surface is shown in Figure 1.3 on the left. In optimization the goal is to find a global minimum or sometimes a "good" local minimum. At such a minimum the partial derivatives of the cost function with respect to each parameter are zero. In practice many algorithms don't have a closed-form solution. Gradient-based optimization methods can solve this problem iteratively. The gradient is a vector containing the partial derivatives with respect to parameters:

$$\nabla \mathbf{J}(\theta) = \frac{\partial J(\theta)}{\partial \theta}(\theta) = \begin{pmatrix} \frac{\partial J(\theta)}{\partial \theta_1}(\theta) \\ \frac{\partial J(\theta)}{\partial \theta_2}(\theta) \\ \vdots \\ \frac{\partial J(\theta)}{\partial \theta_m}(\theta) \end{pmatrix} \tag{1.9}$$

The gradient at any point in parameter space points towards the direction of steepest ascent on the cost surface. The idea behind the gradient descent method is to take steps in the opposite direction to get closer to a minimum as shown in Figure 1.3 on the right. Formally, the gradient descent update rule can be written as

$$\theta \leftarrow \theta - \eta \nabla \mathbf{J}(\theta), \tag{1.10}$$

where η is a scalar hyperparameter called the *learning rate*. The learning rate controls the velocity by scaling the magnitude of the gradient. If the gradient is computed on the whole training set, the parameters are updated only once for each pass through the data (an *epoch*). A good approximation of this gradient is usually enough and for large training sets the update rate grows and results in faster convergence. This approximation is usually done by computing gradients on so-called *mini-batches*, which contain a small random subset of training samples. This method is known as Stochastic Gradient Descent (SGD). The number of samples in a mini-batch is a hyperparameter.

The approximation of the gradient can be noisy and contain many oscillations around optimal direction in parameter space. One method to reduce these oscillations and increase convergence speed is *momentum* (Polyak, 1964; Sutskever *et al.*, 2013). The update rule with momentum is the following:

$$\mathbf{v} \leftarrow \mu\mathbf{v} - \eta\nabla\mathbf{J}(\theta) \tag{1.11}$$

$$\theta \leftarrow \theta + \mathbf{v} \tag{1.12}$$

where $\mathbf{v}$ is a velocity term that accumulates gradients and μ is the decay factor that regulates the influence of updates from previous iterations.

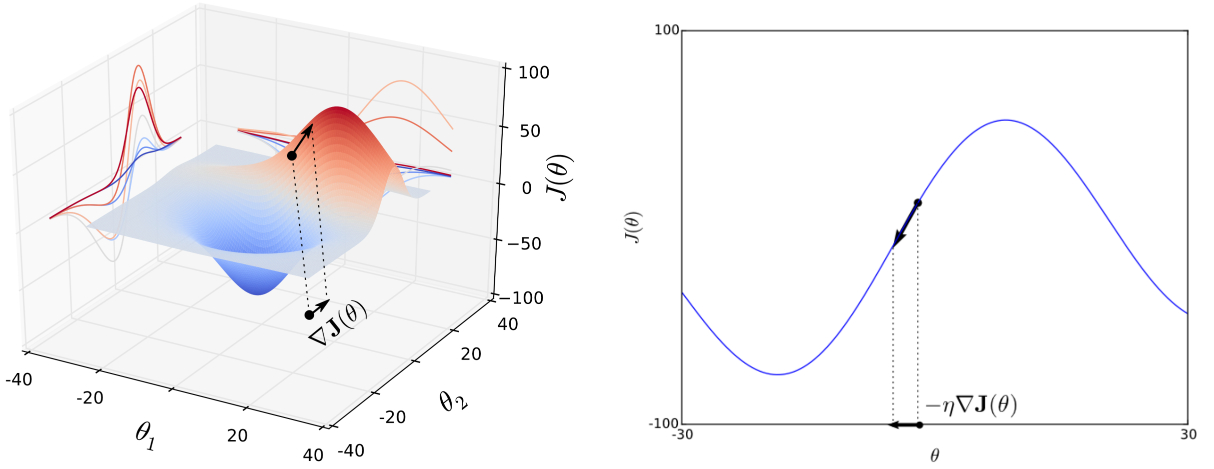


Figure 1.3 Left: A cost surface $J(\theta)$. The gradient computed at one point in parameter space is indicated by $\nabla\mathbf{J}(\theta)$. Right: The gradient descent method takes one step in the direction of steepest descent as indicated by $-\eta\nabla J(\theta)$.

1.2.3 Principal component analysis

Principal Component Analysis (PCA) is a method that is useful in dimensionality reduction, feature extraction and visualization (Bishop, 2006). PCA projects data to a linear subspace, where the coordinate axes correspond to the principal axes of the variation of the data. This projection is a rotation applied to mean-centered data. Given n d-dimensional data points

$\mathbf{x}_i, i = 1, ..., n$, we first subtract the mean vector $\overline{\mathbf{x}}$:

$$\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{1.13}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \overline{\mathbf{x}} \tag{1.14}$$

The transformation can be written as

$$z_{ij} = \mathbf{x}_i^{\mathrm{T}}\mathbf{u}_j, \tag{1.15}$$

where $\mathbf{u}_j$ is the j-th basis vector of the subspace, corresponding to the j-th principal component and z_{ij} is the j-th coordinate of the i-th data point $\mathbf{x}_i$ in that subspace. The $\mathbf{u}_j, j = 1, ..., d$ form an orthonormal basis, i.e.

$$\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}, \tag{1.16}$$

where $\mathbf{U}$ is the matrix with columns $\mathbf{u}_1, ..., \mathbf{u}_d$ and $\mathbf{I}$ is the d-dimensional identity matrix. We can compute the reconstruction $\tilde{\mathbf{x}}_i$ of original vector $\mathbf{x}_i$:

$$\tilde{\mathbf{x}}_i = \sum_{j=1}^{d} z_{ij}\mathbf{u}_j = \sum_{j=1}^{d}(\mathbf{x}_i^{\mathrm{T}}\mathbf{u}_j)\mathbf{u}_j \tag{1.17}$$

To determine the basis vector $\mathbf{u}_j$, one can minimize the reconstruction error (Kokiopoulou *et al.*, 2011):

$$J = \frac{1}{n}\sum_{i=1}^{n}||\mathbf{x}_i - \tilde{\mathbf{x}}_i||^2 = ||\mathbf{X}^{\mathrm{T}} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}||_F^2 \tag{1.18}$$

$$= \mathrm{tr}\left((\mathbf{X}^{\mathrm{T}} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}})^{\mathrm{T}}(\mathbf{X}^{\mathrm{T}} - \mathbf{U}\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}})\right) \tag{1.19}$$

$$= \underbrace{\mathrm{tr}\left(\mathbf{X}\mathbf{X}^{\mathrm{T}}\right)}_{\text{constant w.r.t. } \mathbf{U}} - \mathrm{tr}\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{U}\right), \tag{1.20}$$

where $\mathbf{X}$ is the matrix which contains the data points $\mathbf{x}_1, ..., \mathbf{x}_n$ in its rows and $||.||_F^2$ is the squared Frobenius norm

$$||\mathbf{A}||_F^2 = \sum_{i=1}^{M}\sum_{j=1}^{N}|a_{ij}|^2. \tag{1.21}$$

Minimization of Equation 1.20 w.r.t. $\mathbf{U}$ is equivalent to:

$$\underset{\mathbf{U}}{\mathrm{argmax}}\left[\mathrm{tr}\left(\mathbf{U}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{U}\right)\right] \tag{1.22}$$

The solution to this equation under the constraint in Equation 1.16, is given by the eigendecomposition of the data covariance matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$. The columns of $\mathbf{U}$ are the eigenvectors, sorted by the corresponding eigenvalues in descending order. The transformation is computed by Equation 1.15. In dimensionality reduction, we only keep the first m principal components.

1.2.4 K-means

K-means is an unsupervised learning algorithm for discovering clusters in data. K is a hyperparameter that specifies the number of clusters. The algorithm consists of the following steps:

1. Initialize K cluster centers or centroids $\mu_1, ..., \mu_K$.
2. Compute distances for each data point to the K centroids.
3. Assign the data point to the cluster with minimum distance.
4. Update the centroids by setting them equal to the mean of data points assigned to the corresponding cluster.
5. Recompute distances to the updated centroids.
6. If any data point is not assigned to the nearest cluster, repeat from step 3.

The centroids can be initialized randomly, for instance by averaging a few data samples for each cluster. The distance metric used in the experiments of this thesis is the Euclidean distance.

1.2.5 Logistic regression

Logistic regression is a probabilistic model used for classification. It is defined by the following equation:

$$p(Y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^{\mathrm{T}}\mathbf{x})}{\sum\limits_{j=1}^{c} \exp(\mathbf{w}_j^{\mathrm{T}}\mathbf{x})}, \tag{1.23}$$

where $p(Y = k|\mathbf{x})$ is the predicted probability that input vector $\mathbf{x}$ belongs to class k, $\mathbf{w}_k$ are the parameters of class k and the number of classes is c. In order to estimate the parameters, we minimize the negative log-likelihood:

$$\mathcal{L}(\mathbf{W}) = -\sum_{i=1}^{n} \log \left(\frac{\exp(\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i)}{\sum\limits_{j=1}^{c} \exp(\mathbf{w}_{y_j}^{\mathrm{T}} \mathbf{x}_i)} \right) \tag{1.24}$$

$$= -\sum_{i=1}^{n} \left(\mathbf{w}_{y_i}^{\mathrm{T}} \mathbf{x}_i - \log \sum_{j=1}^{c} \exp(\mathbf{w}_{y_j}^{\mathrm{T}} \mathbf{x}_i) \right), \tag{1.25}$$

where y_i is the true class index of the i-th data point. To fit the logistic regression model, we can use gradient descent. The gradient of the negative log-likelihood w.r.t. weight vector $\mathbf{w}_k$ is:

$$\frac{\partial \mathcal{L}(W)}{\partial \mathbf{w}_k} = -\sum_{i=1}^{n} \left(I_{\{y_i = y_k\}} - p(y_k | x_i) \right) \mathbf{x}_i, \tag{1.26}$$

where

$$I_{\{y_i = y_k\}} = \begin{cases} 1, & \text{if } y_i = y_k \\ 0, & \text{otherwise.} \end{cases} \tag{1.27}$$

To prevent overfitting, $L2$ regularization is often added to the cost function:

$$\mathcal{L}_\lambda(\mathbf{W}) = \mathcal{L}(\mathbf{W}) + \lambda ||\mathbf{W}||_F^2, \tag{1.28}$$

where λ is a hyperparameter that controls the regularization strength.

1.2.6 Support vector machines

Support Vector Machines (SVMs) are popular supervised learning models that are used for classification. They are linear classifiers, i.e. they find a hyperplane which separates two classes. The geometric motivation behind SVMs is to place the hyperplane in such a way, that the margin is maximized as shown in Figure 1.4. The margin is the distance between the hyperplane and the closest data sample. The objective function is

$$J(\mathbf{w}, \lambda) = \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i, \mathbf{w})) + \lambda ||\mathbf{w}||_F^2, \tag{1.29}$$

where $L(.)$ is the 0-1 loss, $f(\mathbf{x}, \mathbf{w})$ is

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \mathbf{x} + w_0, \tag{1.30}$$

and $\lambda||\mathbf{w}||_2^2$ is the regularization term explained in Section 1.2.5. This objective function is hard to optimize due to 0-1 loss. We can alternatively solve the quadratic program

$$\min_{\mathbf{w},w_0} ||\mathbf{w}||_F^2 \text{ subject to } y_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1, \tag{1.31}$$

given that the data is separable. However, if the data is not separable, we have to introduce *slack variables* ξ_i. The ξ_i indicates to what degree a data point is misclassified, i.e. how far it lies on the wrong side of a tube defined by margin and the hyperplane (Murphy, 2012). The quadratic program in this case can be written as

$$\min_{\mathbf{w},w_0,\xi_i} ||\mathbf{w}||_2^2 + C\sum_{i=1}^{n}\xi_i, \tag{1.32}$$

subject to

$$y_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1 - \xi_i,\ \ \xi_i \geq 0. \tag{1.33}$$

In this case, the $L(.)$ in Equation 1.29 is the *hinge-loss*

$$L(y, f(\mathbf{x})) = \max(0, 1 - yf(\mathbf{x})), \tag{1.34}$$

where C is a hyperparameter that expresses the importance of achieving low misclassification rate against regularization. It is possible to show that the optimal solution is a linear combination of input vectors (Schölkopf and Smola, 2002; Murphy, 2012):

$$\hat{\mathbf{w}} = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i,\ \ \alpha_i \geq 0. \tag{1.35}$$

Data points $\mathbf{x}_i$ with $\alpha_i \neq 0$ are the *support vectors.* Given the optimal solution, Equation 1.30 can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i\mathbf{x}_i^{\mathrm{T}}\mathbf{x} + w_0. \tag{1.36}$$

The non-linear version of SVMs is obtained by applying the *kernel trick* to first map data into a richer feature space and then search for a separating hyperplane. This can be efficiently done by using a kernel function $\kappa(\mathbf{x}_i, x)$ instead of $\mathbf{x}_i^{\mathrm{T}}\mathbf{x}$ in Equation 1.36. Two commonly used kernel functions are the polynomial $(\mathbf{x}_i^{\mathrm{T}}\mathbf{x}+1)^d$ with hyperparameter d and Radial Basis Function (RBF) $\exp(-\gamma||\mathbf{x}_i - \mathbf{x}||^2)$ with hyperparameter γ.

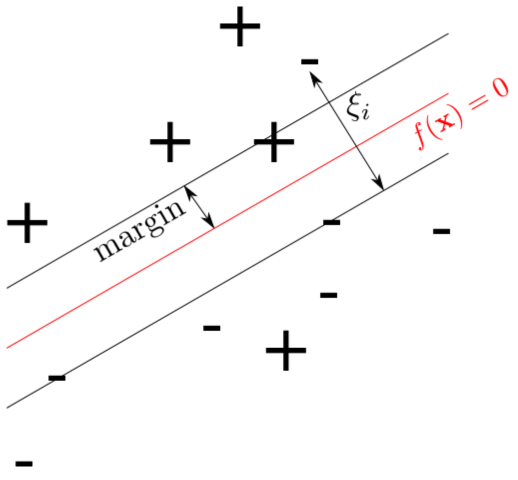


Figure 1.4 An example of binary classification with an SVM. +/- are the data points and the red line is the separating hyperplane.

1.2.7 Neural networks

Neural networks are a family of machine learning models which are inspired by neuroscientific studies (McCulloch and Pitts, 1943; Rosenblatt, 1961; Rumelhart *et al.*, 1988). These models can be visualized as graphs as shown in Figure 1.5 on the left, with nodes representing *neurons* or *units* and edges representing *synapses* or *weights*. The simplest form of neural networks is a feed-forward network which is often called Multi-Layer Perceptron (MLP) due to its similarity to the Perceptron (Rosenblatt, 1961). MLPs have a layered structure, where the output of one layer is the input to the next layer.

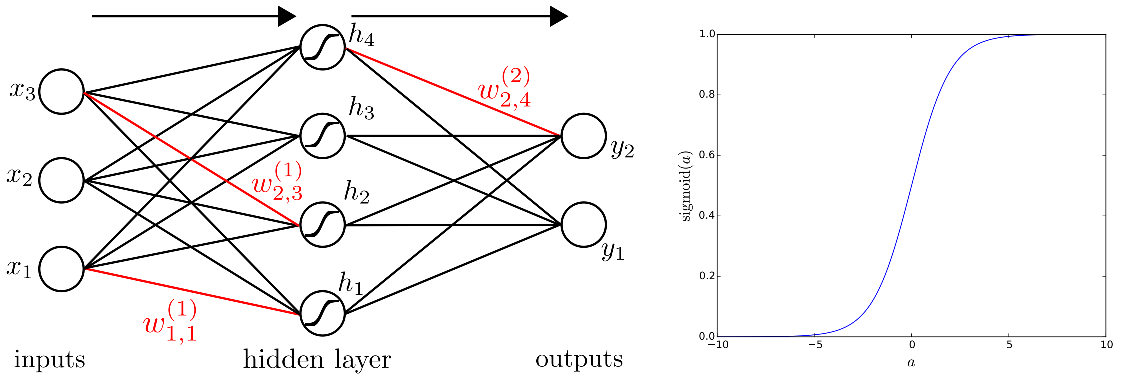


Figure 1.5 Left: A single hidden-layer feed-forward neural network with three input, four hidden and two output units. Example weights are highlighted in red. Right: The logistic sigmoid function.

To be specific, there is an input layer with units $x_1, ..., x_D$, where D is the dimensionality of data points. The output layer has units $y_1, ..., y_K$, where K is the number of classes in case of a classification task. The layers between the input and output layer are called *hidden* layers. The network implements a non-linear transformation of inputs. In this section and Section 1.2.8, the notation is borrowed from Bishop (2006). The units in the first hidden layer compute their output in two steps as follows:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{1.37}$$

$$h_j = \sigma(a_j) \tag{1.38}$$

In Equation 1.37, a_j refers to the pre-activation computation of hidden unit j, $w_{ji}^{(1)}$ is the weight for the connection from input unit i to hidden unit j and $w_{j0}^{(1)}$ is the *bias*. In Equation 1.38, the output or *activation* h_j of hidden unit j is given by a non-linear activation function $\sigma(.)$ applied to a_j. Popular activation functions are the logistic sigmoid $\sigma(a) = \frac{1}{1+\exp(-a)}$ shown in Figure 1.5 on the right, the hyperbolic tangent and the Rectified Linear Unit (ReLU) function $\mathrm{ReLU}(a) = \max(0, a)$. The output layer units perform the following computation given the last hidden layer's outputs h_j:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} h_j + w_{k0}^{(2)} \tag{1.39}$$

Depending on the task, an activation function can also be applied to the output units. In multi-class classification, it is common to use the *softmax* function:

$$y_k = \mathrm{softmax}_k(a) = \frac{\exp(a_k)}{\sum\limits_{i=1}^{K} \exp(a_i)} \tag{1.40}$$

The combination of Equation 1.39 and Equation 1.40 is equivalent to the logistic regression computation in Equation 1.23 applied to the output of the hidden layer. Here, only a single hidden layer is described. However, in general there can be many hidden layers in an MLP and there are many more complex neural network architectures.

1.2.8 Backpropagation

Before presenting other neural network architectures, we explain how to train a simple neural network by gradient-based optimization. In order to use the gradient descent method as described in Section 1.2.2, one has to compute the gradient of the cost function w.r.t. parameters. In neural networks, the parameters are weights and biases of all layers in the network, and the gradient is usually computed using the *backpropagation* algorithm (Werbos, 1974; Bryson, 1975; Rumelhart *et al.*, 1988). We consider the cost function $J(w)$ to be the sum over squared errors for all training samples

$$J(\mathbf{w}) = \sum_{n=1}^{N} J_n(\mathbf{w}) = \sum_{n=1}^{N} \frac{1}{2} \sum_{k} (y_{nk} - t_{nk})^2, \tag{1.41}$$

where $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$ is the kth output for the nth training input $\mathbf{x}_n$ and t_{nk} is the corresponding target value. For simplicity, we describe the derivations for one sample using a single hidden-layer MLP as explained in Section 1.2.7. Bias terms are omitted in the following equations as it is possible to include them in the weight matrices as 0-th column and add a constant one to the 0 index of layer inputs. The general computation of the MLP can be written as:

$$a_j = \sum_i w_{ji} z_i \tag{1.42}$$

$$z_j = h(a_j) \tag{1.43}$$

with z_i denoting the previous layer outputs, a_j the pre-activations and z_j the layer outputs. Backpropagation consists of the application of the chain rule of differentiation. An initial *forward pass* that computes the activations of each layer and the value of the cost function is followed by a *backward pass* which computes partial derivatives in a layer-wise fashion by applying the chain rule. To compute the partial derivative of the cost function J_n w.r.t. a parameter w_{ji}, the chain rule is as follows:

$$\frac{\partial J_n}{\partial w_{ji}} = \frac{\partial J_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}} \tag{1.44}$$

As in Bishop (2006), we adopt a short notation

$$\delta_j = \frac{\partial J_n}{\partial a_j}. \tag{1.45}$$

From Equation 1.42, we see that $\frac{\partial a_j}{\partial w_{ji}} = z_i$. Therefore,

$$\frac{\partial J_n}{\partial w_{ji}} = \delta_j z_i. \tag{1.46}$$

Given the squared error cost function in Equation 1.41, the δs for the output layer are

$$\delta_k = y_k - t_k. \tag{1.47}$$

Given the chain rule, we can derive the computation of

$$\delta_j = \frac{\partial J_n}{\partial a_j} = \sum_k \frac{\partial J_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} = h'(a_j) \sum_k w_{kj} \delta_k, \tag{1.48}$$

which shows how to compute the δ_j of a unit j as a weighted sum of the next layer's δs multiplied by the derivative of the activation function of unit j.

To summarize, the backpropagation algorithm first computes activations and the cost in the forward pass. Then the δs of the output units are computed according to the selected cost function. In the backward pass, we iteratively update the δs of each layer and evaluate the partial derivatives w.r.t. parameters, which are then used for optimization.

1.2.9 Auto-encoders

Auto-encoders are feed-forward neural networks for unsupervised learning. They map the inputs $\mathbf{x}$ into a code $\mathbf{h}$ in the *encoding* phase and map the code back to the reconstruction $\tilde{\mathbf{x}}$ of the inputs in the *decoding* phase. During optimization the reconstruction error is minimized. Hinton and Salakhutdinov (2006) introduced auto-encoders for dimensionality reduction, where the authors built a deep auto-encoder with seven hidden layers. The encoder consists of the first half of the network with the fourth layer being the code layer and the remaining layers being the decoder which outputs the reconstruction. The weights are shared between the encoder and the decoder. The decoder uses the transposed weights of the encoder in reverse order. It is obvious that such a network can achieve perfect reconstruction by learning an identity mapping. For this reason Hinton and Salakhutdinov constrained the size of the code layer to be smaller than the input layer to add a "bottleneck".

Later, Vincent *et al.* (2010) presented another way of training an auto-encoder by adding noise to the inputs. The objective was still the reconstruction of the original uncorrupted inputs, so the model learns to denoise inputs. The *denoising auto-encoder* can learn higher dimensional codes than the input dimension.

1.2.10 Convolutional neural networks

Convolutional Neural Networks (CNNs) (Fukushima, 1980; LeCun *et al.*, 1998) are a type of neural networks that have been successfully used to achieve state-of-the-art performance in multiple vision tasks (Krizhevsky *et al.*, 2012; Neverova *et al.*, 2014). The motivation behind CNNs is the observation that the visual cortex has cells that respond only to stimuli in sub-regions of the visual field (Hubel and Wiesel, 1962). There are multiple cells detecting the same type of stimulus. Each of them only responds to a different sub-region, together covering the whole visual field. CNNs behave similarly by detecting the same *local* patterns, using *shared* weights, all over the input image. Therefore, comparing to MLPs, they have *sparse* connections. Mathematically, one-dimensional convolution is defined as

$$s(t) = (x * w)(t) = \int x(\tau)w(t - \tau)\, d\tau, \tag{1.49}$$

where x is a signal, w is a filter or kernel and s is the output signal (Bengio *et al.*, 2015). For temporal signals, t denotes time. Digital images can be considered as discrete two-dimensional signals expressing pixel value as a function of spatial coordinates. For such signals, convolution is formulated as

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n), \tag{1.50}$$

where I is an image, K is a two-dimensional kernel and S is the resulting *feature map*. The region of non-zero elements of K is usually much smaller than the image size. In implementation, the kernel only contains this region and is applied by sliding it across the image and computing the filter responses at each location. The image and kernel can contain multiple channels, in which case the convolution is applied per channel and the results are summed over channels.

A convolutional layer in a CNN consists of multiple of these kernels which are applied separately to the input. The output is a multi-channel feature map where the number of channels corresponds to the number of kernels. This is visualized in Figure 1.6 on the left. As in an MLP layer, one can apply an activation function to the output. The output can be fed to another convolutional layer or fully-connected MLP layer. In case the next layer is fully-connected, the output is reshaped into a single vector.

One property resulting from applying the same kernel to all positions in the input is *translation equivariance*. This means any translation in input space results in a similar translation in feature space.

It is common that some convolutional layers are followed by a *pooling* layer. Pooling layers summarize local regions of a feature map by replacing them with statistics such as the average or maximum value of that region. This operation is applied separately to each channel and it can be useful for two main reasons:

- The size of the resulting feature map can become smaller if the row and/or column offsets (*strides*) between neighboring regions are larger than one. An example of max-pooling is shown in Figure 1.6 on the right.

- Pooling adds *invariance* to small translations which helps in recognition tasks. However, if the task is to locate an object within an image, pooling can decrease performance due to loss in resolution.

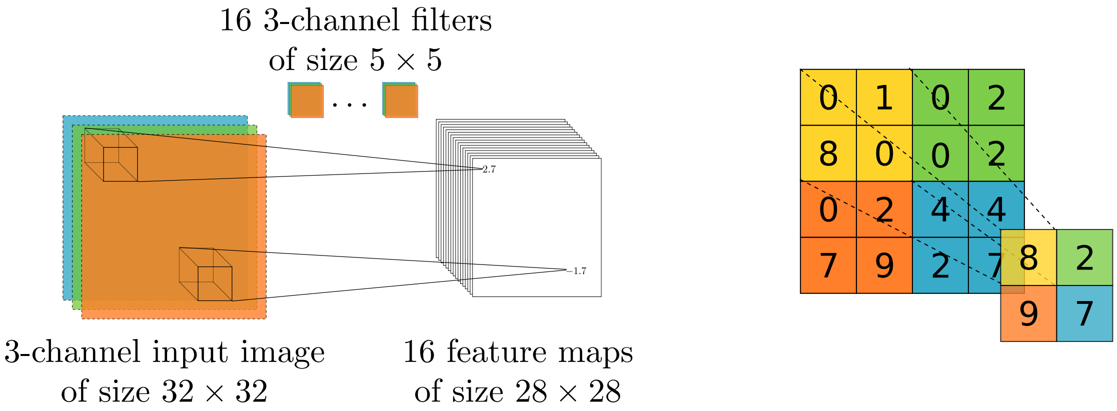


Figure 1.6 Left: A convolutional layer that takes a three-channel (RGB) image as input and applies a filter bank of size 16 × 3 × 5 × 5 yielding 16 feature maps of size 28 × 28. Right: Two-by-two maxpooling with non-overlapping pooling regions.

There are many software libraries offering efficient implementations of convolution and pooling operations. Many deep-learning frameworks use the cuDNN library for fast computation of convolutions (Chetlur *et al.*, 2014).

1.2.11 Recurrent neural networks and their optimization

When dealing with sequential data, such as video streams, one approach in learning is to treat the whole sequence as one large input. This approach does not work for varying-length sequences. As an alternative, one can learn a model on video segments and later pool over the outputs of segments for each video. This approach ignores dependencies between segments of each video. Recurrent Neural Networks (RNNs) are a group of neural networks that address these issues by adding loop connections in the network graph.

An RNN reads a sequence of inputs $\mathbf{x}_1, \dots, \mathbf{x}_t$ frame-by-frame and at each time step t updates an internal hidden state according to

$$\mathbf{h}_t = \sigma(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{rec}\mathbf{h}_{t-1}), \tag{1.51}$$

where $\mathbf{x}_t, \mathbf{h}_t$ are the input and hidden state at time step t, $\mathbf{W}_{in}$ is the input weight matrix, $\mathbf{W}_{rec}$ are the recurrent weights going from $\mathbf{h}_{t-1}$ to $\mathbf{h}_t$ and $\sigma(.)$ is the hidden activation function. Usually, $\mathbf{h}_0$ is initialized with zeros. The outputs at each time step are computed as

$$\mathbf{y}_t = f(\mathbf{W}_{out}\mathbf{h}_t), \tag{1.52}$$

where $\mathbf{W}_{out}$ contains the hidden-to-output weights and $f(.)$ is the activation function of the output layer. An example of an RNN graph is shown in Figure 1.7.

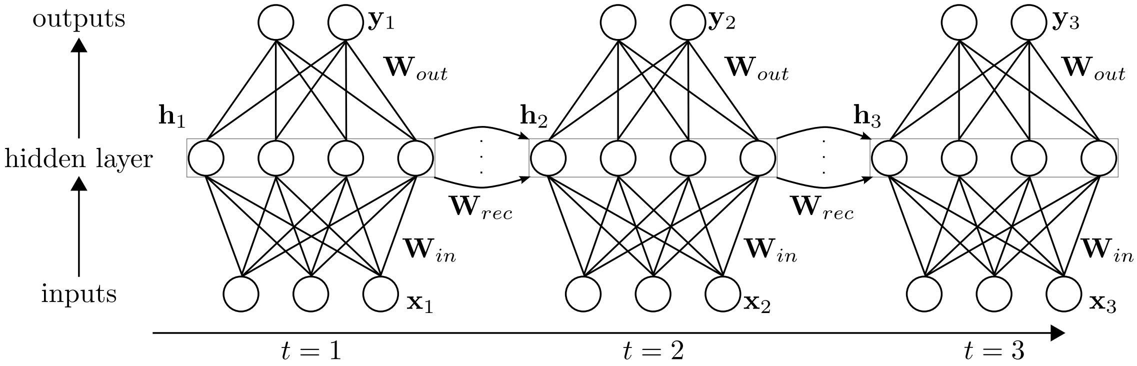


Figure 1.7 A simple example of an unrolled RNN graph with three input, four hidden and two output units. The figure is reproduced with permission from Michalski (2013).

RNNs can be trained using the Backpropagation Through Time (BPTT) algorithm (Werbos, 1990). The weights in unrolled graph depicted in Figure 1.7 are shared through time steps. One can define a cost term at each output $\mathbf{y}_1, \ldots, \mathbf{y}_t$ and backpropagate each of them separately. In our simple example of an RNN the gradient flows downward (downward through layers) and to the left (backward in time). Partial derivative for each cost term w.r.t. each copy of a weight matrix are computed. The partial derivatives of the total cost are then the sum of all partial derivatives summed over cost terms and copies.

The update in Equation 1.51 which is applied at each time step can result in highly non-linear dynamics. Even though this gives the RNN high representational power, it can cause problems in optimization due to *vanishing* and *exploding gradients* (Hochreiter, 1991; Hochreiter *et al.*, 2001). These terms refer to the potential of rapid shrinking or growing of the gradient's norm as we repeatedly apply linear and non-linear transformations during BPTT. Both of these phenomena prevent learning of long-term dependencies. Hochreiter and Schmidhuber (1997) introduced the Long Short-Term Memory (LSTM) network, a more sophisticated RNN which addresses these issues. This is however outside of the scope of this thesis as only simple RNNs are used in the last two articles.

To prevent exploding gradients one can use *gradient clipping* (Pascanu *et al.*, 2012). Gradient clipping scales the gradient norm down to a threshold value T if the norm is greater than T by inserting the following step before the update of SGD described in Section 1.2.2:

$$\nabla\mathbf{J}(\theta) \leftarrow \begin{cases} T\frac{\nabla\mathbf{J}(\theta)}{||\nabla\mathbf{J}(\theta)||}, & \text{if } ||\nabla\mathbf{J}(\theta)|| > T \\ \nabla\mathbf{J}(\theta), & \text{otherwise} \end{cases} \tag{1.53}$$

CHAPTER 2 LITERATURE REVIEW

Chapters 4 to 7 each provide an overview of related work. This chapter focuses on more general context and includes a review of more recent work.

Facial expressions have been studied for decades (Ekman and Keltner, 1970). Ekman proposed categories of human emotions which are linked to facial expressions that appeared to be universal among cultures (Ekman, 1992). The Facial Action Coding System (FACS) was developed to describe motion in faces in terms of so-called action units (Hjortsjö, 1969; Ekman and Friesen, 1977). An action unit expresses to what degree a specific facial muscle is contracted or relaxed. Facial expressions can be defined as combination of active facial muscles.

According to a survey of Samal and Iyengar, the research on automation of facial expression recognition was not very active before the 1990s (Samal and Iyengar, 1992). Early works of Bartlett *et al.* (1996, 1999) attempted to automate the FACS annotation, exploring various representations based on holistic analysis, wrinkles and motion flow fields. A hybrid approach, combining all representations outperformed human non-experts. These results indicated that computer vision methods can simplify this task. Lien *et al.* (1997) built a system using Hidden Markov Models (HMMs) to automate facial expression recognition based on the FACS annotation. As feature input to the HMM, tracked keypoints for the lower part of the face and pixel-wise motion flow for the upper part of the face were explored.

Since then, there have been many advances in the field of face-related tasks, such as face detection (Viola and Jones, 2001; Zhu and Ramanan, 2012; Sun *et al.*, 2013), facial landmark localization (Zhu and Ramanan, 2012; Sun *et al.*, 2013) and face verification (Chopra *et al.*, 2005; Taigman *et al.*, 2014). These advances helped to improve facial expression recognition systems and data set collection (Dhall *et al.*, 2013). The trend in employed data sets seems to go from lab-controlled environments (Lucey *et al.*, 2010; Susskind *et al.*, 2010) to more close-to-real-world settings (GENKI-4K, 2008; Dhall *et al.*, 2012), from static facial expression image data sets (Susskind *et al.*, 2010; GENKI-4K, 2008) to dynamic multi-modal video data sets (Ringeval *et al.*, 2013; Dhall *et al.*, 2013), and from relatively small number of samples (Lucey *et al.*, 2010) to large-scale data sets (Susskind *et al.*, 2010).

There are two main reasons for using larger data sets: the need to cover many factors of variation in realistic scenarios and the benefit of applying data-driven learning approaches. While many approaches to emotion recognition use neural networks only for classification of pre-extracted hand-engineered features, the application of feature-learning methods is becoming more popular. This can largely be attributed to the success of deep learning in multiple domains such as speech recognition (Hinton *et al.*, 2012a), computer vision (Krizhevsky *et al.*, 2012) and sequence modeling (Hochreiter and Schmidhuber, 1997; Sutskever *et al.*, 2014). Features learned by deep neural networks can improve generalization compared with hand-engineered features and are transferable, even across remote tasks (Le *et al.*, 2011; Kahou *et al.*, 2013; Konda *et al.*, 2014b; Yosinski *et al.*, 2014).

Deep learning models such as CNNs, RNNs and Auto-encoders have been successfully applied to emotion recognition. Graves *et al.* used uni-directional and bi-directional RNNs to classify sequences of facial images into emotion categories (Graves *et al.*, 2008). As input to the RNNs, features describing the 3-D shape of a fit face model were used. A discriminative variant of Contractive Auto-encoders (Rifai *et al.*, 2011) was shown to yield good features for emotion recognition on static images (Rifai *et al.*, 2012). Tang trained an CNN, replacing the softmax classification layer with a linear SVM layer (Tang, 2013). This model was state-of-the-art on the Facial Expression Recognition (FER2013) data set (Carrier *et al.*, 2013).

The two most active yearly challenges in emotion recognition are the Audio/Visual Emotion Challenge (AVEC) (Schuller *et al.*, 2011) and the Emotion Recognition in the Wild (EmotiW) challenge (Dhall *et al.*, 2013). One main characteristic of AVEC is that it uses a lab-controlled data set with continuous annotations in the two-dimensional *arousal-valence* space. Arousal and valence are an alternative way of representing the emotional state of a subject. The lab-controlled setting allows for providing additional modalities from physiological sensors. In contrast, the EmotiW challenge uses a data set collected by cropping short video clips from Hollywood movies, each annotated with one of the basic emotion classes or neutral. This data set offers fewer modalities but has significantly more variation in the visual modality.

In the first EmotiW challenge in 2013, Kahou *et al.* trained an ensemble of modality-specific deep-learning models for emotion recognition in videos (Kahou *et al.*, 2013). The ensemble includes an aggregation of per-frame features from an CNN trained on a large data set of static images. This was shown to be a useful way of transferring knowledge about facial expressions from static images to the video domain. A detailed description with an extended experiment section is presented in Chapter 5. This approach inspired other works in emotion recognition, for instance Kim *et al.* (2015); Ng *et al.* (2015).

Convolutional Neural Networks (CNNs) have become a well-established tool for extracting visual features. The work of Khorrami *et al.* showed that CNNs can learn feature detectors which are sensitive to specific facial Action Units even when not explicitly trained to do so (Khorrami *et al.*, 2015). They trained an CNN without biases (Konda *et al.*, 2014a) with basic emotion labels as target values. The results showed that some filters strongly correspond to specific Action Units.

Variations of RNNs are becoming more widely applied to emotion recognition in video. Besides the early work of Graves *et al.* (2008) which is already mentioned, here some other approaches are listed. Wöllmer *et al.* (2010) used bi-directional LSTMs to classify emotion from speech and features describing the 3-D coordinates of 46 facial markers. They compared LSTMs with HMMs and SVM-based classifiers and showed that it outperforms these models. Nicolaou *et al.* (2011) also used bi-directional LSTMs. They use a data set with continuous annotation of arousal and valence values. As input to the LSTM they used audio features and coordinates of facial and shoulder keypoints.

Recurrent Neural Networks (RNNs) have recently been used in the AVEC and EmotiW challenges. For instance, Chao *et al.* (2015) trained an LSTM on features from multiple modalities. For the visual modality, the authors combined four types of features including keypoint-based face representation and features extracted from an CNN trained on face recognition data sets. The input to the LSTM is a nonlinear transformation of the concatenation of temporally pooled modalities. This approach yielded a considerable improvement over classification based on single modality features.

One challenge in the data used in AVEC is to deal with multiple modalities which are not perfectly aligned. He *et al.* (2015), in their winning submission to the 2015 competition, employed multiple LSTM networks. Each sequence of modality-specific features is transformed by a separate deep bi-directional LSTM into a sequence of arousal and valance values which are all fed to another deep bi-directional LSTM after Gaussian smoothing.

Similar to the work on the EmotiW challenge data presented in Chapter 6 (Ebrahimi Kahou *et al.*, 2015), the work of Khorrami *et al.* (2016) on the AVEC challenge applied an RNN on features from a pre-trained CNN. Khorrami *et al.* focus on the visual modality and perform a detailed study of various hyper-parameter settings. In contrast to our work, the authors use a deep RNN.

As the last article presented in Chapter 7 studies the application of attention mechanisms, here we list some early related works which applies such techniques specifically to emotion recognition. One early interesting work by Fragopanagos and Taylor presents an artificial neural network for attention and emotion (Fragopanagos and Taylor, 2005). The proposed architecture involves a feed-back loop that enables the network to weight input features. Larochelle and Hinton (2010) apply an attention mechanism together with an Restricted Boltzmann Machine (RBM) for image classification tasks based on sequential processing of several “glimpses”. Here, a glimpse is a sample of the image that contains high-resolution information of a small local region and low-resolution information of surroundings. A controller is learned that selects glimpse locations from a discrete set on a 7×7 grid. In contrast to differentiable soft-attention mechanisms (Gregor *et al.*, 2015), this approach requires sampling. Among other data sets, Larochelle and Hinton (2010) experimented with a static image facial expression recognition data set. The authors show that this sequential processing of glimpses can be beneficial compared to the single-step processing of the whole image.

CHAPTER 3 OVERVIEW OF RESEARCH GOALS AND CONTRIBUTIONS

Computer vision and machine learning research are advancing rapidly. Deep learning methods in particular have yielded top performing results on several competitive benchmark challenges (Krizhevsky *et al.*, 2012; Simonyan and Zisserman, 2014). One of many important fields of research that can benefit from these advances is the automated analysis of facial expressions and emotion recognition. Traditional methods usually required a lot of parameter tuning or simplifying assumptions that do not reflect realistic conditions. In contrast to most previous works which rely on hand-engineered features, deep learning can be used to learn representations from data. Features learned by deep neural networks have been shown to generalize well to unseen data (Yosinski *et al.*, 2014).

My research objective is to work towards fully automated facial expression analysis and emotion recognition. The long-term goal as in many machine learning projects is to build an end-to-end system, that can learn to solve these tasks given raw data. My work contributes to this endeavor by studying representations and learning approaches that can be useful for facial analysis. The main part of my work focuses upon deep-learning approaches, developed to deal with challenges posed by the recent availability of multi-modal emotion recognition data sets. However, the last part of this thesis focuses on a technical solution to the tracking problem using a recurrent attention mechanism. In the context of facial analysis tasks in videos, detection and tracking of faces are integral components of the pipeline and it is desirable to be able to tune these components along with the recognition part of the pipeline. Having an attention mechanism as part of an end-to-end trainable system would allow to retain task-relevant information in contrast to separately tuned components for detection and tracking. Moreover, many pipelines discard information such as motion of head and facial features, which can be informative in emotion recognition or analysis of human behaviour in general.

The initial study in Chapter 4 is based on a workshop paper "Facial Expression Analysis Based on High Dimensional Binary Features" (Kahou *et al.*, 2014). It proposes a new variant of LBPs for the tasks of expression recognition on the Extended Cohn-Kanade (CK+) data set (Lucey *et al.*, 2010) and smile detection on the GENKI data set (GENKI-4K, 2008). The most common approach in building image representations based on LBPs is to use histogramming of patterns within local regions of an image. In contrast, this work skips the step of histogramming and uses a binary vector of patterns. Inspired by Chen *et al.* (2013), patches of different scales are cropped around facial keypoints and the whole face is represented by high dimensional binary features. A pipeline of dimensionality reduction methods and an SVM classifier is then used to predict facial expressions from this representation. This approach yielded state-of-the-art performance at the time. While this work includes experiments on the CK+ data set which contains image sequences, only the peak image is used in these experiments and the dynamic content is ignored. This method also relies on pre-extracted facial keypoints, however, it is shown to be robust to noisy keypoints. As the lead author, I was responsible for experimentation and validation of the proposed methodology. This included data preparation, programming of the pipeline and experimental setup. I also contributed to the writing and editing of the document including bibliographic research.

The first article "EmoNets: Multimodal Deep Learning Approaches for Emotion Recognition in Video" (Kahou *et al.*, 2015a) presents deep learning approaches for emotion recognition from video clips and provides an analysis of different ways of model fusion. The EmotiW challenge (Dhall *et al.*, 2013) consists of short videos extracted from Hollywood movies, which present more realistic scenarios in comparison to data sets collected in controlled lab environments. The data set introduces large variations, such as backgrounds, lighting conditions, and human poses. The aim of the challenge is to develop models for different modalities, i.e. video and audio and finding a way of combining them to reach a higher accuracy for classification into the six basic emotions defined by Ekman (1992) plus a "Neutral" class. Reports of good generalization performance of deep-learning based methods in recognition tasks (Krizhevsky *et al.*, 2012) suggested an attempt of leveraging their representational power for the EmotiW challenge. The models described in this article cover multiple modalities:

- A novel CNN-based video representation aggregates static facial image features into a fixed-length vector describing the whole video. These features are extracted using a CNN that was trained on a large data set of images annotated with emotion labels. The use of an external data set helped to cope with the challenge of the relatively low number of training samples for deep-learning standards in the challenge data set.
- A Deep Belief Network (DBN) models and classifies the audio stream of the clips.
- A k-means based "Bag-of-Mouths" model extracts visual features around the mouth region.
- A relational auto-encoder activity recognition pipeline explicitly addresses the spatio-temporal aspect in the visual modality.

The outputs of these models are combined using an SVM and a random search strategy. This pipeline achieved an improvement in test set accuracy of 13.47% over the baseline and won the 2013 challenge. One weakness of this approach is that the aggregation of per-frame features into video descriptors is hand-engineered. A learned aggregation strategy is likely to improve this method. As the lead author of this project, I was responsible for the coordination of this joint effort and I was the major contributor in writing, editing, bibliographic research and experimentation in the visual modality. I proposed the strongest model and programmed for the data processing and experimental setup. Since the pipeline was quite complex, I put a lot of effort in debugging. I have also spent a significant amount of time on model combination along with Xavier Bouthillier.

The second article, titled "Recurrent Neural Networks for Emotion Recognition in Video" (Ebrahimi Kahou *et al.*, 2015), addresses the shortcomings of the previous article. Specifically, it introduces a better way of aggregating per-frame facial features into a video descriptor by means of explicit sequential modeling using an RNN. In contrast to the temporal averaging, RNNs are able to preserve information about temporal ordering of detected patterns. In this work the RNN is trained to sequentially read per-frame facial features and make a prediction of the emotion label of the video in the last time step. These per-frame facial features come from a lower level of a fixed CNN, that was trained in a similar way as in the previous article with minor adaption in architecture design to avoid over-fitting. This approach yielded a significantly higher performance compared to the averaging method. Two more modalities, a simple audio model based on PCA followed by an SVM and an activity recognition pipeline are also added to benefit from audio and spatio-temporal visual cues. Inspired by Wu *et al.* (2015b), a regularized MLP was used for feature-level fusion. Finally, a random search is performed to linearly combine modality-specific classifiers and the MLP-classifier. This approach was the second runner-up in the 2015 EmotiW challenge. As mentioned before, it is desirable to train a system end-to-end. The pipeline in this paper uses cropped faces provided by the challenge organizers. Although we re-aligned them to better match the additional data sets, having a pipeline that can learn to focus on a region of interest, i.e. faces, trained directly on the task of emotion recognition would remove a lot of hand-engineering and improve performance, given enough data. I was the main author of this article and contributed to writing, editing, bibliographic research and experimentation. Compared to the previous paper, we ported the major part of the pipeline to Theano (Bastien *et al.*, 2012) for a more uniform framework. I was the lead programmer in this effort and contributed significantly to the code of the CNN, RNN and fusion experiments.

The last article "RATM: Recurrent Attentive Tracking Model" (Kahou *et al.*, 2015b) focuses on one of the central problems in computer vision research, the task of tracking an object in video, given its initial position. While this article is not directly concerned with facial analysis or emotion recognition, progress in the tracking problem is highly relevant for development of an intelligent system that can recognize and reason about human emotions. The proposed modular framework combines a recurrent attention mechanism (Gregor *et al.*, 2015) with a deep neural network. This neural architecture is fully differentiable and can thus be trained by simple gradient-based learning methods. The Recurrent Attentional Tracking Model (RATM) is initially evaluated on artificial data sets of bouncing balls and moving digits. The performance is also shown on an annotated version of the KTH human action data set (Schüldt *et al.*, 2004; Jiang *et al.*, 2012) for tracking humans in video. The central motivation behind this unified neural framework is to overcome the separation of relevant sub-tasks that can be learned jointly. As an example, one could train a model to track faces, extract features and recognize emotions. The joint training of modules for all these sub-tasks enables the model to tune all parameters to solve the target task of emotion recognition. This would remove the dependency of the approaches in the previous articles on a separate pre-processing step for face detection. However, developing such a framework requires exploration of various configurations, many design choices and interpretable results. Experiments on more challenging tracking benchmarks are ongoing work. My responsibilities as the lead author of this article included writing, editing, bibliographic research and experimentation. My contribution to experimentation was mainly in development of the model and experimental setup. I also performed the evaluation of the model.

CHAPTER 4 FACIAL EXPRESSION ANALYSIS BASED ON HIGH DIMENSIONAL BINARY FEATURES

This chapter contains an initial study of facial expression analysis using a variant of Local Binary Patterns (LBPs). The work presented in this chapter was published as a workshop paper at the ECCV 2014 workshop on computer vision with local binary patterns (Kahou *et al.*, 2014). The proposed method is compared with a strong baseline CNN-model and is shown to yield comparable performance. At the time, CNNs gained a lot of popularity in computer vision due to their strong performance. The main experiments of this study were done by myself. However, Pierre Froumenty helped with the CNN experiments presented in Section 4.4.2.

High dimensional engineered features have yielded high performance results on a variety of visual recognition tasks and attracted significant recent attention. Here, we examine the problem of expression recognition in static facial images. We first present a technique to build high dimensional, ∼60k features composed of dense Census transformed vectors based on locations defined by facial keypoint predictions. The approach yields state of the art performance at 96.8% accuracy for detecting facial expressions on the well known Cohn-Kanade plus (CK+) evaluation and 93.2% for smile detection on the GENKI dataset. We also find that the subsequent application of a linear discriminative dimensionality reduction technique can make the approach more robust when keypoint locations are less precise. We go on to explore the recognition of expressions captured under more challenging pose and illumination conditions. Specifically, we test this representation on the GENKI smile detection dataset. Our high dimensional feature technique yields state of the art performance on both of these well known evaluations.

4.1 Introduction

Local binary patterns (LBPs) (Ojala *et al.*, 1996) are well known texture descriptors that are widely used in a number of applications. LBP features have been found to be particularly effective for face related applications (Ahonen *et al.*, 2006). As an example, high dimensional features based on LBPs have yielded highly competitive results on the well known Labeled Faces in the Wild face verification evaluation (Chen *et al.*, 2013; Lu and Tang, 2014).

We are interested here in recognizing facial expressions in static imagery. Facial expression analysis can be a particularly challenging problem, especially when using imagery taken under “in the wild” conditions as illustrated by the recent Emotion Recognition in the Wild Challenge (Sikka *et al.*, 2013). Here we examine both controlled environment facial expression analysis and an “in the wild” problem through evaluations of our proposed method using the Extended Cohn-Kanade (CK+) database (Kanade *et al.*, 2000; Lucey *et al.*, 2010) and the GENKI-4K smile detection evaluation. The CK+ database is a widely used standard evaluation dataset containing acted expressions. The expressions to be recognized are based on Ekman’s six basic universal categories of: happiness, sadness, surprise, fear, anger, and disgust (Eisert and Girod, 1998). The GENKI-4K (GENKI-4K, 2008) dataset contains comparatively low resolution images harvested from the web.

We provide a number of technical contributions in this chapter. First, we provide a formulation of high dimensional features that is different from other standard formulations. Our descriptor is a high dimensional feature vector in which each dimension consists of the bits derived from Census transformation. Features are obtained based on image patches centered on facial keypoints. We use a slight variant of LBPs known as the Census transform (Zabih and Woodfill, 1994). To the best of our knowledge this representation yields the highest known performance on CK+ using the same evaluation criteria as in Lucey *et al.* (2010).

We go on to adapt our technique to be more robust to inaccurately localized facial keypoints using a multi-resolution technique and local Fisher discriminant analysis (LFDA) (Sugiyama, 2007) - a recently proposed extension to the widely used Fisher discriminant analysis technique. The issue of keypoint localization accuracy is particularly important when turning to the problem of recognition in the wild, but even in controlled environments there are well known degradations in performance when per subject keypoint training data is not used to fit a facial keypoint model. Turning to the problem of smile recognition using in the wild GENKI imagery, it is much harder to detect a large number of keypoints due to the quality and variability of the imagery. For the GENKI evaluation in particular we are however able to detect five keypoints reliably. Adapting our method to this setting, here again our proposed method yields the highest known performance of which we are aware on this well known evaluation.

The remainder of this chapter is structured as follows: We provide a brief review of some other relevant work in section 4.2, but also discuss other relevant work throughout the following sections. In section 4.3, we present our novel feature extraction technique based on high dimensional binary features, multi-scale patches and discriminative dimensionality reduction. In section 4.4, we benchmark our high dimensional feature vector technique using CK+, examining experimentally the issue of facial landmark prediction quality, its impact on prediction performance and our motivations for extending our basic formulation to include multi-scale analysis and discriminative dimensionality reduction. We then provide our experiments on GENKI-4K, where we also compare directly with a state of the art convolutional neural network technique that does not rely on keypoints. We provide conclusions and additional discussion in section 4.5.

4.2 Other Relevant Work

A number of modern, state of the art approaches to expression detection are based on handcrafted features, such as: Local binary patterns or LBP features (Ojala *et al.*, 1996), Histograms of oriented gradients or HOG features (Dalal and Triggs, 2005), or Lowe's Scale-invariant feature transform (SIFT) descriptors (Lowe, 1999). For example, the influential work of Shan et al. (Shan *et al.*, 2009) studied histograms of LBP features for facial expression recognition. They introduced Boosted-LBP by using AdaBoost (Freund and Schapire, 1995) for feature selection. Their experiments showed that LBP features are powerful for low resolution images. Dahmane et al. (Dahmane and Meunier, 2011) built face representation based on histograms of HOG features from dense grids. Their representation followed by nonlinear SVM outperforms an approach based on *uniform* LBP. Other work has used SIFT features for facial expression analysis (Sikka *et al.*, 2012), yielding competitive results on CK+.

Techniques based on convolutional neural networks have also yielded state of the art performance for the task of emotion recognition, including top performing results on competitive challenges (Kahou *et al.*, 2013; Tang, 2013; Rifai *et al.*, 2012). The CK+ data and classification tasks were introduced in Lucey et al. (Lucey *et al.*, 2010). They provided both the additional facial examples that were used to extend the original Cohn-Kanade (CK) dataset of Kanade *et al.* (2000), yielding the combined dataset known as CK+ as well a number of experimental analyses. They provided a variety of baseline experiments and a state of the art result at the time in which they combine a landmark based representation (SPTS) and appearance features both before and after shape normalization using landmarks, which they refer to as CAPP features. They combine two different classifiers for landmarks and appearance using a logistic regression on the outputs of the classifiers. This procedure yields their best result with an average accuracy of 83.33%.

Jeni et al. (Jeni *et al.*, 2011) used shape only information for expression recognition experiments with CK+; however, they removed the sequences with noisy landmarks. The work of Sikka et al. (Sikka *et al.*, 2012) compares the performance for a variety of techniques on the CK+ expression recognition task, including the well known uniform LBP histogram technique in Shan *et al.* (2009) which they state yields 82.38% ±2.34 average accuracy. They state that their own bag of words architecture yields 95.85% ±1.4 average per subject accuracy using a leave one subject out evaluation protocol. Other work has also explored the problem of smile detection using the GENKI-4K data. Jain et al. (Jain and Crowley, 2013) report 92.97% accuracy using multi-scale gaussian derivatives combined with an SVM, but they removed ambiguous cases and images with serious illumination problems (423 removed faces). Shan et al. (Shan, 2012) report 89.70% ±0.45 using an Adaboost based technique; however, they manually labeled eye positions which is not practical for many applications. Liu et al. (Liu *et al.*, 2013a) report 92.26% ±0.81 accuracy and also provide the splits used for their evaluation. We therefore use their splits in our evaluation below to permit our technique to be directly compared to their results.

4.3 Our Models

In this section, we present our technique which we show later is capable of obtaining state of the art results on both the CK+ and GENKI evaluations. We also present a deep neural network approach for expression recognition that we shall use for additional comparisons on the GENKI evaluation.

4.3.1 High Dimensional Engineered Features

Our high dimensional feature approach is conceptually simple. We extract a form of local binary pattern known as the Census transform for each pixel found within small image patches, each centered on a facial keypoint. Unlike previous work which typically creates histograms of LBPs, here we create our feature vector by concatenating the bits for each pixel of the image patch into a binary vector. We also concatenate bits obtained from patches extracted at multiple scales centered on the keypoints. As far as we are aware this is different from previous uses of LBP techniques which have relied on histogramming operations. This high dimensional binary feature vector is then projected into a smaller dimensional space via principal component analysis (PCA), followed by a recently proposed variation of multiclass Fisher Discriminant Analysis (FDA) known as local FDA or LFDA (Sugiyama, 2007). The resulting vector is then used within a Support Vector Machine (SVM). There are a number of choices to be made throughout this processing and classification pipeline and we search over key subsets of these choices using cross validation techniques. We discuss the different steps of our procedure in more detail below.

The Census Transform

The Census transform (Zabih and Woodfill, 1994) is computed as follows. If $\mathbf{p} = \{u, v\}$ is the index of a pixel and $I(\mathbf{p})$ is its intensity, define $\xi(\mathbf{p}, \mathbf{p}') = 1$, if $I(\mathbf{p}') < I(\mathbf{p})$; otherwise $\xi(\mathbf{p}, \mathbf{p}') = 0$. The Census transform simply concatenates the bits obtained from comparisons using a fixed ordering of pixels within spatial neighborhood around the pixel. The result is a bit string with ones representing the pixels that are less than the value of the central pixel. Using $\bigotimes$ to denote concatenation, the Census Transform (CT) for the pixel at location $\mathbf{p} = \{u, v\}$ is simply

$$I^c(\mathbf{p}) = \bigotimes_{j=-n}^{n} \bigotimes_{i=-m}^{m} \xi(I(u, v), I(u + i, v + j)), \tag{4.1}$$

typically computed using a window of size $(2m + 1) \times (2n + 1)$. In other words, for a given image patch the CT simply compares each pixel with the center pixel. If its value is greater than the center pixel's value it assigns 0 and 1 otherwise. Common window sizes are 3 and 5. In our experiment, we used 3 as the window size which allows the information to be stored in an 8-bit binary number if desired. The ability to store such descriptors using a binary encoding means that even if our descriptor is of extremely high dimension the information can be stored in a highly compact format. Various other operations using these types of binary descriptors can also be implemented very efficiently.

Keypoint Guided Feature Extraction

As outlined above, we construct our descriptors by cropping small patches out of the larger facial image, applying the Census transform to each pixel for each patch and concatenating the resulting bits into a high dimensional vector. In our experiment below, each scale yields 19,992 features for CK+ and 4,312 for GENKI, due to the different number of keypoints produced by different methods. Patches are extracted centered on each landmark, excluding the face contour. The patches have two parameters that are optimized by cross validation: patch width, defined in proportion to face size and the patch size. The optimal values for our initial CK+ experiment for example were 2/5ths of the face size and 9 pixels in width respectively. Each cropped patch is also resized before computing the Census transform allowing us to control both the dimensionality and the size or spatial extent of the patch separately. We will also present experiments where we extend this approach by extracting patches at each keypoint at three different scales. Depending on the experiment this produces about 60k features.

To obtain keypoints there are a variety of automated placement techniques which can be applied depending on the circumstances. For example, the CK+ dataset comes with landmark positions that were estimated by fitting an Active Appearance Model (AAM) (Matthews and Baker, 2004). AAMs can yield state of the art performance when labeled keypoints have been provided to train models for each subject of interest. For our first set of experiments we use the landmarks provided with the CK+ data. However, AAMs yield poor performance when per subject training data is unavailable. In many real world situations it is impractical to label keypoints for each subject. For this reason there has been a great deal of recent activity focused towards improving alternative approaches that are not identity dependent. For our second CK+ experiments we use the structured max margin technique of Zhu and Ramanan (2012). For GENKI experiments we use the convolutional neural network cascade technique in Sun *et al.* (2013).

Dimensionality Reduction

As we shall see in our experimental work, our high dimensional Census feature technique can yield encouraging results on the CK+ evaluation. However, working with high dimensional vectors can be impractical for many applications. We therefore employ a two phase dimensionality reduction procedure based on an initial projection using PCA followed by LFDA (Sugiyama, 2007). LFDA obtains a discriminative linear projection matrix through minimizing an objective function of the same form as FDA. The underlying problem is therefore also equivalent to solving a generalized eigenvalue problem. More precisely, a projection matrix $\mathbf{M}$ is obtained from

$$\arg\max_{\mathbf{M}} \operatorname{Tr}\Big\{(\mathbf{M}^T\mathbf{S}_W\mathbf{M})^{-1}\mathbf{M}^T\mathbf{S}_B\mathbf{M}\Big\}, \tag{4.2}$$

where there are $i = 1, \ldots, n$ feature vectors $\mathbf{x}_i$ with class labels C_i, given by $c = 1, \ldots, n_c$ class indices, and

$$\mathbf{S}_W = \frac{1}{2}\sum_{i,j=1}^{n} \mathbf{W}_{i,j}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T, \tag{4.3}$$

which defines a *local* within-class scatter matrix using

$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{A}_{i,j} & C_i = C_j = c \\ 0 & C_i \neq C_j, \end{cases} \tag{4.4}$$

and a *local* between-class scatter matrix defined by

$$\mathbf{S}_B = \frac{1}{2}\sum_{i,j=1} \mathbf{B}_{i,j}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T, \tag{4.5}$$

where

$$\mathbf{B}_{i,j} = \begin{cases} \mathbf{A}_{i,j}\left(\frac{1}{n} - \frac{1}{n_c}\right) & C_i = C_j = c \\ \frac{1}{n} & C_i \neq C_j, \end{cases} \tag{4.6}$$

and for both types of local scatter matrix one uses an affinity matrix $\mathbf{A}$ defined, for example by

$$\mathbf{A}_{i,j} = \exp(\|\mathbf{x}_i - \mathbf{x}_j\|^2). \tag{4.7}$$

4.3.2 A Deep Convolutional Neural Network Approach

We shall also compare with a deep convolutional neural network approach to expression recognition based on the framework presented in Krizhevsky *et al.* (2012) which was used to win the recent ImageNet challenge (Deng *et al.*, 2009). The particular architecture we used here for expression recognition is shown in Fig. 4.1. A similar deep neural network architecture and training approach for expression recognition in the wild was used in Kahou *et al.* (2013) to win the recent Emotion Recognition in the Wild Challenge (Dhall *et al.*, 2013) where the goal was to predict expressions in short clips from movies. In Kahou *et al.* (2013) the deep network was only trained on the Toronto Face Database TFD (Susskind *et al.*, 2010) - a large set of different standard expression datasets including Cohn-Kanade and a dataset mined from Google image search results (Carrier *et al.*, 2013) containing 35,887 images tagged with the corresponding emotion categories. In contrast for our GENKI experiments here we do not use additional training data.

Since this implementation and architectural variants of it have won a number of competitive challenges we believe the approach is representative of a state of the art deep neural network approach for expression recognition with wild imagery. We therefore use it here to provide a point of comparison for our GENKI experiments.

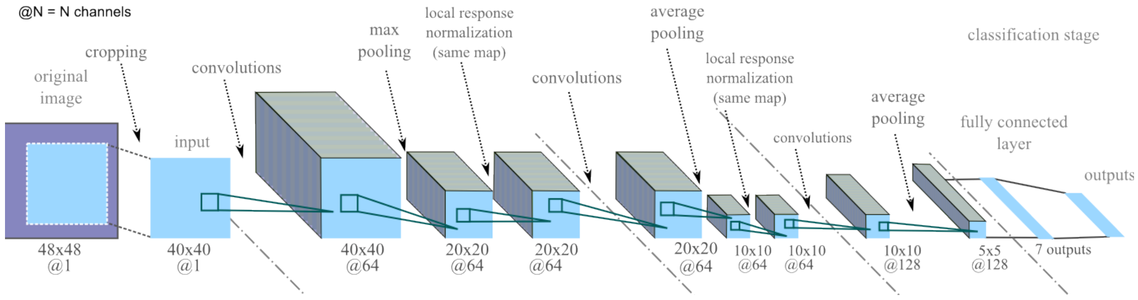


Figure 4.1 The architecture of the convolutional neural network used in our experiments

4.4 Experiments and Results

Here we provide two sets of experiments. First, we present experiments using the standard CK+ evaluation and our high dimensional feature technique. We examine in particular the sensitivity of our approach to keypoint localization quality, the results of which partly motivated the development of the multi-resolution extensions to our basic approach - making it more robust to inaccurate keypoints. We then present results for the smile detection problem using the GENKI-4K dataset, comparing with the deep convolutional neural network approach presented above.

For our last CK+ experiment with noisy keypoints and for our GENKI experiment we apply our full approach in which multi-scale patches are extracted and feature descriptors are reduced in dimensionality using LFDA. The dimensionality reduction is applied on a per patch basis. For PCA we search in the region of dimension reductions that capture 95% of the variance. For LFDA we search in the region of reductions that reduce the final output to 5-20% of the original dimensionality. It is interesting to note that the multi-scale descriptor has about 60k dimensions for our CK+ experiment and is reduced to about 6k dimensions.

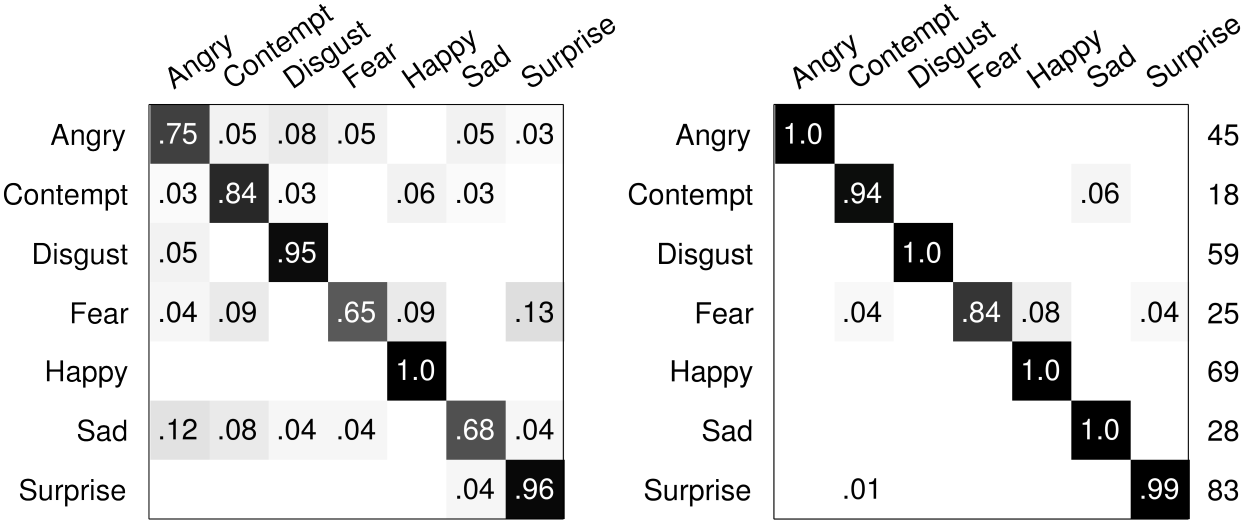
4.4.1 Experiments on CK+

The CK+ database (Kanade *et al.*, 2000; Lucey *et al.*, 2010) is a widely used benchmark for evaluating emotion recognition techniques. It is perhaps more precise to characterize the emotion recognition task using CK+ as facial expression recognition since the majority of sequences were acted. The evaluation includes image sequences with 6 basic expressions. Each sequence starts with a neutral face and ends with an image showing the most exaggerated variation of a given expression. CK+ has large variation in gender, ages and ethnicity. The database consists of 593 image sequences of 123 different subjects and covers both spontaneous and acted expressions. Only one expression "Happy" is spontaneous and it's because some actors smiled during video recordings. CK+ dataset includes labels for expressions, landmarks and labels for the Facial Action Coding System (FACS). We focus here on the expression recognition task.

We use the CK+ data in our work to benchmark and evaluate our approach on a standard dataset before tackling data that is of principal interest to our work in which expressions are exhibited by subjects in natural and spontaneous situations. We begin by placing our high dimensional feature technique in context with the state of the art by showing the complete result of Lucey et al.'s top performing SPTS+CAPP technique discussed in more detail in our literature review (Lucey *et al.*, 2010). To evaluate our technique performance when high precision keypoints are not available we then show the impact of using realistic keypoint predictions from the keypoint predictor in Zhu and Ramanan (2012).

High Dimensional Binary Feature Vectors

For our first experiment here we created a high dimensional binary vector from densely sampled keypoint locations as discussed in section 4.3. We give the resulting vector to a linear support vector machine using the implementation in Chang and Lin (2011). We perform leave one subject out experiments and optimize hyperparameters using an inner cross validation procedure within the training set. Results are shown in Fig. 4.2 (right). We are aware of no other published result with higher performance. The best result of which we are aware on CK+ also gives an accuracy of 96% (Jeni *et al.*, 2011); however, they exclude five subjects from their evaluation. Table 4.1 provides comparison of our results to other methods.



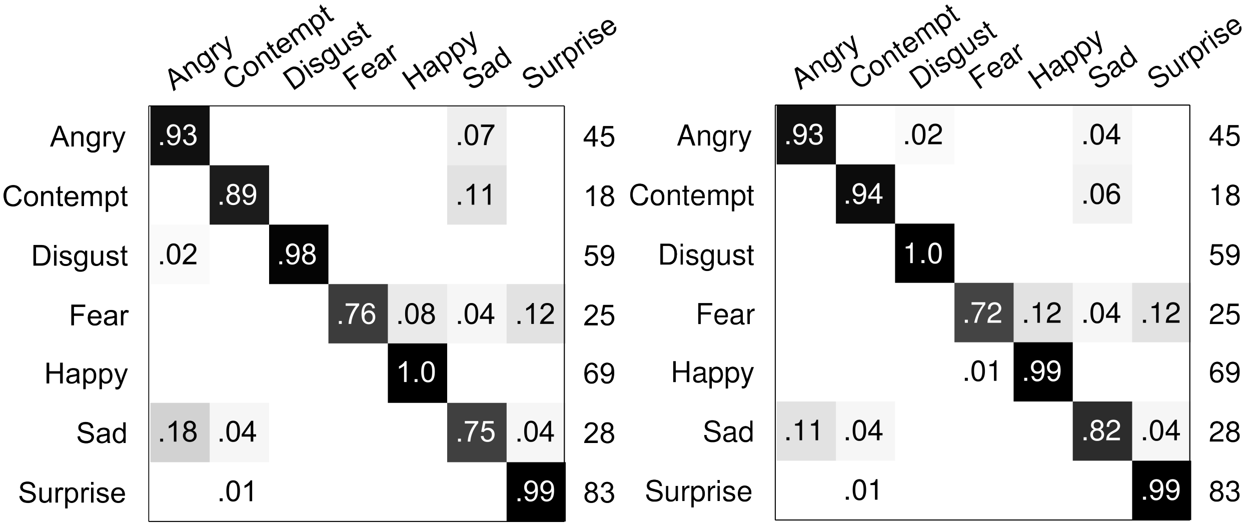
	Angry	Contempt	Disgust	Fear	Happy	Sad	Surprise
Angry	.75	.05	.08	.05		.05	.03
Contempt	.03	.84	.03		.06	.03	
Disgust	.05		.95				
Fear	.04	.09		.65	.09		.13
Happy					1.0		
Sad	.12	.08	.04	.04		.68	.04
Surprise						.04	.96

	Angry	Contempt	Disgust	Fear	Happy	Sad	Surprise	
Angry	1.0							45
Contempt		.94				.06		18
Disgust			1.0					59
Fear		.04		.84	.08		.04	25
Happy					1.0			69
Sad						1.0		28
Surprise		.01					.99	83

Figure 4.2 (**left**) A confusion matrix for expression detection from the SPTS + CAPP result of Lucey et al. (Lucey *et al.*, 2010). The average per class recognition rate was 83.3%. The matrix is row normalized as in Lucey *et al.* (2010). (**right**) The confusion matrix for expression detection on CK+ using our high dimensional binary features. The average per class accuracy is 96.8%. The overall average accuracy is 98.2%. We give the number of examples per class in the column on the right

The Impact of Noisy Keypoints

As we have discussed, in many practical situations it is not possible to obtain highly accurate keypoints such as those possible when using an AAM trained on labeled examples of each subject. For this reason we perform the same experiment above but using the keypoint detector of Zhu and Ramanan (2012). As seen in Fig. 4.3 (left), there is a drop in performance (i.e. 90.0% vs 96.8%), but it is not as dramatic as one might expect due in part to the improved quality for subject independent keypoint predictions afforded by Zhu and Ramanan (2012).



	Angry	Contempt	Disgust	Fear	Happy	Sad	Surprise	
Angry	.93					.07		45
Contempt		.89				.11		18
Disgust	.02		.98					59
Fear				.76	.08	.04	.12	25
Happy					1.0			69
Sad	.18	.04				.75	.04	28
Surprise		.01					.99	83

	Angry	Contempt	Disgust	Fear	Happy	Sad	Surprise	
Angry	.93		.02			.04		45
Contempt		.94				.06		18
Disgust			1.0					59
Fear				.72	.12	.04	.12	25
Happy				.01	.99			69
Sad	.11	.04				.82	.04	28
Surprise		.01					.99	83

Figure 4.3 (**left**) Confusion matrix for expression detection on CK+ using our high dimensional binary features, but based on less accurate keypoints. The average per class accuracy is 90.0%. The overall average accuracy is 93.4%. (**right**) The average per class accuracy when using our multi-scale strategy increases to 91.3% as does the average accuracy, which increases to 94.5%.

The Impact of Multiscale Patches

We then evaluated the hypothesis that the use of multiscale patches centered on each keypoint could make the approach more robust to keypoint localization errors. The result of this experiment is shown in Fig. 4.3 (right). While we cannot recover the original performance, we do see a slight boost in performance over the original single resolution technique.

Table 4.1 CK+ Experiments: Comparison and summary

Method	%
Lucey et al. (2010) [average accuracy] using a landmark based representation and appearance features (Lucey *et al.*, 2010)	83.33
Sikka et al. (2012) [average accuracy] LBP histogram architecture (Shan *et al.*, 2009; Sikka *et al.*, 2012)	82.38
Sikka et al. (2012) [average per subject accuracy] bag of words (Sikka *et al.*, 2012)	95.85
Our technique [average class accuracy], accurate keypoints	96.8
Our technique [average accuracy], accurate keypoints	98.2
Our technique [average accuracy], noisy keypoints	94.5

4.4.2 Smile Detection Experiments

The GENKI-4K dataset (GENKI-4K, 2008; Whitehill *et al.*, 2009) consists of 4,000 facial images labelled with pose and smile content. The images are relatively low resolution and in jpeg format. This dataset has large variations in pose, illumination and ethnicity. We extracted faces from the original images using a combination of the opencv's Haar cascade face detection (Viola and Jones, 2001) and the convolutional neural network cascade of Sun *et al.* (2013). Where these detectors failed to detect any face, we just kept the original.

The resolution of imagery in this dataset was such that we were only able to detect a set of 5 keypoints reliably for our high dimensional feature technique. In order to cover the whole face we computed 6 more points located between eyes, mouth corners and the nose. We provide a comparison with the convolutional neural network (Convnet) architecture discussed in section 4.3.2, which does not rely on keypoints. For both our high dimensional feature technique and our ConvNet experiments we split the dataset into 4 equal folds using the precise splits defined in Liu *et al.* (2013a).

For each experiment with the convolutional neural network, we used random cropping with a 4-pixel border for 48×48 images. Also images were flipped horizontally with a probability of 0.5 at each epoch. The model with no pre-processing yielded 91.5% 1-fold accuracy. We explored preprocessing with isotropic smoothing (Štruc and Pavešić, 2009; Štruc and Pavešić, 2011), yielding 91.5%, and histogram equalization on the grayscale imagery, which yielded 91.7%. From these experiments we found that these preprocessing options did not alter performance in a substantial way. We therefore ran a full four fold experiment using grayscale faces with no preprocessing at 96×96 pixel resolution, which yielded 92.97% ±0.71 accuracy.

Using our complete high dimensional feature technique consisting of both the initial feature construction and including the use of multi-resolution patches and the local fisher discriminant analysis step, followed by the application of an SVM with radial basis function kernel for the final classification, we were able to achieve 93.2% ±0.92 average accuracy. We place our results here in context with prior work in Table 4.2.

4.5 Final Conclusions and Discussion

It is important to emphasize that traditionally LBP based techniques have used histogramming operations to create underlying feature representations. In contrast, in our work we do not compute histograms and use bits directly. For example, previous work (Sikka *et al.*, 2012) has given an accuracy of 82.38% on CK+ for a traditional LBP approach using histograms computed on grid locations defined by a face bounding box using a boosted SVM classification approach. Since we use LFDA to learn a discriminative reduced dimensionality space, our work thus also blurs the lines between traditional notions of engineered feature representations and learned representations. Since we use LBP-like descriptors defined by keypoint locations, in a sense we also blur the lines between keypoint vs. non-keypoint based representations. We hope that our results here will help motivate further work exploring other alternative approaches using LBP descriptors as underlying input representations.

Table 4.2 GENKI-4K Experiments (Accuracies)

Method	%
Shan et al. (2012), using an Adaboost based technique; however, they manually labeled eye positions (Shan, 2012)	89.70
Jain et al. (2013), using multi-scale Gaussian derivatives combined with an SVM; however, they removed ambiguous cases & images with serious illumination problems (423 faces removed) (Jain and Crowley, 2013)	92.97
Liu et al. (2013), using HOG features and SSL (Liu *et al.*, 2013a)	92.29
Liu et al. (2013), with only labeled data	91.85
Our ConvNet at 48 × 48 pixel resolution (no preprocessing)	91.5
Our ConvNet at 96 × 96 pixel resolution (±0.71)	93.0
Our high dimensional LBP technique (±0.92)	93.2

CHAPTER 5 ARTICLE 1: EMONETS: MULTIMODAL DEEP LEARNING APPROACHES FOR EMOTION RECOGNITION IN VIDEO

Journal on Multimodal User Interfaces, June 2016

Samira Ebrahimi Kahou, Xavier Bouthillier, Pascal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, Sébastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, Raul Chandias Ferrari, Mehdi Mirza, David Warde-Farley, Aaron Courville, Pascal Vincent, Roland Memisevic, Christopher Pal and Yoshua Bengio

Abstract

The task of the emotion recognition in the wild (EmotiW) Challenge is to assign one of seven emotions to short video clips extracted from Hollywood style movies. The videos depict acted-out emotions under realistic conditions with a large degree of variation in attributes such as pose and illumination, making it worthwhile to explore approaches which consider combinations of features from multiple modalities for label assignment.

In this paper we present our approach to learning several specialist models using deep learning techniques, each focusing on one modality. Among these are a convolutional neural network, focusing on capturing visual information in detected faces, a deep belief net focusing on the representation of the audio stream, a K-Means based “bag-of-mouths” model, which extracts visual features around the mouth region and a relational autoencoder, which addresses spatio-temporal aspects of videos.

We explore multiple methods for the combination of cues from these modalities into one common classifier. This achieves a considerably greater accuracy than predictions from our strongest single-modality classifier. Our method was the winning submission in the 2013 EmotiW challenge and achieved a test set accuracy of 47.67% on the 2014 dataset.

5.1 Introduction

This is an extended version of the paper describing our winning submission (Kahou *et al.*, 2013) to the Emotion Recognition in the Wild Challenge (EmotiW) in 2013 (Dhall *et al.*, 2013). Here we describe our approach in more detail and present results on the new data set from the 2014 competition (Dhall *et al.*, 2014). The task in this competition is to assign one of seven emotion labels (angry, disgust, fear, happy, neutral, sad, surprise) to each short video clip in the Acted Facial Expression in the Wild (AFEW) dataset (Dhall *et al.*, 2012). The video clips are extracted from feature films. Given the low number of samples per emotion category, it is difficult to deal with the large variety of subjects, lighting conditions and poses in these close-to-real-world videos. The clips are approximately 1 to 2 seconds long and also feature an audio track, which might contain voices and background music.

We explore different methods of combining predictions of modality-specific models, including: (1) a deep convolutional neural network (ConvNet) trained to recognize facial expressions in single frames; (2) a deep belief net that is trained on audio information; (3) a relational autoencoder that learns spatio-temporal features, which help to capture human actions; and (4) a shallow network that is trained on visual features extracted around the mouth of the primary human subject in the video. We discuss each model, their performance characteristics and different aggregation strategies. The best single model, without considering combinations with other experts, is the ConvNet trained to predict emotions given still frames. It has been trained only on additional facial expression datasets, i.e. not using the competition data. The ConvNet was then used to extract class probabilities for the competition data. The extracted probability vectors of the challenge training and validation sets were aggregated to fixed-length vectors and then used to train and validate hyperparameters of a support vector machine (SVM) for final classification. This yielded a test set accuracy of 35.58% for the 2013 dataset. Using our best strategy (at the time) for the combination of top performing expert models into a single predictor, we were able to achieve an accuracy of 41.03% on the 2013 challenge test set. The next best competitor achieved a test accuracy of 35.89%. We reran our pipeline on the 2014 challenge data with improved settings for our combination model and achieved a test set accuracy of 47.67%, compared to 50.37% reported by the challenge winners (Liu *et al.*, 2014).

This paper is structured as follows: Section 5.2 gives an overview of related work, Section 5.3 describes the modality-specific models, Section 5.4 presents experimental results with focus on various fusion strategies and finally, Section 5.5 draws conclusions and provides further insights.

5.2 Related work

The task of assigning an emotion label to a short video clip is well suited for methods and models that combine features from different modalities. As such, many other successful approaches in the Emotion recognition in the Wild (EmotiW) 2013 and 2014 challenges focus on the fusion of modalities. These include (Sikka *et al.*, 2013), who used Multiple Kernel

Learning (MKL) for fusion of visual and audio features. The recent success of deep learning methods in challenging computer vision (Krizhevsky *et al.*, 2012; Neverova *et al.*, 2014; Kahou *et al.*, 2014), language modeling (Kalchbrenner *et al.*, 2014) and speech recognition (Hinton *et al.*, 2012a) tasks seems to carry over to emotion recognition, taking into account that the 2014 challenge winners (Liu *et al.*, 2014) also employed a deep convolutional neural net, which they combined with other visual and audio features using a Partial Least Squares (PLS) classifier. The adoption of deep learning for visual features likely played a big role in the considerable improvement compared to their submission in the 2013 competition (Liu *et al.*, 2013b), although the first and second runners up also reached quite good performances without deep learning methods; (Sun *et al.*, 2014) used a hierarchical classifier for combining audio and video features and Chen *et al.* introduced an extension of Histogram of Oriented Gradients (HOG) descriptors for spatio-temporal data, which they fuse with other visual and audio features using MKL.

5.3 Models for modality-specific representation learning

5.3.1 A convolutional network approach for faces

ConvNets are artificial neural network architectures, that assume a topological input space, e.g. a 2d image plane. A set of two-dimensional or three-dimensional (if the inputs are color images) filters is applied to small regions over the whole image using convolution, yielding a bank of filter response maps (one map per filter), which also exhibit a similar 2d topology.

To reduce the dimensionality of feature banks and to introduce invariance with respect to slight translations of the input image, convolutional layers are often followed by a pooling layer, which subsample the feature maps by collapsing small regions into a single element (for instance by choosing the maximum or mean value in the region). ConvNets have recently been shown to achieve state of the art performance in challenging object recognition tasks (Krizhevsky *et al.*, 2012).

Because of the small number of training samples, our initial experiments with ConvNets showed severe overfitting on the training set, achieving an accuracy of 96.73% on the AFEW2 training set, compared to only 35.32% on the validation set. For this reason we decided to train on a separate dataset, which we refer to as 'extra data'. It consists of two face image datasets and is described in Section 5.3.1.

The approach for the face modality can roughly be divided into four stages:

1. Training the ConvNet on faces from extra data. The architecture is described in Section

5.3.1.

2. Extraction of 7-class probabilities for each frame of the facetubes (described in Section 5.3.1).

3. Aggregation of single frame probabilities into fixed-length video descriptors for each video in the competition dataset by expansion or contraction.

4. Classification of all video-clips using a support vector machine (SVM) trained on video descriptors of the competition training set.

Stage three and four are described in detail in Section 5.3.1. The pipeline is depicted in Figure 5.1. The strategy of training on extra data and using the competition data only for classifier training and early stopping yielded a much lower training set accuracy of 46.87%, but it achieved a considerably better validation set accuracy of 38.96%.

Additional Face Dataset

The 'extra data' we used for training of the deep network is composed of two large static image datasets of facial expressions for the seven emotion classes.

The first and larger one is the Google dataset (Carrier *et al.*, 2013) consisting of 35,887 images with the seven facial expression classes: angry, disgust, fear, happy, sad, surprise and neutral. The dataset was built by harvesting images returned from Google's image search using keywords related to expressions, then cleaned and labeled by hand. We use the grayscale 48 × 48 pixel versions of these images. The second one is the Toronto Face Dataset (TFD) (Susskind *et al.*, 2010) containing 4,178 images labeled with basic emotions, essentially with only fully frontal facing poses.

To make the datasets compatible (there are big differences, for instance variation among subjects, lighting and poses), we applied the following registration and illumination normalization strategies:

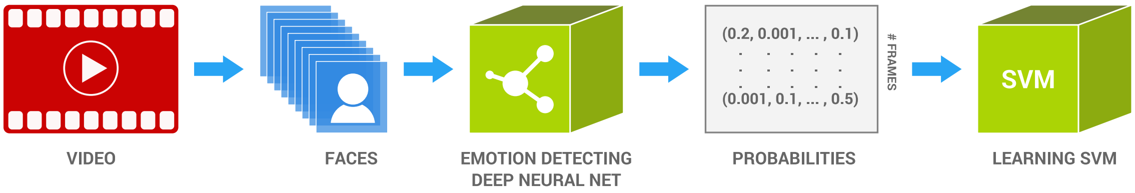


Figure 5.1 Complete pipeline describing the final strategy used for our ConvNet №1 model.

Registration To build a common dataset, TFD images and frames from the competition dataset had to be integrated with the Google dataset, for which we used the following procedure: For image registration we used 51 of the 68 facial keypoints extracted by the mixture of trees method from Zhu and Ramanan. The face contour keypoints returned by this model were ignored in the registration process. Images from the Google dataset and the AFEW datasets have different poses, but most faces are frontal views.

To reduce noise, the mean shape of frontal pose faces for each dataset was used to compute the transformation between the two shapes. For the transformation the Google data was considered as base shape and the similarity transformation was used to define the mapping. After inferring this mapping, all data was mapped to the Google data. TFD images have a tighter fit around faces, while Google data includes a small border around the faces. To make the two datasets compatible, we added a small noisy border to all images of TFD.

Illumination normalization using isotropic smoothing To compensate for varying illumination in the merged dataset, we used the diffusion-based approach introduced in Heusch *et al.*. We used the isotropic smoothing (IS) function from the INface toolbox (Štruc and Pavešić, 2009; Štruc and Pavešić, 2011) with the default smoothness parameter and without normalization as post-processing. A comparison of original and IS-preprocessed face images is shown in figure 5.2.

Extracting frame-wise emotion probabilites

Our ConvNet uses the C++ and CUDA implementation written by Alex Krizhevsky (Krizhevsky, 2012) interfaced in Python. The network's architecture used here is presented in Figure 5.3. The ConvNet takes batches of 48 × 48 images as input and performs a random cropping into smaller 40 × 40 sub-images at each epoch. These images are then randomly flipped

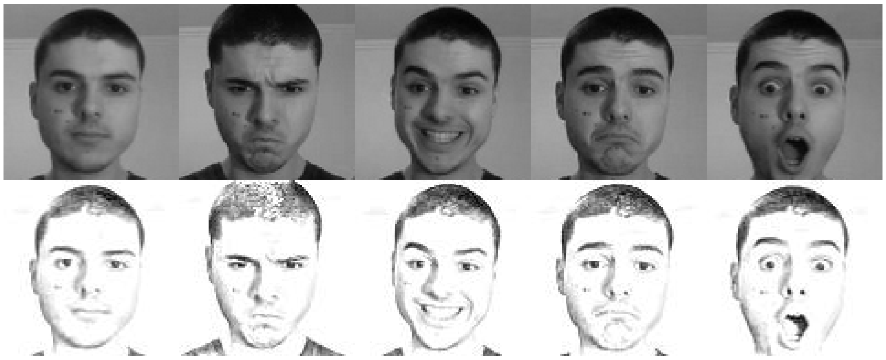


Figure 5.2 Raw images at the top and the corresponding IS-preprocessed images below.

horizontally with a probability of 0.5. These two common methods allow us to expand the limited training set and avoid over-fitting.

The ConvNet architecture has 4 stages containing different layers. The first two stages include a convolutional layer followed by a pooling layer, then a local response normalization layer (Krizhevsky *et al.*, 2012). The third stage includes only a convolutional layer followed by a pooling layer. Max-pooling is used in the first stage, while average-pooling is used in the next stages. The last stage consists of seven softmax units, which output seven probabilities, one for each of the seven emotion labels. The activation function used in the convolutional layers is the rectified linear unit (ReLU) activation function. The two first convolutional layers use 64 filters each, and the last one 128, all of size 5 × 5 pixels. Each convolutional layer has the same learning parameters: a 0.001 learning rate for the filters and 0.002 for biases, 0.9 momentum for both filters and biases and a weight decay of 0.004 per epoch. The fully-connected layer shares the same hyperparameters except for the weight decay, which we set to 1. These hyperparameters are the same as the one provided by Krizhevsky in his example layers configuration files. The architecture is depicted in Figure 5.3.

Classification at test time is done using the 40 × 40 sub-images cropped from the center of the original images. We stopped learning at 453 epochs using early-stopping on the competition validation and train sets. As stated earlier, we only used extra data to train the network, and the competition training and validation datasets were only used for early stopping and the subsequent training of the SVM.

A shallower ConvNet was explored for the 2013 competition. It performed worse than ConvNet 1 and we did not revisit it for the 2014 dataset. In the tables for the AFEW2 results, it is referred to as ConvNet 2. For details on the architecture see Kahou *et al.*.

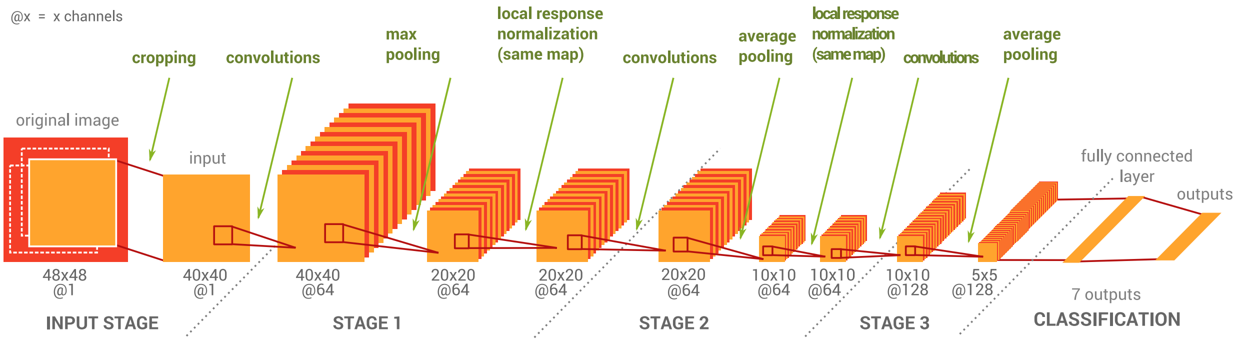


Figure 5.3 The architecture of our ConvNet №1.

Facetube extraction procedure

For the competition dataset video frames were extracted preserving the original aspect ratio. Then the Google Picasa face detector (Google, 2013) was used to crop detected faces in each frame. To get the bounding box parameters in the original image, we used rectangular Haar-like features (Viola and Jones, 2001) for matching, which allowed us to rapidly compute the best matching regions from a precomputed integral image. Picasa did not detect faces in every frame. To fix this, we searched the spatial neighborhood of the temporally closest bounding box for regions with an approximately matching histogram of color intensities. We used heuristics, such as the relative positioning, sizes and overlap, to associate bounding boxes of successive frames and generate a continuous sequence of face-containing subframes for each subject in the video. If one imagines a stack of video frames as a 3-dimensional volume of width × height × time, each extracted sequence of subframes forms a "tube" embedded in this volume. We therefore refer to these extracted sequences as *facetubes*.

For a few clips in the competition test sets, the Picasa face detector did not detect any faces. So we used the combined landmark placement and face detection method described in Zhu and Ramanan to find faces in these clips. Using the facial keypoints output by that model we built bounding boxes and assembled them into facetubes with the previously described procedure.

Facetube smoothing In order to get image sequences where face sizes vary gradually, we applied a smoothing procedure on the competition facetube bounding boxes. Bounding box parameters were smoothed by averaging values inside a temporal window of size 11 frames. The largest centered squares, that fit into these smoothed rectangular bounding boxes, yielded new bounding boxes which more tightly frame the detected faces. To restrict the amount of motion of the bounding boxes the same kind of smoothing was also applied to the center of the bounding boxes.

Side lengths of the bounding boxes can vary due to changes of camera position or magnification (e.g. changing from a medium shot to a close-up shot). To be able to handle this, a further polynomial smoothing technique was applied directly on the bounding box side lengths. Two low-order polynomials of degree 0 (constant) and 1 (linear) were fit through the side lengths of the bounding boxes. If the slope of the linear polynomial is above a scale threshold ($slope \cdot facetube\ length$), we use the values of the linear polynomial as side lengths, else we use values from the constant smoothing polynomial. Empirically, we found that a threshold of 1.5 yielded reasonable results.

The final facetubes were then generated by cropping based on the smoothed bounding boxes and resizing the patches to 48×48. Per-frame emotion label probabilities were extracted for each facetube using the ConvNet.

Aggregation into video descriptors and classification

We aggregated the per-frame probabilities for all frames of a facetube for which a face was detected into a fixed-length video descriptor to be used as input to an SVM classifier. For this aggregation step we concatenated the seven-dimensional probability vectors of ten successive frames, yielding 70 dimensional feature vectors. Most videos have more than ten frames and some are too short and there are frames without detected faces. We resolved these problems using the following two aggregation approaches:

- Video averaging: For videos that were too long, we averaged the probability vectors of 10 independent groups of frames taken uniformly along time, contracting the facetube to fit into the 10-frame video descriptors. This is depicted in Figure 5.4.
- For videos that contain too few frames with detected faces, we expanded by repeating frames uniformly to get 10 frames in total. This is depicted in Figure 5.5.

The video descriptors for the training set were then used to train an SVM (implemented by Chang and Lin) with a radial basis function (RBF) kernel. The hyperparameters, γ and c were tuned on the competition validation set. The SVM type used in all experiments was a C-SVM classifier and the outputs are probability estimates so that the fusion with other results was simpler.

5.3.2 Audio & Deep Belief Networks

As we have described earlier, deep learning based techniques have led to important successes in speech recognition (Hinton *et al.*, 2012a; Graves *et al.*, 2013). In the context of emotion recognition on audio features extracted from movie clips, we used a deep learning approach for performing emotion recognition just by pretraining a deep MLP as a deep belief network (DBN) (Hinton *et al.*, 2006). A DBN is a probabilistic generative model where each layer can be greedily trained as a Restricted Boltzmann Machine (RBM). We trained the network as a DBN in an unsupervised manner with greedy layerwise training procedure and then we used supervised finetuning . In order to tune the hyperparameters of our model, we performed cross-validation using the competition validation dataset. We initially used a random search for hyperparameters and after the random search, we did manual finetuning of hyperparameters.

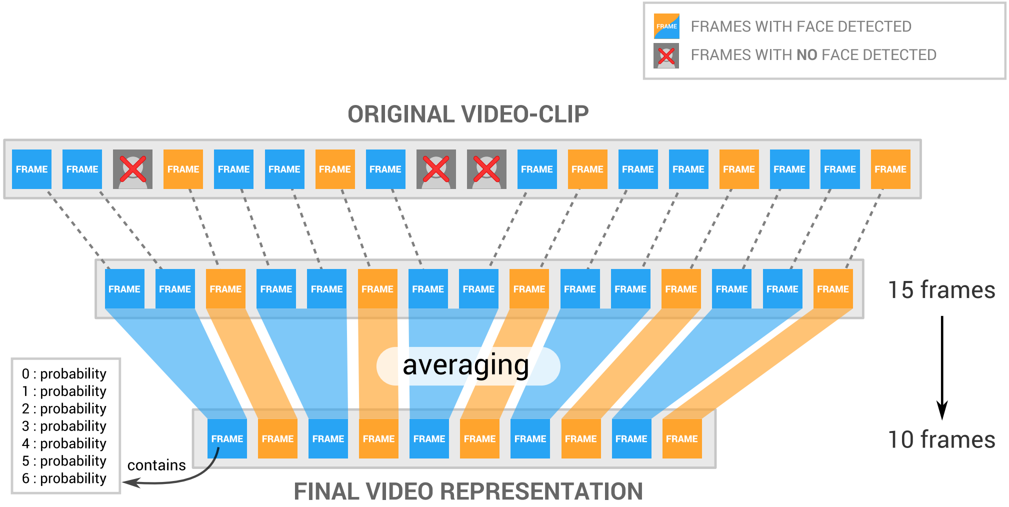


Figure 5.4 Frame aggregation via averaging

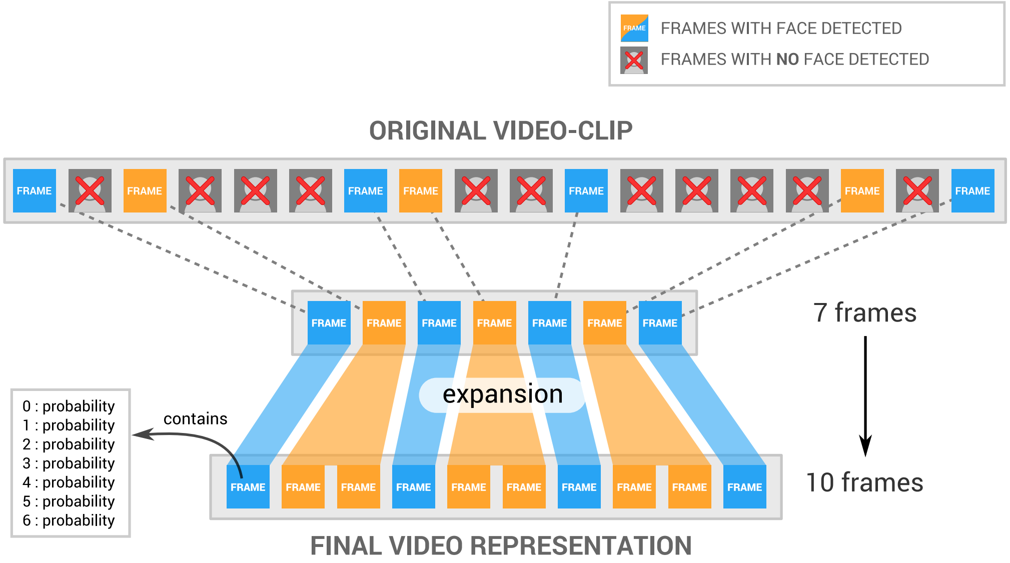


Figure 5.5 Frame aggregation via expansion

Audio Preprocessing

Choosing the right features is a crucial aspect of the audio classification. Mel-frequency cepstral coefficients (MFCCs) are widely used for speech recognition; however, in this task we are mainly interested in detecting emotions from the extracted audio features.

On the other hand emotion recognition on film audio is quite different from other audio tasks. In addition to speech in the audio track, background noise and the soundtrack can also be significant indicators of emotion. For the EmotiW challenge, we extracted 29 features from each audio track using the Yaafe library[1] with a sampling rate of 48 kHz. We used all features provided by the Yaafe library except "Frames". Additionally 3 types of MFCC features are used: the first used 22 cepstral coefficients (excluding the first coefficient), the second used a feature transformation with the temporal first-order derivative and the last one employed second-order temporal derivatives. All features were concatenated, yielding a 909-dimensional vector for each timestep. Online PCA was applied on the extracted features for whitening (Hamel *et al.*, 2011).

DBN Pretraining

We used unsupervised pre-training with deep belief networks (DBN) on the extracted audio features. The DBN has three layers of RBMs, the first layer is a Gaussian RBM with noisy rectified linear unit (ReLU) nonlinearity (Dahl *et al.*, 2013), the second and third layer are both Gaussian-Bernoulli RBMs. We trained the RBMs using stochastic maximum likelihood and contrastive divergence with one Gibbs step (CD-1).

Each RBM layer had 350 hidden units. The three layers of RBM were trained with learning rates of 0.0006, 0.0005 and 0.001 for each layer respectively. An L2 penalty of 2×10^{-3} and 2×10^{-4} was used for the first and second layer, respectively. Both the first and second layer RBMs were trained for 15 epochs on the competition training dataset. We bounded the noisy ReLU activations of the first layer Gaussian RBM, specifically we used the activation function: $min(\alpha, max(0, \mathbf{x} + \psi))$, where $\psi \sim N(0, \sigma(\mathbf{x}))$ with $\alpha = 6$. Otherwise large activations of the first layer RBM were causing problems training the second layer Gaussian Bernoulli RBM. We used a Gaussian model of the form $N(0, \sigma(\mathbf{x}))$, with 0 mean and standard deviation of $\sigma(\mathbf{x}) = \frac{1}{1+exp(-\mathbf{x})}$. At the end of unsupervised pre-trainining, we initialized a multilayer perceptron (MLP) with the ReLU nonlinearities of all layers using the weights and biases of the DBN.

[1] Yaafe: audio features extraction toolbox: `http://yaafe.sourceforge.net/`

Temporal Pooling for Audio Classification

We used a temporal pooling based learning model as proposed in Hamel *et al.* for the MLP where we pooled the last hidden representation layer of an MLP so as to aggregate information across frames before a final softmax layer. We experimented with various pooling methods including max pooling and mean pooling, but we obtained the best results with a specifically designed type of pooling for the MLP features discussed below.

Assume that we have a matrix A for the activations of the MLP's last layer features that includes activations of all timesteps/frames in the clip where $A \in R^{d_t \times d_f}$ and d_t is the variable number of timesteps, d_f is the number of features at each timestep. We sort the columns of A in decreasing order and get the top N rows using the map $f : R^{d_t \times d_f} \rightarrow R^{N \times d_f}$. The most active N features are summarized with a weighted average of the top-N features:

$$F = \frac{1}{N} \sum_{i=0}^{N} w_i f^{(i)}(A; N) \tag{5.1}$$

where $f^{(i)}(A; N)$ is the i^{th} highest active feature over time and weights should be: $\sum_{i=0}^{N} w_i = N$. During the supervised finetuning, we feed the reduced features to the top level softmax, we backpropagate through this pooling function to the lower layers. We only used the top 2 ($N = 2$) most active features in the weighted average. Weights of the features were not learned and they were chosen as $w_1 = 1.4, w_2 = 0.6$ during training and $w_1 = 1.3, w_2 = 0.7$ during test time. This kind of feature pooling technique worked best, if the features are extracted from a bounded nonlinearity such as $sigmoid(.)$ or $tanh(.)$.

Supervised Fine-tuning

The competition training dataset was used for supervised fine-tuning and we applied early stopping by measuring the error on the competition validation dataset. The features were centered prior to training. Before initiating the supervised training, we shuffled the order of clips. During the supervised fine-tuning phase, at each iteration on the training dataset, we randomly shuffled the order of the features in the clip as well. At each training iteration, we randomly dropped out 98 clips from the training dataset and we randomly dropped out 40% of the features in the clip. The dropout rate for hidden units is set to 12.1% and we used a norm constraint on the weights such that the L2 norm of the incoming weights to a hidden unit does not exceed 1.29 (Hinton *et al.*, 2012b). In addition to drop-out and maximum norm constraint on the weights, a L2 weight penalty with coefficient of 10^{-5} was used. The rmsprop adaptive learning rate algorithm was used to tune the learning rate with a variation of Nesterov's Momentum (Sutskever *et al.*, 2013). RMSProp scales down parameter updates by a running average of the gradient norm. At each iteration we keep track of the mean square of the gradients by:

$$RMS(\Delta_{t+1}) = \rho RMS(\Delta_t) + (1-\rho)\Delta_t^2 \tag{5.2}$$

and compute the momentum, then do the stochastic gradient descent (SGD) update:

$$v_{t+1} = \mu v_t - \epsilon_0 \frac{\partial f(x^{(i)};\theta_t)}{\partial \theta_t}, \tag{5.3}$$

$$\theta_{t+1} = \theta_t + \frac{\mu v_{t+1} - \epsilon_0 \frac{\partial f(x^{(i)};\theta_t)}{\partial \theta_t}}{\sqrt{RMS(\Delta_{t+1})}} \tag{5.4}$$

After performing crossvalidation, we decided to use an $\epsilon_0 = 0.0005$, $\mu = 0.46$ and $\rho = 0.92$. We used early stopping based on the validation set performance, yielding an accuracy of 32.90% on EmotiW 2014 validation set. Once supervised fine-tuning had completed 50 iterations, if the validation error continued increasing, the learning rate was decreased by a factor of 0.99.

5.3.3 Activity recognition using a relational autoencoder

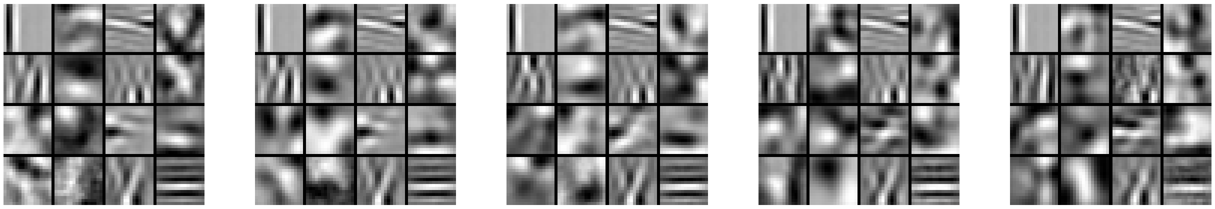


Figure 5.6 Subset of filters learned by SAE model on the AFEW4 training set. Left to right: Frames 1,3,5,7 and 9.

Given a video sequence with the task of extracting human emotion labels, it seems reasonable to also consider the temporal evolution of image frames. To this end we employ an activity recognition system for emotion recognition based on local spatio-temporal feature computation. Using local motion features for activity recognition is a popular approach employed in many previous works (Le *et al.*, 2011; Konda *et al.*, 2014b; Taylor *et al.*, 2010; Wang *et al.*, 2009).

Traditional motion energy models (Adelson and Bergen, 1985) encode spatio-temporal features of successive video frames as sums of squared quadrature Fourier or Gabor coefficients across multiple frequencies and orientations (Le *et al.*, 2011). Summing induces invariance w.r.t. content, allowing the model to yield a pure motion representation. In contrast to the motion energy view, in Konda *et al.* it has been shown that the learning of transformations and introduction of invariance can be viewed as two independent aspects of learning. Based on that view, a single layered autoencoder based model named *synchrony autoencoder* (SAE) for learning motion representations was introduced. The classic approach is to use hand-engineered features for spatio-temporal feature extraction (Wang *et al.*, 2009). In contrast to hand-engineered features, deep learning based methods have been shown to yield low-level motion features, which generalize well across datasets (Le *et al.*, 2011; Konda *et al.*, 2014b).

We use a pipeline commonly employed in works on activity recognition (Le *et al.*, 2011; Konda *et al.*, 2014b; Wang *et al.*, 2009) with the SAE model for local motion feature computation. We chose to use the SAE model because, compared to other learning based methods like ISA (Le *et al.*, 2011) and convGBM (Taylor *et al.*, 2010) with complex learning rules, it can be trained very efficiently, while performing competitively. The activity recognition pipeline follows a bag-of-words approach. It consists mainly of three modules: motion feature extraction, K-means vector quantization and a χ^2 kernel SVM for classification. The SAE model acts as feature extractor. It is trained on small video blocks of size $10 \times 16 \times 16$ ($time \times rows \times columns$) randomly cropped from the competition training set. They are preprocessed using PCA for whitening and dimensionality reduction, retaining 300 principal components. The number of randomly cropped training samples is $200,000$. The size of the SAE's hidden layer was fixed at 300. The model was trained using SGD with a learning rate of 0.0001 and momentum 0.9 for $1,000$ epochs. The filters learned by the model on videos from the AFEW4 training set are visualized in Figure 5.6.

In past works it has been shown that spatially combining local features learned from smaller input regions leads to better representations than features learned on larger regions (Le *et al.*, 2011; Coates *et al.*, 2011). Here, we utilize the same method by computing local feature descriptors for sub blocks cropped from the corners of a larger $14 \times 20 \times 20$ "super block" and concatenating them, yielding a descriptor of motion for the region covered by the super block. PCA was applied to this representation for dimensionality reduction, retaining the first 100 principal components. To generate descriptors for a whole video, super blocks are cropped densely for each video with a stride of 7 on the temporal axis and 10 on the spatial axes, i.e. with 50% overlap of neighboring super blocks. The K-means clustering step produces a dictionary of 3000 words, where each word represents a motion pattern. A normalized histogram over $K-means$ cluster assignment frequencies was generated for each video as input to the classifier.

In our experiments we observed that the classifier trained on the motion features seemed to overfit on the training set and all investigated measures to avoid this problem (e.g. augmenting the data set by randomly applying affine transformations to the input videos) were also not helpful. This could be due to the videos showing little to no motion cues that correlate heavily with the emotion labels. The motion model by itself is not very strong at discriminating emotions, but it is useful in this task, nonetheless. It helps to disambiguate cases, where other modalities are not very confident, because it represents some characteristics of the data additional to those described by the other modalities.

5.3.4 Bag of mouth features and shallow networks

Some emotions may be recognized from mouth features. For example, a smile often indicates happiness while an “O”-shaped open mouth may signal surprise. For our submission, facetubes, described in section 5.3.1, in resolution 96 × 96 were cropped around a region where the mouth usually lies. This region was globally chosen by visualizing many training images, but a more precise method, such as mouth keypoint extraction (Zhu and Ramanan, 2012), could also be applied.

We mostly follow the method introduced by Coates *et al.*, which achieved state-of-the-art performance on the CIFAR-10 dataset (Krizhevsky, 2009) in 2011, even though that method has since been superseded by convolutional networks. As a first step, each mouth image is divided into 16 equally sized sections, from which many 8 × 8 pixel patches are extracted. These are normalized by individually setting the mean pixel intensity to 0 and the variance to 1. After centering all patches from the same spatial region, we apply whitening, which was shown to be useful for this kind of approach (Coates *et al.*, 2011), keeping 90% of the variance. For each of the 16 regions, 400 centroids are found by applying the k-means algorithm on the whitened patches.

For any given image, patches are densely extracted from each of the 16 regions and preprocessed as described above. Each patch is assigned a 400-dimensional vector by comparing it to the centroids with the triangle activation function (Coates *et al.*, 2011), where the Euclidean distance z_k between the patch and each centroid is computed, as well as the mean μ of these distances. The activation of each feature is given by $max(0, \mu - z_k)$, so that only centroids closer than the mean distance are assigned a positive value, while distant ones stay at 0. As we have a 400-dimensional representation for each patch, the image representation would become extremely large if we simply concatenated all feature vectors. For this reason, we pool over all features of a region to get a local region descriptor. The region descriptors are then concatenated to obtain a 6,400 dimensional representation of the image.

This pooling generally uses the average activation of each feature, although we also tried taking the standard deviation across patches for each feature. A regularized logistic regression classifier is trained on a frame-by-frame basis with the pooled features as input. When classifying a test video, the predictions of the model are averaged over all its frames.

5.4 Experimental results

In figure 5.7 (a-d) we show the validation set confusion matrices from the models yielding the highest AFEW4 validation set accuracy for each of the techniques discussed in section 5.3. A second convolutional network for faces (Convnet #2), which we explored, is not presented here as it obtained lower performance compared to Convnet #1 and used similar information to make its predictions. A more detailed analysis of Convnet #2 and comparisons on AFEW2 can be found in Kahou *et al.*, but we provide some highlights here.

AFEW2 From our experiments with AFEW2 we found that ConvNet1 yielded the highest validation set accuracy. We therefore selected this model as our first submission and it yielded a test set accuracy of **35.58**%. This is also indicated in table 5.1 which contains a summary of all our submissions. ConvNet2 was our second highest performer, followed closely by the bag of mouth and audio models at 30.81%, 30.05% and 29.29% respectively.

AFEW4 Here again our ConvNet1 model achieved the best results on the validation set for AFEW4. It was followed by our audio model which here yields higher performance than the bag of mouths model by a good margin, at 34.20% and 27.42% accuracy respectively. We explored the strategies outlined in Sections 5.4.1, 5.4.2 and 5.4.3 to combine models for the AFEW2 evaluation. Section 5.4.4 presents the strategy we used for our experiments with the AFEW4.

5.4.1 Averaged Predictions – AFEW2

A simple way to make a final prediction using several models is to take the average of their predictions. We had 5 models in total, which gives $\sum_{i=1}^{n} \binom{n}{i} = 31$ possible combinations (order has no importance). In this context it is possible to test all combinations on the validation set to find those which are the most promising.

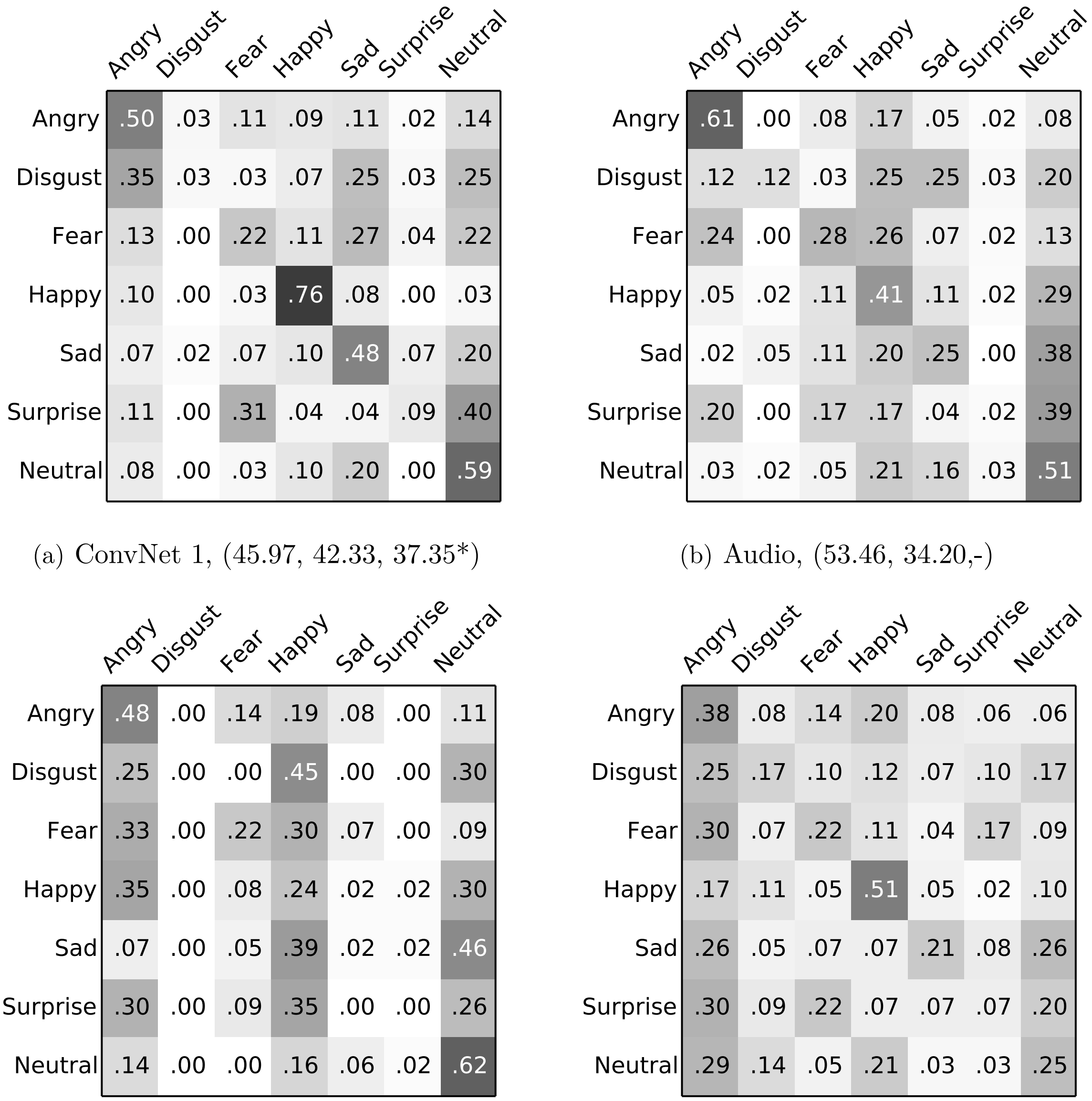
Through this analysis we found that the average of all models yielded the highest validation set performance of 40.15% on AFEW2. The validation set confusion matrix for this model is shown in figure 5.8 (a). For our third 2013 submission we therefore submitted the results of

Table 5.1 Our 7 submissions with training, validation and test accuracies for the EmotiW 2013 competition. Best accuracies are indicated in bold.

Sub.	Train	Valid	Test	Method
1	45.79	38.13	35.58	Google data & TFD used to train ConvNet 1, SVM trained on aggregated frame scores
2	71.84	42.17	38.46	ConvNet 1 (from submission 1) combined with Audio model using another SVM
3	97.11	40.15	37.17	Mean prediction from: Activity, Audio, Bag of mouth, ConvNet 1, ConvNet 2
4	98.68	43.69	32.69	SVM with detailed hyperparameter search: Activity, Audio, Bag of mouth, ConvNet 1
5	94.74	47.98	39.42	Short uniform random search: Activity, Audio, Bag of mouth, CN1, CN1 + Audio
6	94.74	48.48	40.06	Short local random search: Activity, Audio, Bag of mouth, CN1, CN1 + Audio
7	92.37	**49.49**	**41.03**	Moderate local random search: Activity, Audio, Bag of mouth, CN1, CN1 + Audio

Table 5.2 Our selected submissions with test accuracies for the EmotiW 2014 competition. Best accuracy is indicated in bold.

Sub.	Test	Method
1	39.80	Trained model on 2013 data, BoM failed due to different data format and replaced by uniform
2	37.84	Trained model on 2013 data, re-learning random search without failed BoM
3	44.71	ConvNet 1 + Audio model combined with SVM, all trained on train+valid
4	41.52	ConvNet 1 + Audio model combined with SVM trained on swapped predictions
5	37.35	Google data & TFD used to train ConvNet 1, frame scores aggregated with SVM
6	42.26	All models combined with SVM trained on validation predictions
7	44.72	All models combined with random search optimized on validation predictions
8	42.51	Only two models were trained on train+validation in combination, others used train set only
9	**47.67**	All models combined with random search optimized on full swapped predictions
10	45.45	Bagging of 350 models similar to submission 9



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.50	.03	.11	.09	.11	.02	.14
Disgust	.35	.03	.03	.07	.25	.03	.25
Fear	.13	.00	.22	.11	.27	.04	.22
Happy	.10	.00	.03	.76	.08	.00	.03
Sad	.07	.02	.07	.10	.48	.07	.20
Surprise	.11	.00	.31	.04	.04	.09	.40
Neutral	.08	.00	.03	.10	.20	.00	.59

(a) ConvNet 1, (45.97, 42.33, 37.35*)

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.61	.00	.08	.17	.05	.02	.08
Disgust	.12	.12	.03	.25	.25	.03	.20
Fear	.24	.00	.28	.26	.07	.02	.13
Happy	.05	.02	.11	.41	.11	.02	.29
Sad	.02	.05	.11	.20	.25	.00	.38
Surprise	.20	.00	.17	.17	.04	.02	.39
Neutral	.03	.02	.05	.21	.16	.03	.51

(b) Audio, (53.46, 34.20,-)

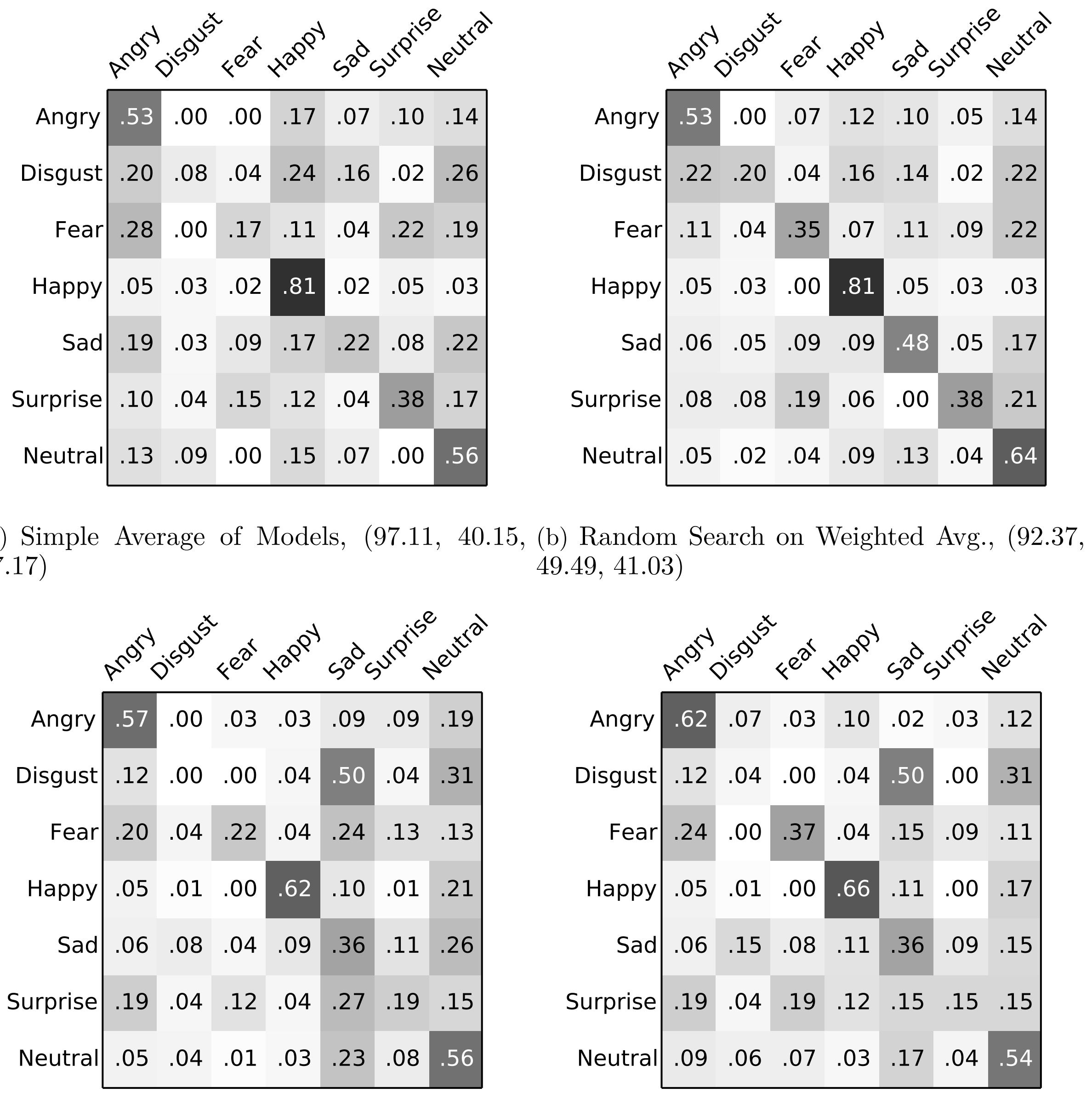
	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.48	.00	.14	.19	.08	.00	.11
Disgust	.25	.00	.00	.45	.00	.00	.30
Fear	.33	.00	.22	.30	.07	.00	.09
Happy	.35	.00	.08	.24	.02	.02	.30
Sad	.07	.00	.05	.39	.02	.02	.46
Surprise	.30	.00	.09	.35	.00	.00	.26
Neutral	.14	.00	.00	.16	.06	.02	.62

(c) Activity Rec., (46.37, 25.07,-)

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.38	.08	.14	.20	.08	.06	.06
Disgust	.25	.17	.10	.12	.07	.10	.17
Fear	.30	.07	.22	.11	.04	.17	.09
Happy	.17	.11	.05	.51	.05	.02	.10
Sad	.26	.05	.07	.07	.21	.08	.26
Surprise	.30	.09	.22	.07	.07	.07	.20
Neutral	.29	.14	.05	.21	.03	.03	.25

(d) Bag of mouth, (93.08, 27.42,-)

Figure 5.7 Confusion matrices for the AFEW4 validation set. Accuracies for each method are specified in parentheses (training, validation & test sets, if applicable). *Model has been retrained on both training and validation set prior to testing



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.53	.00	.00	.17	.07	.10	.14
Disgust	.20	.08	.04	.24	.16	.02	.26
Fear	.28	.00	.17	.11	.04	.22	.19
Happy	.05	.03	.02	.81	.02	.05	.03
Sad	.19	.03	.09	.17	.22	.08	.22
Surprise	.10	.04	.15	.12	.04	.38	.17
Neutral	.13	.09	.00	.15	.07	.00	.56

(a) Simple Average of Models, (97.11, 40.15, 37.17)

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.53	.00	.07	.12	.10	.05	.14
Disgust	.22	.20	.04	.16	.14	.02	.22
Fear	.11	.04	.35	.07	.11	.09	.22
Happy	.05	.03	.00	.81	.05	.03	.03
Sad	.06	.05	.09	.09	.48	.05	.17
Surprise	.08	.08	.19	.06	.00	.38	.21
Neutral	.05	.02	.04	.09	.13	.04	.64

(b) Random Search on Weighted Avg., (92.37, 49.49, 41.03)

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.57	.00	.03	.03	.09	.09	.19
Disgust	.12	.00	.00	.04	.50	.04	.31
Fear	.20	.04	.22	.04	.24	.13	.13
Happy	.05	.01	.00	.62	.10	.01	.21
Sad	.06	.08	.04	.09	.36	.11	.26
Surprise	.19	.04	.12	.04	.27	.19	.15
Neutral	.05	.04	.01	.03	.23	.08	.56

(c) ConvNet 1 & Audio, (-, -, 44.71*)

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
Angry	.62	.07	.03	.10	.02	.03	.12
Disgust	.12	.04	.00	.04	.50	.00	.31
Fear	.24	.00	.37	.04	.15	.09	.11
Happy	.05	.01	.00	.66	.11	.00	.17
Sad	.06	.15	.08	.11	.36	.09	.15
Surprise	.19	.04	.19	.12	.15	.15	.15
Neutral	.09	.06	.07	.03	.17	.04	.54

(d) All modalities (submission 9), (-, -, 47.67*)

Figure 5.8 Confusion matrices on the test set of AFEW2 (a-b) and AFEW4 (c-d). Accuracies for each method are specified in parentheses (training, validation & test sets, if applicable). *Model has been retained on both training and validation set prior to testing

the averaged predictions of all models, yielding **37.17**% on the test. From this analysis we also found that the exact same validation set performance was also obtained with an average not including our second convolutional network, leading us to make the conclusion that both convolutional networks were providing similar information. We thus left it out for subsequent strategies and experiments on the AFEW4.

The next highest performing simple average was 39.90% and consisted of simply combining ConvNet 1 and our audio model. Given this observation and the fact that the conference baselines included both video, audio and combined audio-video models we decided to submit a model in which we used only these two models. However, we first explored a more sophisticated way to perform this combination.

5.4.2 SVM and MLP Aggregation Techniques – AFEW2

To further boost the performance of our combined audio and video model we simply concatenated the results of our ConvNet 1 and audio model using vectors and learned a SVM with an RBF kernel using the challenge training set. The hyperparameters of the SVM were set via a two stage coarse, then fine grid search over integer powers of 10, then non-integer powers of 2 within the reduced region of space. The hyperparameters correspond to a kernel width term, γ and the c parameter of SVMs. This process yielded an accuracy of 42.17% on the validation set, which became our second submission and produced a test accuracy of **38.46**%.

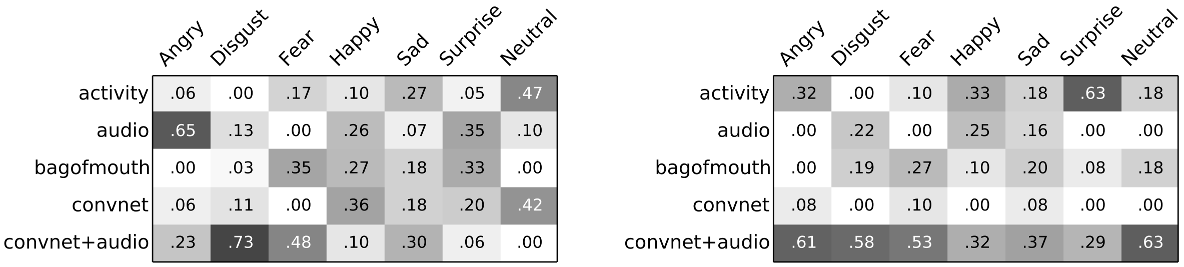
Given the success of our SVM combination strategy, we tried the same technique using the predictions of all models. However, this process quickly overfit the training data and we were not able to produce any models that improved upon our best validation set accuracy obtained via the ConvNet 1 and audio model. We observed a similar effect using a strategy based upon an MLP to combine the results of all model predictions.

We therefore tried a more sophisticated SVM hyperparameter search to re-weight different models and their predictions for different emotions. We implemented this via a search over discretized $[0, 1, 2, 3]$ per dimension scaling factors. While this resulted in 28 additional hyperparameters this discretization strategy allowed us to explore all combinations. This more detailed hyperparameter tuning did allow us to increase the validation set performance to 43.69%. This became our fourth 2013 submission; however, the strategy yielded a decreased test set performance at **32.69**%.

5.4.3 Random Search for Weighting Models – AFEW2

Recent work (Bergstra and Bengio, 2012) has shown that random search for hyperparameter optimization can be an effective strategy, even when the dimensionality of hyperparameters is moderate (ex. 35 dimensions). Analysis of our validation set confusion matrices shows that different models have very different performance characteristics across the different emotion types. We therefore formulated the re-weighting of per-model and per-emotion predictions as a hyperparameter search over simplexes, weighting the model predictions for each emotion type.

To perform the random search, we first sampled random weights from a uniform distribution and then normalized them to produce seven simplexes. This process is slightly biased towards weights that are less extreme compared to other well known procedures that are capable of generating uniform values on simplexes. After running this sampling procedure for a number of hours we used the weighting that yielded the highest validation set performance (47.98%) as our 5th 2013 submission. This yielded a test set accuracy of **39.42**%. We used the results of this initial random search to initiate a second, local search procedure which is analogous in a sense to the typical two level coarse, then fine level grid search used for SVMs. In this procedure we generated random weights using a Gaussian distribution with 0.1 standard deviation around the best weights found so far. The weights were tested by calculating the accuracy of the so-weighted average predictions on the validation set. We also rounded these random weights to 2 decimals to help to avoid overfitting on the validation set. This strategy yielded **40.06**% test set accuracy with a short duration search and **41.03**% with a longer search - our best performing 2013 submission on the test. The validation set confusion matrix for this model is shown in figure 5.8 (b) and the weights obtained through this process are shown in figure 5.9 (a).



	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
activity	.06	.00	.17	.10	.27	.05	.47
audio	.65	.13	.00	.26	.07	.35	.10
bagofmouth	.00	.03	.35	.27	.18	.33	.00
convnet	.06	.11	.00	.36	.18	.20	.42
convnet+audio	.23	.73	.48	.10	.30	.06	.00

(a) Emotiw 2013

	Angry	Disgust	Fear	Happy	Sad	Surprise	Neutral
activity	.32	.00	.10	.33	.18	.63	.18
audio	.00	.22	.00	.25	.16	.00	.00
bagofmouth	.00	.19	.27	.10	.20	.08	.18
convnet	.08	.00	.10	.00	.08	.00	.00
convnet+audio	.61	.58	.53	.32	.37	.29	.63

(b) EmotiW 2014

Figure 5.9 Final weights used for model averaging in our best submissions.

5.4.4 Strategies for the Emotiw 2014 Challenge and the AFEW4 Data

While we did not participate in the EmotiW 2014 challenge we have performed a sequence of experiments using the underlying AFEW4 dataset and the training, validation and test sets partitions defined by the challenge organizers. We have performed these experiments after the challenge period so as to explore the behaviour of our general technique as well as some different training strategies arising from the fact that the challenge is defined differently. Specifically, in the EmotiW 2014 challenge it is permitted to re-train all models using the combined training and validation set if desired. We correspondingly explored the following set of strategies.

As an initial step, we simply re-ran our best model from the 2013 competition, without retraining it on the 2014 competition dataset. Predictions of Bag-of-mouth model were replaced by uniform distribution. Our Bag-of-mouth model was trained on faces provided by the organizers which were RGB in 2013 and grayscale in 2014, this caused the model to fail on new dataset. Using models trained on AFEW2, we computed predictions on AFEW4 test set, which gave 39.80% accuracy. The 1% loss could possibly be attributed to the substitution of the Bag-of-mouth model with uniform distribution. However, sound comparison with previous results cannot be made as AFEW2 and AFEW4 test sets are different. Retraining the combination model on all models trained on AFEW2 but bag-of-mouth resulted in a lower 37.84% accuracy. We used a more aggressive random search procedure by starting hundreds of random searches with different initializations. The generalization decrease from submission 1 to 2 was most likely caused by overfitting because of this aggressive random search. Nevertheless, as AFEW4 training and validation sets are larger than their AFEW2 relatives, models trained on the latter might not be competitive in the Emotiw 2014 Challenge. Therefore, we trained our models on AFEW4 data for submission 3 to 10.

In preparation for the following sets of experiments all sub-models were trained on training set and validation set alone. They were also trained on training set combined with validation set. This yields three different sets of predictions from which one may explore and compare different training and combination strategies. Training on the training set and the validation set separately allowed us to easily do 2-fold cross-validation, while training on all data combined is a commonly used strategy to exploit all available data but can involve different techniques for setting model hyperparameters.

An SVM combination approach using all data

One simple method for learning when working with a single training set and a single validation set is to use the training set to train a model multiple times with different hyperparameters, then select the best model using the validation set. One can then simply use these hyperparameter settings and retrain the model using the combined training and validation set. This method is known to work well in practice.

We first used this method to train an SVM to combine the predictions of the ConvNet1 model and the audio model. It resulted in 44.71% test accuracy, an impressive 7% improvement over ConvNet1 alone (37.35%) and 6% improvement over the same combination trained only on the 2013 AFEW2 training set (38.26%). An important factor might be that we are using predictions on data not seen during sub-model training to train the combination model. That is, they are less biased than training predictions, which makes it possible for the SVM to generalize better. The validation set alone is, however, too small to train a good combination model.

To capitalize on this effect, we trained another SVM on swapped predictions, i.e. the predictions on the validation set came from sub-models trained on training set and predictions on training set came from sub-models trained on the validation set. An SVM was trained on both swapped sets separately to select the best hyper-parameters before training a final SVM on all swapped predictions. With 41.52% test accuracy, this model is worse then the previous one (44.71%). A possible reason for this is that the training and validation sets are unbalanced and relatively small. Good sub-models trained on the larger training set tend to generate good predictions on small validation sets, while worse sub-models trained on the small validation set generate worse predictions on the bigger training set. An obvious solution would be to generate swapped predictions in a manner similar to leave-one-out cross-validation, the drawback is that for our setting we would need to train 5 times 900 models on each fold to generate the predictions for the meta-model.

Finally, similar to section 5.4.3, we trained the SVM only on validation data. We hoped training an SVM would yield results similar to running random search. It did not. As explained in next section, running random search on the validation set predictions gives 44.72% while training an SVM on same data gives only 42.26%.

Weighting models and random search using all data

A random search procedure for determining the parameters of a linear per-class and per-model weighting was computed as described in section 5.4.3, but for the AFEW4 (EmotiW

2014 challenge data). For our first experiment we run a random search using the validation set predictions, then used the resulting weights to compute the weighted average of predictions of sub-models trained on all data. To be clear, the only difference to our best model from 2013 submissions, was that we applied the weighted average on sub-models trained on the combined training and validation set of the 2014 dataset. This yielded a test accuracy of 44.72%, 2% higher than the same procedure with SVM training, but no gain over the best combination of ConvNet1 with audio models (44.71%).

Random search can also be applied to swapped predictions such as those explained in the previous section. Running random search on such predictions gave our best results on AFEW4, 47.67%, slightly higher than the first runner up in the EmotiW 2014 competition (Sun *et al.*, 2014). The weights found through this procedure are shown in Figure 5.9 (b). A comparison of test accuracies for both the 2013 and 2014 EmotiW datasets with other methods is shown in table 5.3.

As some models were overfitting to the training data, we tried to separate overfitters from the other models and combine them together. We ran a random search on ConvNet1, Bag-of-mouth and activity recognition predictions of validation data. Then we ran a second random search on top of their weighted average with our ConvNet1+Audio SVM combination of submission 3. This final weighted average was used to compute the test predictions, giving only 42.51%.

Weights found by random search varied a lot from one run to another. We tried bagging of 350 independent weighted averages found by random searches similar to submission 9 (which obtained 47.67%). Surprisingly, the bagging approach achieved a lower accuracy of 45.45%, our second best result on AFEW4.

5.5 Conclusions and discussion

Our experiments with both competition datasets (2013 and 2014) have lead to a number of contributions and insights which we believe may be more broadly applicable. First, we

Table 5.3 Test accuracies of different approaches on AFEW2 (left) and AFEW4 (right)

Method	%
MKL (Sikka *et al.*, 2013)	35.89%
PLS (Liu *et al.*, 2013b)	34.61%
Linear SVM (Gehrig and Ekenel, 2013)	29.81%
Our method (Kahou *et al.*, 2013)	41.03%

Method	%
PLS (Liu *et al.*, 2014)	50.37%
HCF (Sun *et al.*, 2014)	47.17%
MKL (Chen *et al.*, 2014)	45.21%
Our method	47.67%

believe that our approach of using the large scale mining of imagery from Google image search to train our deep neural network has helped us to avoid overfitting to the provided challenge dataset.

We achieved better performance when we used the competition data exclusively for training the classifier and used additional face image data for training of the convolutional network. The validation set accuracy was significantly higher than in our experiment in which we trained the network directly on extracted faces from the challenge data. It is our intuition that video frames in isolation are not always representative of the emotional tag assigned to the clip, and using one label for video length introduces noise to the training set. In contrast, our additional data contained only still images with a clear correspondence between image and label. The problem of overfitting had both direct consequences on per-model performance on the validation set as well as indirect consequences on our ability to combine model predictions. Our analysis of simple model averaging showed that no combination of models could yield superior performance to an SVM applied to the outputs of our audio-video models. Our efforts to create both SVM and MLP aggregation models lead to similar observations in that models quickly overfit the training data and no settings of hyperparameters could be found which would yield increased validation set performance. We believe this is due to the fact that the activity recognition and bag of mouth models severely overfit the challenge training set and the SVM and MLP aggregation techniques - being quite flexible - overfit the data in such a way that no traditional hyperparameter tuning could yield validation set performance gains.

These observations led us to develop the novel technique of aggregating the per model and per class predictions via random search over simple weighted averages. The resulting aggregation technique is therefore of extremely low complexity and the underlying prediction was therefore highly constrained - using simple weighted combinations of complex deep network models, each of which did reasonably well at this task. We were therefore able to explore many configurations in a space of moderate dimensionality quite rapidly as we did not need to re-evaluate the predictions from the neural networks and we did not adapt their parameters. As this obtained a marked increase in performance on both the challenge validation and test sets, it lead us to the following interpretation: Given the presence of models that overfit the training data, it may be better practice to search a moderate space of simple combination models. This is in contrast to traditional approaches such as searching over the smaller space of SVM hyperparameters or even a moderately sized space of traditional MLP hyperparameters including the number of hidden layers and the number of units per layer.

Acknowledgements

The authors would like to thank the developers of Theano (Bastien *et al.*, 2012; Bergstra *et al.*, 2010). We thank NSERC, Ubisoft, the German BMBF, project 01GQ0841 and CIFAR for their support. We also thank Abhishek Aggarwal, Emmanuel Bengio, Jörg Bornschein, Pierre-Luc Carrier, Myriam Côté, Guillaume Desjardins, David Krueger, Razvan Pascanu, Jean-Philippe Raymond, Arjun Sharma, Atousa Torabi, Zhenzhou Wu, and Jeremie Zumer for their work on the 2013 submission.

Reference

SE Kahou, X Bouthillier, P Lamblin, C Gulcehre, V Michalski, K Konda, S Jean, P Froumenty, Y Dauphin, N Boulanger-Lewandowski, RC Ferrari, M Mirza, D Warde-Farley, A Courville, P Vincent, R Memisevic, C Pal, Y Bengio, “EmoNets: Multimodal deep learning approaches for emotion recognition in video”, In Journal on Multimodal User Interfaces (JMUI), (Kahou *et al.*, 2015a)

CHAPTER 6 ARTICLE 2: RECURRENT NEURAL NETWORKS FOR EMOTION RECOGNITION IN VIDEO

Proceedings of the 2015 ACM on International Conference on Multimodal Interaction, November 2015

Samira Ebrahimi Kahou, Vincent Michalski, Kishore Konda, Roland Memisevic and Christopher Pal

Abstract

Deep learning based approaches to facial analysis and video analysis have recently demonstrated high performance on a variety of key tasks such as face recognition, emotion recognition and activity recognition. In the case of video, information often must be aggregated across a variable length sequence of frames to produce a classification result. Prior work using convolutional neural networks (CNNs) for emotion recognition in video has relied on temporal averaging and pooling operations reminiscent of widely used approaches for the spatial aggregation of information. Recurrent neural networks (RNNs) have seen an explosion of recent interest as they yield state-of-the-art performance on a variety of sequence analysis tasks. RNNs provide an attractive framework for propagating information over a sequence using a continuous valued hidden layer representation. In this work we present a complete system for the 2015 Emotion Recognition in the Wild (EmotiW) Challenge. We focus our presentation and experimental analysis on a hybrid CNN-RNN architecture for facial expression analysis that can outperform a previously applied CNN approach using temporal averaging for aggregation.

6.1 Introduction

Human emotion analysis is a challenging machine learning task with a wide range of applications in human-computer interaction, e-learning, health care, advertising and gaming. Emotion analysis is particularly challenging as multiple input modalities, both visual and auditory, play an important role in understanding it. Given a video sequence with a human subject, some of the important cues which help to understand the user's emotion are facial expressions, movements and activities. In some cases speech or high level scene context can also be useful to infer emotion. Most of the time there is a considerable overlap between emotion classes making it a challenging classification task. In this paper we present a deep learning based approach to modeling different input modalities and to combining them in order to infer emotion labels from a given video sequence.

The Emotion recognition in the wild (EmotiW 2015) challenge (Dhall *et al.*, 2015) is an extension of a similar challenge held in 2014 (Dhall *et al.*, 2012). The task is to predict one of seven emotion labels: angry, disgust, fear, happy, sad, surprise and neutral. The dataset used in the challenge is the Acted Facial Expressions in the Wild (AFEW) 5.0 dataset, which contains short video clips extracted from Hollywood movies. The video clips present emotions with a high degree of variation, e.g. actor identity, age, pose and lighting conditions. The dataset contains 723 videos for training, 383 for validation and 539 test clips.

Traditional approaches to emotion recognition were based on hand-engineered features (Kahou *et al.*, 2014; Shan *et al.*, 2009). With the availability of big datasets, deep learning has emerged as a general approach to machine learning yielding state-of-the-art results in many computer vision and natural language processing tasks (Krizhevsky *et al.*, 2012; Kalchbrenner *et al.*, 2014). The basic principle of deep learning is to learn hierarchical representations of input data such that the learned representations improve classification performance.

The primary contribution of this work is to model the spatio-temporal evolution of facial expressions of a person in a video using a Recurrent Neural Network (RNN) combined with a Convolutional Neural Network (CNN) in an underlying CNN-RNN architecture. In addition to this, we also employed an Autoencoder based activity recognition pipeline for modelling user activity and a simple Support Vector Machine (SVM) based approach over energy and spectral features for audio. We also present a neural network-based feature level fusion technique to combine different modalities for the final emotion prediction for a short video clip.

Previous work (Kahou *et al.*, 2013; Liu *et al.*, 2014) has achieved state-of-the-art results in the emotion recognition challenge using deep learning techniques which includes our work that won the 2013 EmotiW challenge. In contrast to Kahou *et al.*; Kahou *et al.*, which use an averaging-based aggregation method for visual features in video, here we employ an RNN to model the temporal evolution of facial features in video. We also explore feature-level fusion of our modality-specific models and show that this increases performance.

The remainder of this paper is organized as follows. In Section 6.2, 6.3 and 6.4 we describe each of the models used for different modalities followed by Section 6.5, which provides details on the fusion methods we applied. Section 6.6 presents our experimental results and provides a list of our submissions to the challenge. Finally, in Section 6.7 we draw some conclusions from our experiments.

6.2 Spatio-temporal evolution of facial expressions

Modelling the spatio-temporal evolution of visual information plays an important role in understanding the behavior of objects and users in video. Emotion recognition is one of the tasks which involve modelling the behavior of a user. In this work, we use a two step approach to modelling emotion as the spatio-temporal evolution of image structure. In the first step, an CNN is trained to classify static images containing emotions. In the second step, we train an RNN on the higher layer representation of the CNN inferred from individual frames to predict a single emotion for the entire video. RNNs have undergone a resurgence of interest due in part to their impressive performance in handwriting and speech recognition (Graves and Schmidhuber, 2009; Graves *et al.*, 2013). Much of this interest has been driven by the stability of learning achieved by the use of so-called long short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). RNNs have also proven to be powerful methods for other types of sequential data including video (Baccouche *et al.*, 2011; Donahue *et al.*, 2015) and natural language processing (Bahdanau *et al.*, 2014; Sutskever *et al.*, 2014). As such we use an RNN structure for learning a model for video level representation and classification. The higher layer representation from the CNN provides structural information of a given frame and the RNN models the spatio-temporal evolution of the structure over time.

Unlike other work involving video and RNN techniques such as Baccouche *et al.*; Donahue *et al.*, we do not use LSTMs. Here we use IRNNs (Le *et al.*, 2015) which are composed of rectified linear units (ReLUs) and employ a special initialization strategy based on scaled variations of the identity matrix. These elements of IRNNs are aimed at providing a much simpler mechanism for dealing with the vanishing and exploding gradient problem compared to the more complex LSTM framework. Recent work has compared IRNNs with LSTMs and found that IRNNs are able to yield comparable results in some tasks, including problems which involve long term dependencies (Le *et al.*, 2015).

We provide a detailed explanation of the CNN structure in Section 6.2.1 and of the RNN in Section 6.2.2. To compare with the non-sequential approach presented in Kahou *et al.*, we also aggregated CNN features to a fixed-length feature vector and trained an SVM. This is described in Section 6.2.3.

6.2.1 Frame feature extraction using an CNN

The competition dataset has one emotion label per video which does not correspond to every frame. This introduces a lot of noise if the video labels are used as targets for training an CNN on individual frames. Our visual features are therefore provided by an CNN trained on a combination of two additional emotion datasets of static images. Moreover, using additional data covers a larger variety in age and identity in contrast to the challenge data where the same actor/actress might appear in multiple clips.

Datasets

The additional datasets used in the CNN training consists of two large emotion datasets, namely the Toronto Face Database (TFD) (Susskind *et al.*, 2010) with 4,178 images and the Facial Expression Recognition dataset (FER2013) (Carrier *et al.*, 2013) containing 35,887 images, both with seven basic expressions: angry, disgust, fear, happy, sad, surprise and neutral.

Pre-processing

To account for varying lighting conditions (in particular, across datasets) we applied histogram equalization. We used the aligned faces provided by the organizers to extract features from the CNN. The alignment involves a combined facial keypoints detection and tracking approach explained in Dhall *et al.*. We shall refer to this dataset as AFEW-faces. Different face detection and/or alignment techniques have been used for FER2013, TFD and AFEW-faces. In order to be able to leverage the additional datasets, we re-aligned all datasets to FER2013 using the following procedure:

1. We detected five facial keypoints for all images in the FER2013, TFD and AFEW-faces training set using the convolutional neural network cascade method in Sun *et al.*.

2. For each dataset we computed the mean shape by averaging the coordinates of keypoints.

3. Datasets have been mapped to FER2013 by using a similarity transformation between mean shapes. By computing one transformation per dataset we let the eyes, nose and mouth be roughly in the same location retaining a slight amount of variation. We added a noisy border for TFD and AFEW-faces as faces were cropped more tightly compared to FER2013.

4. AFEW-faces validation and test sets were mapped using the transformation inferred on the training set.

We also performed dataset normalization with the mean and standard deviation image from the combined FER2013 and TFD (FER+TFD).

CNN Architecture

We trained various CNN architectures on FER+TFD without using any challenge data for gradient computations. For early stopping we tried both leaving out 1000 samples of FER+TFD and the challenge data. We observed that the RNN yields slightly better performance when CNN early stopping was done on the challenge data as this avoids over-fitting to FER+TFD. Therefore, for our best CNN structure, we trained on all FER+TFD and performed early stopping on AFEW-faces train+validation.

We have explored three main CNN structures:

- a very deep structure with small 3x3 filter size (Nagadomi, 2014; Simonyan and Zisserman, 2014),
- a three-layer CNN with 5x5 filters (Krizhevsky, 2012; Krizhevsky *et al.*, 2012) and
- a similar three-layer CNN with 9x9 filter size.

The CNN is trained mainly for feature extraction and we have only used the additional dataset for the training phase. Therefore, we searched for a structure that better generalizes to other datasets. Deep structures are known to learn representations that better generalize to other datasets (Simonyan and Zisserman, 2014). However, we observed that the very deep structure quickly over-fitted to FER+TFD, and generalized badly to the challenge dataset. This could be due to the relatively small amount of labeled data available for the emotion recognition task here. For this reason we have tried a shallower network with three layers which appears to have moderately addressed the over-fitting problem. Finally, we increased the filter size from 5 to 9 and reduced the number of filters from 64-64-128 to 32-32-64. For all of the experiments we used data augmentation (horizontal flipping with probability of 0.5 and random cropping), as well as dropout (with rate 0.25).

6.2.2 Learning Sequences Using an RNN

We use an RNN to aggregate frame features for the following reasons:

- The temporal order of frames is respected in contrast to bag-of-features approaches.
- An RNN has the ability to learn to detect an event, such as the presence of a particular expression, irrespective of the time, at which it occurs in a sequence.
- RNNs naturally deal with a variable number of frames.

RNNs are a type of neural network which transforms a sequence of inputs into a sequence of outputs. At each time-step t, a hidden state $\mathbf{h}_t$ is computed based on the hidden state at time $t-1$ and the input $\mathbf{x}_t$ at time t

$$\mathbf{h}_t = \sigma(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{rec}\mathbf{h}_{t-1}), \tag{6.1}$$

where $\mathbf{W}_{in}$ is the input weight matrix, $\mathbf{W}_{rec}$ is the recurrent matrix and σ is the hidden activation function. Each time-step also computes outputs, based on the current hidden state:

$$\mathbf{y}_t = f(\mathbf{W}_{out}\mathbf{h}_t), \tag{6.2}$$

where $\mathbf{W}_{out}$ is the output weight matrix and f is the output activation function. An example of an RNN in which only the last time-step produces an output is shown in Figure 6.1.

We use the IRNN, which as discussed above is a simple RNN with rectified linear hidden units (ReLUs) and with a recurrent matrix, that is initialized with scaled variations of the identity matrix (Le *et al.*, 2015). The identity initialization trick ensures good gradient flow at the beginning of training and it allows us to train it on relatively long sequences.

We train the IRNN to classify a video by feeding the features for each frame from the CNN sequentially to the network and using the last time-step softmax output as class prediction. We used Stochastic Gradient Descent (SGD) with a learning rate of 0.005, gradient clipping (Bengio *et al.*, 2013) at 1.0 and a batchsize of 64 sequences. We experimented by using different layers of the CNN as input features and chose the output of the second convolutional layer after max pooling, as this performed best on validation data.

6.2.3 Aggregated CNN Features

As an alternative way of aggregating the frame level structural representations from the CNN, we employed k-average pooling together with an SVM for classification as in Kahou *et al.*. In this approach the per-frame CNN features are averaged into bins to generate a fixed length vector of size k as video representation. Heuristically, we selected $k = 15$ and we used the pre-softmax outputs of the CNN as per-frame features. For videos with a number of frames

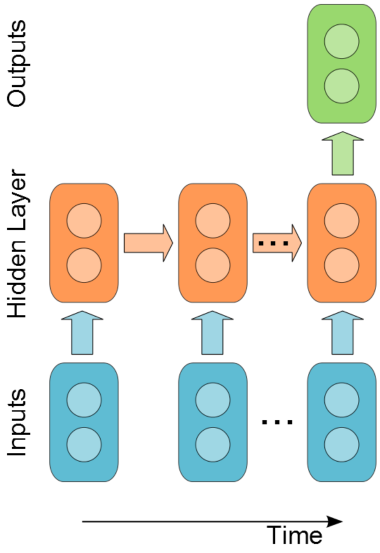


Figure 6.1 Structure of our recurrent neural network.

less than k the frames are locally repeated until sequence length is k.

The vector representations of videos together with corresponding emotion labels are used to train an RBF-kernel SVM. The hyper-parameters of the SVM are set via grid search. As shown in Table 6.1, the RNN achieves a validation accuracy of 39.6%, which is significantly higher than the aggregated CNN. Simple averaging of the per-frame probabilities yielded a validation accuracy of only 23.7%.

6.3 Audio

Given that the primary focus of this work is on vision based emotion recognition, we simply used the audio features employed in Dhall *et al.* for the audio channel of the video clips. These are based on the approach from Schuller *et al.*. It uses 1582 features extracted with the open-source Emotion and Affect Recognition (openEAR) (Eyben *et al.*, 2009) toolkit which uses openSMILE (Eyben *et al.*, 2010) as backend.

The toolkit encapsulates multiple low level audio feature descriptors (LLDs) and different functionals to apply on them. The feature set consists of 34 energy and spectral related LLDs and 21 functionals, 4 voicing related LLD × 19 functionals, 34 delta coefficients of energy and spectral LLD × 21 functionals, 4 delta coefficients of the voicing related LLD × 19 functionals and 2 voiced/unvoiced durational features.

In this work we used Principal Component Analysis (PCA) based dimensionality reduction as preprocessing on the 1582 dimensional input features and an RBF-kernel SVM for classification. The hyper-parameters for the SVM are set via grid search.

6.4 Activity

Spatio-temporal transformations of local image features, or activity, can be an important cue for emotion recognition. A subset of emotions can be represented as changes in facial expressions and in some cases the activity of the entire body of the person. Other approaches, based on vision, described in this work mainly deal with analyzing the emotion in a given video sequence based on static image features and different ways of aggregating them over time. The activity analysis pipeline is the only approach which relies on learning of local spatio-temporal transformations from video.

Our approach for activity analysis is based on the action recognition pipeline from Konda *et al.* (2014b); Le *et al.* (2011) which was also used for emotion recognition previously in Kahou *et al.* (2015a). The pipeline mainly consists of three different modules namely, local motion feature extraction, k-means quantization and SVM based classification. A Synchrony Autoencoder (SAE) (Konda *et al.*, 2014b) trained on cropped 3D video blocks of size $16 \times 16 \times 10$ ($space \times space \times time$) is used for local motion feature extraction. Figure 6.2 shows filters learned by the model on the AFEW 5.0 training set.

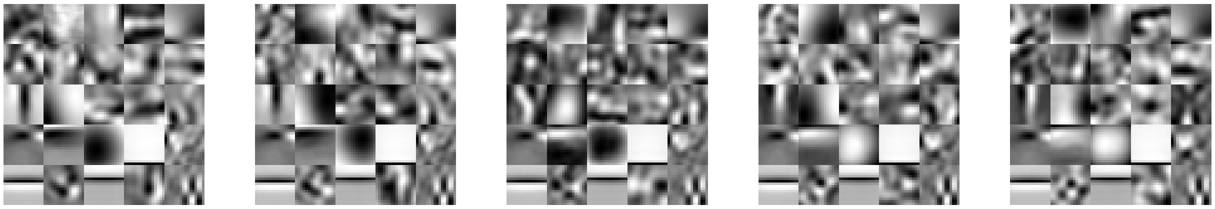


Figure 6.2 Subset of filters learned by SAE model on the AFEW5 training set. Left to right: Frames 1,3,5,7 and 9.

6.5 Fusion

In many discriminative tasks, the fusion of predictions or representations from models trained using different input modalities yields a significant improvement. We use two types of fusion approaches for combining the modality specific models described in previous sections, *feature level* and *decision level* fusion.

6.5.1 Feature Level

In this approach a combination of intermediate-level representations from the trained models is used as input for training an additional model on the classification task. For feature-level fusion we applied a variant of the regularized feature fusion network from Wu *et al.* (2015b). The feature fusion network is a Multilayer Perceptron (MLP) with separate hidden layers for each modality as shown in Figure 6.3. The outputs of these layers are concatenated and fed to another hidden layer which is followed by a softmax layer whose number of units is equal to the number of emotion classes. The first layer of the fusion network, consisting of modality specific layers, is regularized to encourage a common representation by sharing similar subsets of hidden units between modalities, while still retaining the discriminative features present in some modalities.

The network is trained with SGD using a learning rate of 0.1 and gradient clipping using clipping threshold 10. The objective function is the categorical cross-entropy between target label and prediction. As input to the fusion network we used aggregated CNN per-frame features, the PCA-whitened audio features and the hidden layer activations of the last time-step of the RNN. We excluded the activity recognition model from the mix, as it tends to over-fit its training set. We also explored adding dropout to the hidden layers to prevent over-fitting on the small challenge dataset. The number of hidden layers and their sizes are selected using the validation set. Our best architecture has 100, 10 and 50 units in the aggregated CNN-, the audio- and the RNN-specific hidden layers, respectively. The common hidden layer has 70 units. The search space for determining the optimal size of the modality-specific layers was selected considering the input feature sizes and individual models' performances on the AFEW 5.0 validation set while training on the train set. More details are provided in Section 6.6.

6.5.2 Decision Level

For decision-level fusion, i.e. the combination of classifiers, we used a weighted sum of the class probabilities estimated by the modality-specific classifiers and the fusion network. The combined classifier has one weight per modality per class and the resulting score for each class is the weighted sum of all probabilities for the respective class. The combination weights are determined by random search (Bergstra and Bengio, 2012), which was also used for model combination in the winning approach for the 2013 EmotiW challenge (Kahou *et al.*, 2013).

Weights are sampled uniformly from $[0.0, 1.0]$ followed by per class re-scaling, so that they sum up to 1. Then the best sampled weights are chosen based on the validation performance.

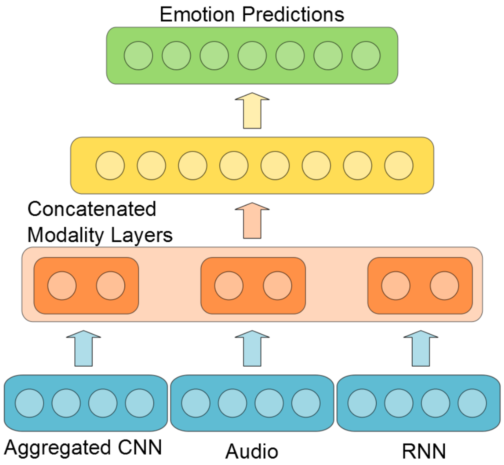
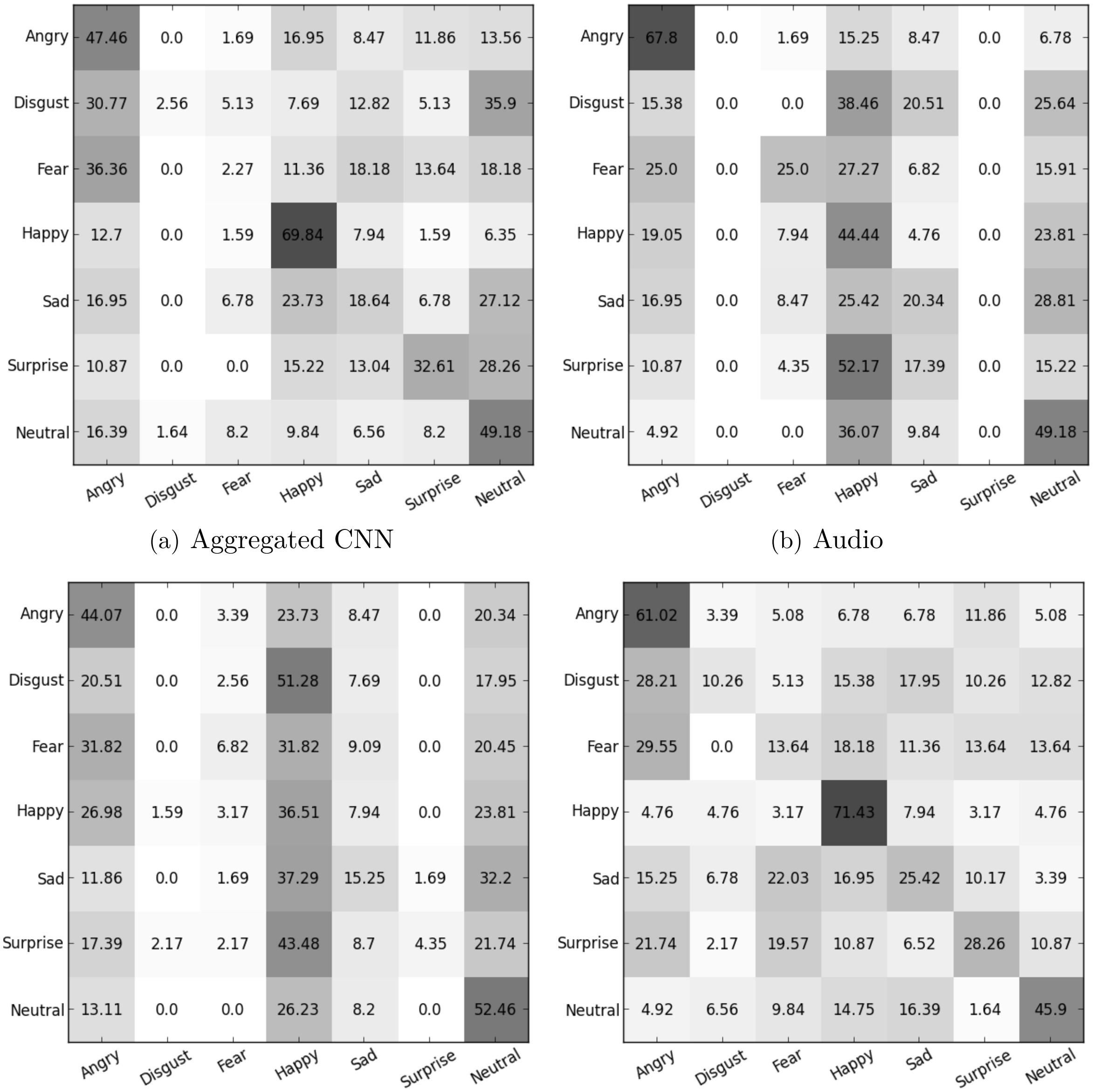


Figure 6.3 Structure of the feature fusion network.

Note that unless noted otherwise, we always use the dataset partition for the random search which was not used for model training, i.e. for models trained on the training set, we perform random search on the validation set and vice versa. After an initial random search with 100,000 iterations, we perform a local random search around the best set of weights found so far. This local random search consists of sampling weights from a Gaussian with mean set to the current best set of weights and standard deviation σ of 0.5. The current best $\tilde{w}$ is updated as soon as a new best is found. After every 100,000 iterations, the σ is decreased by a factor of 0.9 and the local search is stopped when σ is smaller than 0.0001. We also performed uniform local search from $[\tilde{w} - r, \tilde{w} + r]$, where $\tilde{w}$ is the current best set of weights and r is the range in which to search, however it roughly achieved the same performance. We explicitly tried all combinations of subsets of modalities and fusion. Consistently we found that decision level fusion benefited from including all models.

Table 6.1 Training and Validation Accuracies for All Modalities (Training on Train partition)

Model	Training	Validation
Activity	0.983	0.266
Audio	0.418	0.332
Aggregated CNN	0.505	0.350
RNN	0.848	**0.396**



(a) Aggregated CNN

(b) Audio

(c) Activity

(d) RNN

Figure 6.4 Confusion matrices on the challenge validation set.

Table 6.2 Our submissions with training, validation and test accuracies (in percent) for the EmotiW 2015 competition (bold font shows the best accuracy)

Sub.	Train	Valid	Test	Method
1	86.216	54.716	44.341	Training on Train, Validation on Valid
2	81.997	54.447	48.979	Training on Train, Validation on Valid, stable fusion
3	-	-	50.463	Training on Train+Val, hyperparams from submission 2, stable fusion
4	52.320	71.967	50.092	Training on Val, Validation on Train
5	52.742	68.463	47.680	Training on Val, Validation on Train, stable fusion
6	-	-	**52.875**	Training on Train+Val, hyperparams from submission 4
7	-	-	49.907	Random Search over combinations of submission 3 and 6 on Train+Val

6.6 Results

In this section, we describe our submissions to the EmotiW 2015 challenge. We provide details on per model training strategies and variations of our fusion methods. We also present results and discuss the choices we made in each step.

6.6.1 Per-model Performance

This work mainly focuses on an RNN approach for visual features. However, given the challenge context we included three further models to achieve competitive performance. Table 6.1 shows each model's accuracy on the challenge validation set after training on the training set. The corresponding confusion matrices are presented in Figure 6.4. The matrices show different profiles and strengths for specific emotion classes which is beneficial for combination.

6.6.2 Feature Level Fusion

As mentioned before, we excluded the activity model from feature level fusion as it tends to over-fit on its training partition. This can be seen in Table 6.1 where activity has an extremely high discrepancy between training and validation accuracies. The input features to the fusion network are the following:

- The first ten components of the PCA whitened audio features (see Section 6.3).
- The aggregated CNN features, which are 105-dimensional (7×15 bins) vectors as described in Section 6.2.3.

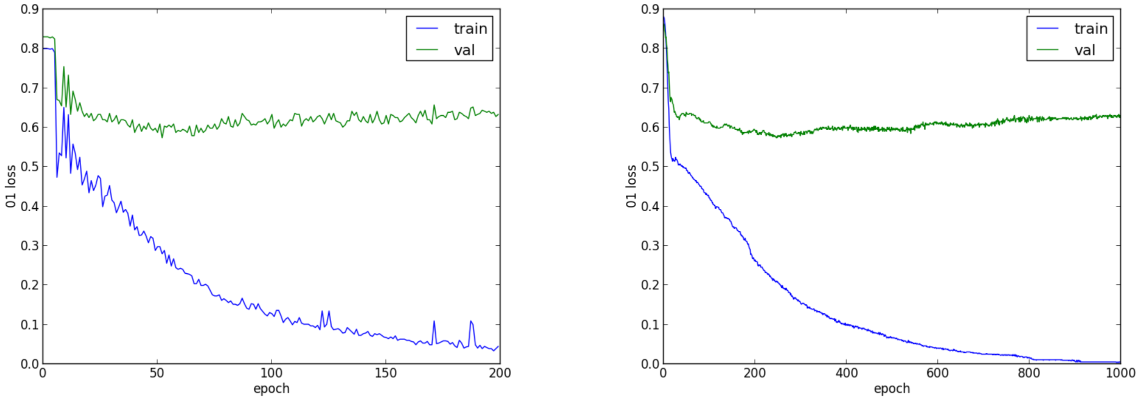
- The RNN features, which are the hidden activations of the last time step. These are the only features which have been learned discriminatively on the video level and which therefore contribute strongly to the fusion network. The number of hidden units in the RNN is 200 (see Section 6.2.2).

For training the fusion network, we tried replacing the sigmoid activation function of the hidden layers with rectified linear units $ReLU(x) = max(0, x)$ and rectified tanh units $RectTanh(x) = max(0, tanh(x))$. While this improved the validation performance by roughly 2%, it did not yield an improvement on the test performance. One observation during training was that the learning curves were oscillating which made the early stopping unreliable. To stabilize the learning, we lowered the learning rate to 0.001 from 0.1 and added momentum of 0.9. Figure 6.5 compares two learning curves before and after stabilization. The number of epochs in each sub-figure corresponds to the selected learning rate. Our fusion network achieves a validation accuracy of 43.7%, which is higher than any modality-specific classifier.

6.6.3 Submissions

Our submissions can be divided into two categories: those which use the training set for training and the validation set for early stopping and random search and those for which the training and validation sets were swapped. For both of these categories, we also submitted a version where models were retrained on the full training plus validation set, retaining all hyper-parameters including early stopping epoch number and combination weights. Note that the models that are CNN-based have also been retrained but not the underlying CNN as we used additional static emotion data for training. For all submissions, random search was done on the data partition that was not used for training the underlying models. For example, if models were trained on the training partition, random search was performed on the validation set. Searching on the same partition that the models were trained on was not an option, as random search would assign high weights to the over-fitters, which would result in poor generalization performance.

Table 6.2 lists our submissions with their training, validation and test accuracies. In the first category we trained modality-specific models and the fusion network on the challenge training data and validation data was used for early stopping. Then for the final predictions we performed random search on the validation set. This achieved a test set accuracy of 44.341%. With the stabilized fusion network the accuracy improved to 48.979%. Retraining the models with the combined training plus validation set, keeping the hyper-parameters of experiment 2, yielded a test accuracy of 50.463%.



(a)

(b)

Figure 6.5 Comparison of the learning curves (a) before and (b) after stabilization.

In the second category, with swapped training and validation sets, our initial submission achieved a test accuracy of 50.092%. Here the stabilized fusion did not improve the performance, yielding a test accuracy of 47.680%. The retrained version achieved our best result of 52.875%. Note that for each category we picked the best submission for retraining. Random search as the last step in our pipeline has a big influence on the generalization potential of the whole model and likely benefits from the larger training set. This explains the higher performance of the swapped partitions.

Our last submission was an attempt to combine our two best submissions that were retrained on the training plus validation set. We combined those two using the same decision-level fusion strategy as before. The inputs to the random search were the probabilities predicted by the two models. A random search on these two models was performed on the full training plus validation set. The resulting test performance was only 49.907%. This might be explained by the fact that the whole data set has been seen which could have led to over-fitting.

6.7 Conclusions

We found that the spatio-temporal evolution of facial features is one of the strongest cues for emotion recognition. We presented the application of an RNN for modelling this spatio-temporal evolution via aggregation of facial features to perform emotion recognition in video. Our experiments in Section 6.2.3 have shown that this approach outperforms all other modalities, the averaging of per-frame vision-based classifications, and also the more sophisticated aggregation method employed by the 2013 challenge winners (Kahou *et al.*, 2013).

Furthermore, we explore two fusion methods, operating on the feature and on the decision level. Our feature-level fusion network combines features from different modalities and achieves a higher validation accuracy than any of the single-modality classifiers. Our experiments show that feature-level and decision-level fusion are complementary, and when combined they achieve a higher classification accuracy. However, care needs to be taken to prevent over-fitting, either by excluding strong over-fitters, as we did with the activity recognition model in the fusion network, or by using different dataset partitions for combination than for model training, as done in the random search.

We found it difficult to draw conclusions from some of our submission results. This might be caused by the large number of ambiguous cases that exist in this domain. We found that a fairly large number of training videos could be argued to show a mixture of two or more basic emotions (such as a mixture of surprise with fear or happy). This suggests that exploring the use of more than a single label for emotion recognition might be a useful direction for future research.

Acknowledgments

The authors would like to thank the developers of Theano (Bastien *et al.*, 2012; Bergstra *et al.*, 2010). This work was supported by an NSERC Discovery Award and the German BMBF, project 01GQ0841. We also thank the Canadian Foundation for Innovation (CFI) for support under the Leaders program.

Reference

SE Kahou, V Michalski, K Konda, R Memisevic, C Pal, “Recurrent Neural Networks for Emotion Recognition in Video”, In proceedings of the 17th ACM International Conference on Multimodal Interaction (ICMI ’15), (Ebrahimi Kahou *et al.*, 2015)

CHAPTER 7 ARTICLE 3: RATM: RECURRENT ATTENTIVE TRACKING MODEL

Submitted to Transactions on Pattern Analysis and Machine Intelligence, April 2016

Samira Ebrahimi Kahou, Vincent Michalski, Roland Memisevic and Christopher Pal

Abstract

We present an attention-based modular neural framework for computer vision. The framework uses a soft attention mechanism allowing models to be trained with gradient descent. It consists of three modules: a recurrent attention module controlling *where* to look in an image or video frame, a feature-extraction module providing a representation of *what* is seen, and an objective module formalizing *why* the model learns its attentive behavior. The attention module allows the model to focus computation on task-related information in the input. We apply the framework to several object tracking tasks and explore various design choices. We experiment with three data sets, bouncing ball, moving digits and the real-world KTH data set. The proposed Recurrent Attentional Tracking Model (RATM) performs well on all three tasks and can generalize to related but previously unseen sequences from a challenging tracking data set.

7.1 Introduction

Attention mechanisms are one of the biggest trends in deep-learning research and have been successfully applied in a variety of neural-network architectures across different tasks. In computer vision, for instance, attention mechanisms have been used for image generation (Gregor *et al.*, 2015) and image captioning (Xu *et al.*, 2015). In natural language processing they have been used for machine translation (Bahdanau *et al.*, 2014) and sentence summarization (Rush *et al.*, 2015). And in computational biology attention was used for subcellular protein localization (Sønderby *et al.*, 2015b).

In these kinds of applications usually not all information contained in the input data is relevant for the given task. Attention mechanisms allow the neural network to focus on the relevant parts of the input, while ignoring other, potentially distracting, information. Besides enabling models to ignore distracting information, attention mechanisms can be helpful in streaming data scenarios, where the amount of data per frame can be prohibitively large for full processing. In addition, some studies suggest that there is a representational advantage of sequential processing of image parts over a single pass over the whole image (Mnih *et al.*, 2014; Larochelle and Hinton, 2010; Gregor *et al.*, 2015; Denil *et al.*, 2011; Ranzato, 2014; Sermanet *et al.*, 2014, see for example).

Recently, Gregor *et al.* (2015) introduced the Deep Recurrent Attentive Writer (DRAW), which involves a Recurrent Neural Network (RNN) that controls a read and a write mechanism based on attention. The read mechanism extracts a parametrized window from the static input image. Similarly, the write mechanism is used to write into a window on an output canvas. This model is trained to sequentially produce a reconstruction of the input image on the canvas. Interestingly, one of the experiments on handwritten digits showed that the read mechanism learns to trace digit contours and the write mechanism generates digits in a continuous motion. This observation hints at the potential of such mechanisms in visual object tracking applications, where the primary goal is to trace the spatio-temporal "contours" of an object as it moves in a video.

Previous work on the application of attention mechanisms for tracking includes Denil *et al.* (2011) and references therein. In contrast to that line of work, we propose a model based on a fully-integrated neural framework, that can be trained end-to-end using back-propagation. The framework consists of three modules: a recurrent differentiable attention module controlling *where* to look in an image, a feature-extraction module providing a representation of *what* is seen, and an objective module formalizing *why* the model learns its attentive behavior. As we shall show, a suitable surrogate cost in the objective module can provide a supervised learning signal, that allows us to train the network end-to-end, and to learn attentional strategies using simple supervised back-prop without resorting to reinforcement learning or sampling methods.

According to a recent survey of tracking methods (Smeulders *et al.*, 2014), many approaches to visual tracking involve a search over multiple window candidates based on a similarity measure in a feature space. Successful methods involving deep learning, such as Nam and Han (2015), perform tracking-by-detection, e.g. by using a CNN for foreground-background classification of region proposals. As in most approaches, the method in Nam and Han (2015) at each time step samples a number of region proposals (256) from a Gaussian distribution centered on the region of the previous frame. Such methods do not benefit from useful correlations between the target location and the object's past trajectory. There are deep-learning approaches that consider trajectories by employing particle filters such as Wang and Yeung (2013), which still involves ranking of region proposals ($1,000$ particles).

In our RATM, an RNN predicts the position of an object at time t, given a real-valued hidden state vector. The state vector can summarize the history of observations and predictions of previous time steps. We rely on a *single* prediction per time step instead of using the predicted location as basis for a search over multiple region proposals. This allows for easy integration of our framework's components and training with simple gradient-based methods.

The main contribution of our work is the introduction of a modular neural framework, that can be trained end-to-end with gradient-based learning methods. Using object tracking as an example application, we explore different settings and provide insights into model design and training. While the proposed framework is targeted primarily at videos, it can also be applied to sequential processing of still images.

7.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are powerful machine learning models that are used for learning in sequential processing tasks. Advances in understanding the learning dynamics of RNNs enabled their successful application in a wide range of tasks (Hochreiter and Schmidhuber, 1997; Pascanu *et al.*, 2012; Graves *et al.*, 2013; Sutskever *et al.*, 2014; Cho *et al.*, 2014; Srivastava *et al.*, 2015, for example). The standard RNN model consists of an input, a hidden and an output layer as illustrated in Figure 7.1.

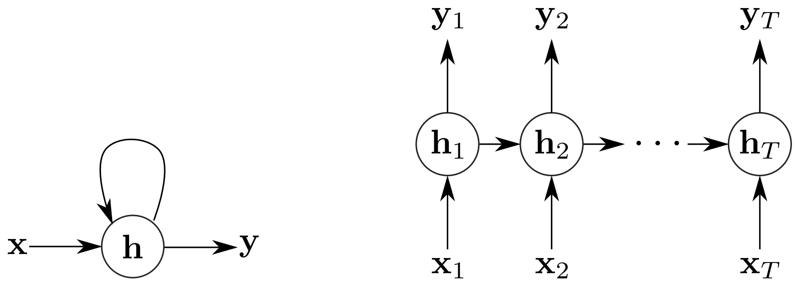


Figure 7.1 Left: A simple recurrent network with one input, one output and a hidden layer, which has a feed-back loop connection. Right: The same network unrolled in time for T time steps.

In each time step t, the network computes a new hidden state $\mathbf{h}_t$ based on the previous state $\mathbf{h}_{t-1}$ and the input $\mathbf{x}_t$:

$$\mathbf{h_t} = \sigma(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{W}_{rec}\mathbf{h}_{t-1}), \tag{7.1}$$

where σ is a non-linear activation function, $\mathbf{W}_{in}$ is the matrix containing the input-to-hidden weights and $\mathbf{W}_{rec}$ is the recurrent weight matrix from the hidden layer to itself. At each time step the RNN also generates an output

$$\mathbf{y}_t = \mathbf{W}_{out}\mathbf{h}_t + \mathbf{b}_y, \tag{7.2}$$

where $\mathbf{W}_{out}$ is the matrix with weights from the hidden to the output layer.

Although the application of recurrent networks with sophisticated hidden units, such as LSTM (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) (Cho *et al.*, 2014), has become common in recent years (Bahdanau *et al.*, 2014; Sutskever *et al.*, 2014; Srivastava *et al.*, 2015, for example), we rely on the simple IRNN proposed by Le *et al.* (2015), and show that it works well in the context of visual attention. The IRNN corresponds to a standard RNN, where recurrent weights $\mathbf{W}_{rec}$ are initialized with a scaled version of the identity matrix and the hidden activation function $\sigma(.)$ is the element-wise ReLU function (Nair and Hinton, 2010)

$$\sigma(x) = max(0, x). \tag{7.3}$$

The initial hidden state $\mathbf{h}_0$ is initialized as the zero vector. Our experiments are based on the Theano (Team *et al.*, 2016) implementation of the IRNN shown to work well for video in Ebrahimi Kahou *et al.* (2015).

7.3 Neural Attention Mechanisms

Our attention mechanism is a modification of the read mechanism introduced in Gregor *et al.* (2015). It extracts *glimpses* from the input image by applying a grid of two-dimensional Gaussian window filters. Each of the filter responses corresponds to one pixel of the glimpse. An example of the glimpse extraction is shown in Figure 7.2.

Given an image $\mathbf{x}$ with A columns and B rows, the attention mechanism separately applies a set of M column filters $\mathbf{F}_X \in \mathbb{R}^{M\times A}$ and a set of N row filters $\mathbf{F}_Y \in \mathbb{R}^{N\times B}$, extracting an $M \times N$ glimpse $\mathbf{p}$:

$$\mathbf{p} = \mathbf{F}_Y\mathbf{x}\mathbf{F}_X^{\mathrm{T}}. \tag{7.4}$$

This implicitly computes $M \times N$ two-dimensional filter responses due to the separability of two-dimensional Gaussian filters. For multi-channel images the same filters are applied to each channel separately.

The sets of one-dimensional row ($\mathbf{F}_Y$) and column ($\mathbf{F}_X$) filters have three parameters each[1]:

[1] The original read mechanism in Gregor *et al.* (2015) also adds a scalar intensity parameter γ, that is multiplied to filter responses.

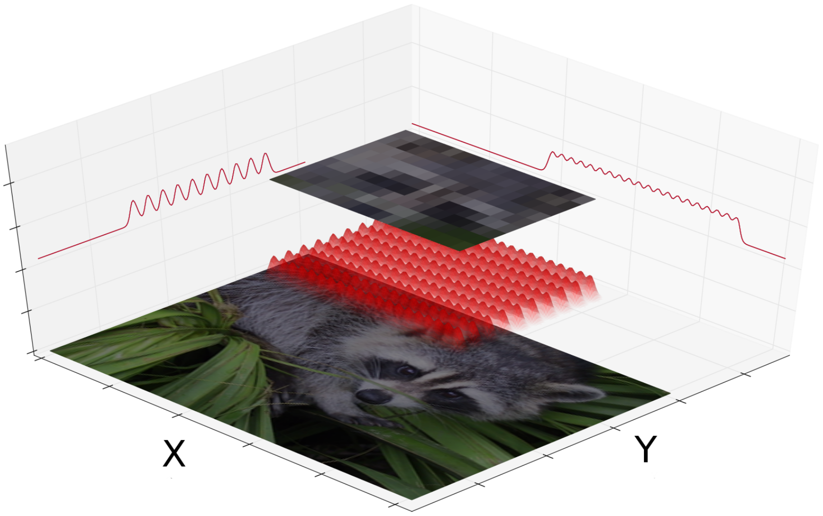


Figure 7.2 A 20 × 10 glimpse is extracted from the full image by applying a grid of 20 × 10 two-dimensional Gaussian window filters. The separability of the multi-dimensional Gaussian window allows for efficient computation of the extracted glimpse.

- the grid center coordinates g_X, g_Y,
- the standard deviation for each axis σ_X, σ_Y and
- the stride between grid points on each axis δ_X, δ_Y.

These parameters are dynamically computed as an affine transformation of a vector of activations $\mathbf{h}$ from a neural network layer:

$$(\tilde{g}_X, \tilde{g}_Y, \tilde{\sigma_X}, \tilde{\sigma_Y}, \tilde{\delta_X}, \tilde{\delta}_Y) = \mathbf{W}\mathbf{h} + \mathbf{b}, \tag{7.5}$$

where $\mathbf{W}$ is the transformation matrix and $\mathbf{b}$ is the bias. This is followed by normalization of the parameters:

$$g_X = \frac{\tilde{g}_X + 1}{2}, \qquad g_Y = \frac{\tilde{g}_Y + 1}{2}, \tag{7.6}$$

$$\delta_X = \frac{A-1}{M-1} \cdot |\tilde{\delta_X}|, \qquad \delta_Y = \frac{B-1}{N-1} \cdot |\tilde{\delta_Y}|, \tag{7.7}$$

$$\sigma_X = |\tilde{\sigma_X}|, \qquad \sigma_Y = |\tilde{\sigma_Y}|. \tag{7.8}$$

The mean coordinates μ_X^i, μ_Y^j of the Gaussian filter at column i, row j in the attention grid are computed as follows:

$$\mu_X^i = g_X + (i - \frac{M}{2} - 0.5) \cdot \delta_X, \tag{7.9}$$

$$\mu_Y^j = g_Y + (j - \frac{N}{2} - 0.5) \cdot \delta_Y \tag{7.10}$$

Finally, the filter banks $\mathbf{F}_X$ and $\mathbf{F}_Y$ are defined by:

$$\mathbf{F}_X[i, a] = \exp\left(-\frac{(a - \mu_X^i)^2}{2\sigma^2}\right), \tag{7.11}$$

$$\mathbf{F}_Y[j, b] = \exp\left(-\frac{(b - \mu_Y^j)^2}{2\sigma^2}\right) \tag{7.12}$$

The filters (rows of $\mathbf{F}_X$ and $\mathbf{F}_Y$) are later normalized to sum to one.

Our read mechanism makes the following modifications to the DRAW read mechanism (Gregor *et al.*, 2015):

- We allow rectangular (not only square) attention grids and we use separate strides and standard deviations for the X and Y-axis. This allows the model to stretch and smooth the glimpse content to correct for distortions introduced by ignoring the original aspect ratio of an input image.
- We use the absolute value function $abs(x) = |x|$ instead of $\exp(x)$ to ensure positivity of strides and standard deviations (see Equations 7.7 and 7.8). The motivation for this modification is that in our experiments we observed stride and standard deviation parameters to often saturate at low values, causing the attention window to zoom in on a single pixel. This effectively inhibits gradient flow through neighboring pixels of the attention filters. Piecewise linear activation functions have been shown to benefit optimization (Nair and Hinton, 2010) and the absolute value function is a convenient trade-off between the harsh zeroing of all negative inputs of the ReLU and the extreme saturation for highly negative inputs of the exponential function.
- We drop the additional scalar intensity parameter γ, because we did not observe it to influence the performance in our experiments.

7.4 A Modular Framework for Vision

The proposed modular framework for an attention-based approach to computer vision consists of three components: the attention module (controlling *where* to look), the feature-extraction module (providing a representation of *what* is seen) and the objective module (formalizing *why* the model is learning its attentive behavior). An example architecture for tracking using these modules is described in Section 7.5.

7.4.1 Feature-extraction module

The feature-extraction module computes a feature representation of a given input glimpse. This representation can be as simple as the identity transformation, i.e. the original pixel representation, or a more sophisticated feature extractor, e.g. an CNN. The extracted features are used by other modules to reason about the visual input. Given a hierarchy of features, such as the activations of layers in an CNN, different features can be passed to the attention and objective modules.

We found that it can be useful to pre-train the feature-extraction module on a large data set, before starting to train the full architecture. After pre-training, the feature extractor's parameters can either be continued to be updated during end-to-end training, or kept fixed.

Figure 7.3 shows the symbol used in the following sections to represent a feature-extraction module.

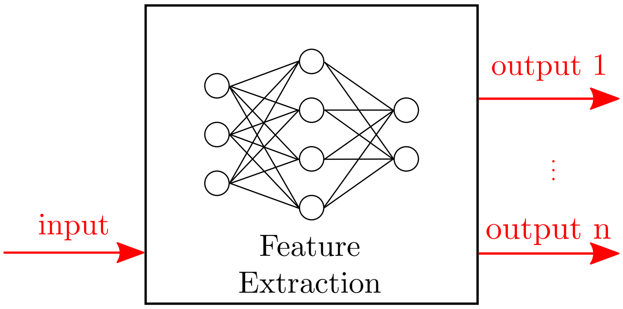


Figure 7.3 The symbol for the feature-extraction module. It represents an arbitrary feature extractor, that can have multiple outputs (e.g. for activations from different layers of an CNN).

7.4.2 Attention Module

The attention module is composed of an RNN (see Section 7.2) and a read mechanism (see Section 7.3). At each time step, a glimpse is extracted from the current input frame using the attention parameters the RNN predicted in the previous time step (see Section 7.3). Note that in this context, Equation 7.5 of the read mechanism corresponds to Equation 7.2 of the RNN. After the glimpse extraction, the RNN updates its hidden state using the feature representation of the glimpse as input (see Equation 7.1). Figure 7.4 shows the symbolic representation used in the following sections to represent the recurrent attention module.

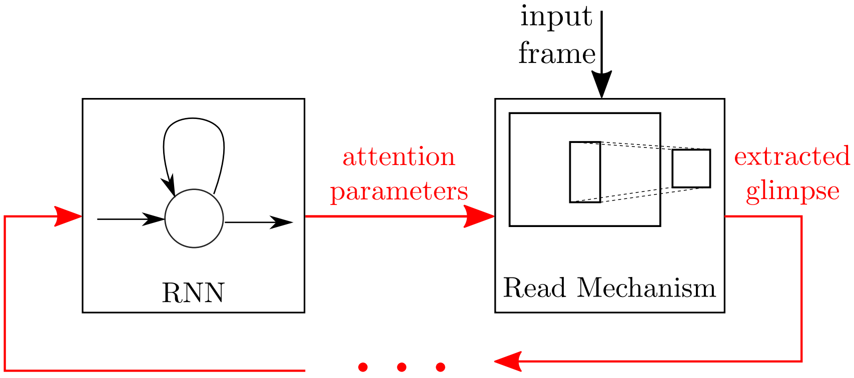


Figure 7.4 The symbolic representation of a recurrent attention module, which is composed of an RNN and a read mechanism that extracts a glimpse from the input frame. The extracted glimpse is fed back to the RNN. The dots indicate, that the feed-back connection can involve intermediate processing steps, such as feature extraction.

7.4.3 Objective Module

An objective module guides the model to learn an attentional policy to solve a given task. It outputs a scalar cost, that is computed as function of its target and prediction inputs. There can be multiple objective modules for a single task. A learning algorithm, such as SGD, uses the sum of cost terms from all objective modules to adapt the parameters of the other modules. Objective modules can receive their input from different parts of the network. For example, if we want to define a penalty between window coordinates, the module would receive predicted attention parameters from the attention module and target parameters from the trainer.

In all our objective modules we use the Mean Squared Error (MSE) for training:

$$\mathcal{L}_{MSE} = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{y}_{target} - \mathbf{y}_{pred}||_2^2, \tag{7.13}$$

where n is the number of training samples, $\mathbf{y}_{pred}$ is the model's prediction, $\mathbf{y}_{target}$ is the target value and $||.||_2^2$ is the squared Euclidean norm. We use the MSE even for classification, as this makes the combination of multiple objectives simpler and worked well. Figure 7.5 shows the symbol used in the following sections to represent an objective module.

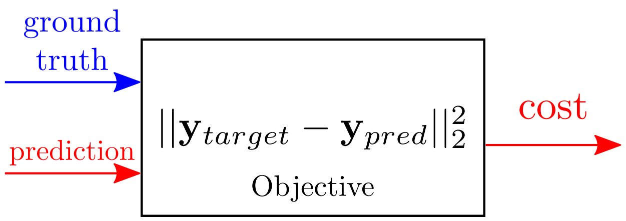


Figure 7.5 The symbol for the objective module. It represents the computation of a scalar cost term given prediction and ground truth inputs.

7.5 Building a Recurrent Attentive Tracking Model

The task of tracking involves mapping a sequence of input images $\mathbf{x}_1, \ldots, \mathbf{x}_T$ to a sequence of object locations $\mathbf{y}_1, \ldots, \mathbf{y}_T$. For the prediction $\hat{\mathbf{y}}_t$ of an object's location at time t, the trajectory $(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$ usually contains highly relevant contextual information, so it is important to choose a hidden state model which has the capacity to represent this trajectory.

7.5.1 Architecture

At each time step, the recurrent attention module outputs a glimpse from the current input frame using the attention parameters predicted at the previous time step. Optionally, a feature-extraction module extracts a representation of the glimpse and feeds it back to the attention module, which updates its hidden state. The tracking behavior can be learned in various ways:

- One can penalize the difference between the glimpse content and a ground truth image. This can be done in the raw pixel space for simple data sets, which do not show much variation in the objects appearance. This loss is defined as

  $$\mathcal{L}_{pixel} = ||\hat{\mathbf{p}} - \mathbf{p}||_2^2, \tag{7.14}$$

  where $\hat{\mathbf{p}}$ is the glimpse extracted by the attention mechanism and $\mathbf{p}$ is the ground truth image. For objects with more variance in appearance, a distance measure between features extracted from the glimpse and from the ground truth image, is more appropriate:

  $$\mathcal{L}_{feat} = ||f(\hat{\mathbf{p}}) - f(\mathbf{p})||_2^2, \tag{7.15}$$

  where $f(.)$ is the function computed by a feature-extraction module.

- Alternatively, a penalty term can also be defined directly on the attention parameters.

For instance, the distance between the center of the ground truth bounding box and the attention mechanism's $\hat{\mathbf{g}} = (g_x, g_y)$ parameters can be used as a localization loss

$$\mathcal{L}_{loc} = ||\hat{\mathbf{g}} - \mathbf{g}||_2^2, \tag{7.16}$$

We explore several variations of this architecture in Section 7.6.

7.5.2 Evaluation of Tracking Performance

Tracking models can be evaluated quantitatively on test data using the average Intersection-over-Union (IoU) (Everingham *et al.*, 2010)

$$IoU = \frac{|B_{gt} \cap B_{pred}|}{|B_{gt} \cup B_{pred}|}, \tag{7.17}$$

where B_{gt} and B_{pred} are the ground truth and predicted bounding boxes. A predicted bounding box for RATM is defined as the rectangle between the corner points of the attention grid. This definition of predicted bounding boxes ignores the fact that each point in the glimpse is a weighted sum of pixels around the grid points and the boxes are smaller than the region seen by the attention module. While this might affect the performance under the average IoU metric, the average IoU still serves as a reasonable metric for the soft attention mechanism's performance in tracking.

7.6 Experimental Results

For an initial study, we use generated data, as described in Sections 7.6.1 and 7.6.2, to explore some design choices without being limited by the number of available training sequences. In Section 7.6.3, we show how one can apply the RATM in a real-world context.

7.6.1 Bouncing Balls

For our initial experiment, we generated videos of a bouncing ball using the script released with Sutskever *et al.* (2009). The videos have 32 frames of resolution 20 × 20. We used $100,000$ videos for training and $10,000$ for testing. The **attention module** has 64 hidden units in its RNN and its read mechanism extracts glimpses of size 5 × 5. The attention parameters are initialized to a random glimpse in the first frame. The input to the RNN are the raw pixels of the glimpse, i.e. the **feature-extraction module** here corresponds to the identity transformation. The **objective module** computes the MSE between the glimpse

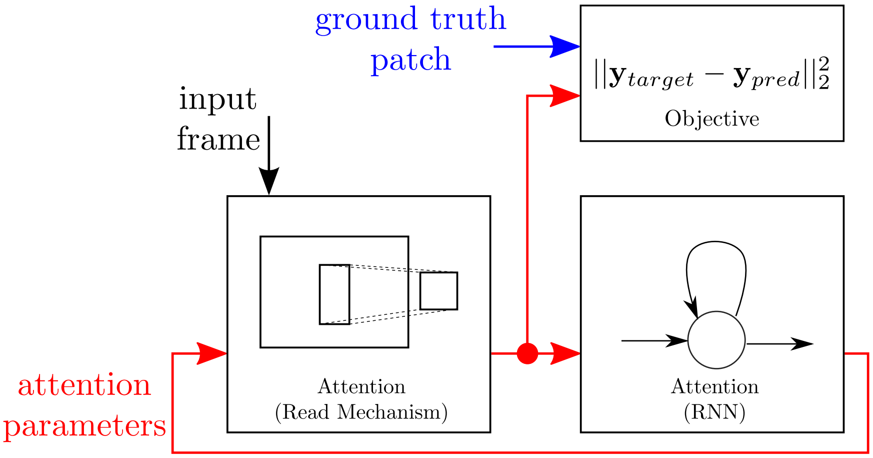

ground truth patch
input frame
$||\mathbf{y}_{target} - \mathbf{y}_{pred}||_2^2$
Objective
attention parameters
Attention (Read Mechanism)
Attention (RNN)

Figure 7.6 The architecture used for bouncing balls experiments.

at the last time step and a target patch, which is simply a cropped white ball image, since shape and color of the object are constant across the whole data set. Figure 7.6 shows a schematic of this architecture.

For **learning**, we use SGD with a mini-batch size of 16, a learning rate of 0.01 and gradient clipping (Pascanu *et al.*, 2012) with a threshold of 1 for 200 epochs. Figure 7.7 shows results of tracking a ball in a test sequence. RATM is able to learn the correct tracking behaviour only using the penalty on the last frame. We also trained a version with the objective module computing the average MSE between glimpses of all time steps and the target patch. An example tracking sequence of this experiment is shown in Figure 7.8. The first two rows of Table 7.1 show the test performance of the model trained with only penalizing the last frame during training. The first row shows the average IoU of the last frame and the second shows the average IoU over all 32 frames of test sequences. The third row shows the average IoU

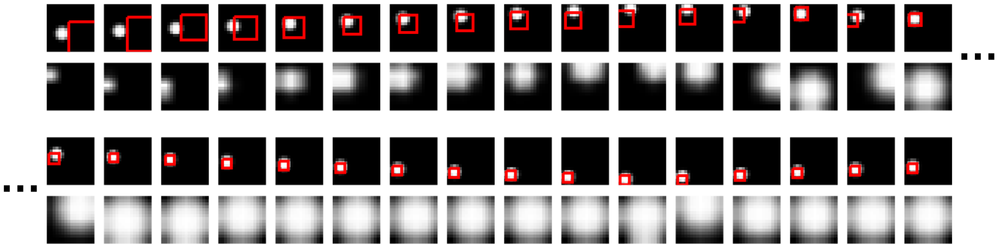


Figure 7.7 An example of tracking on the bouncing ball data set. The first row shows the first 16 frames of the sequence with a red rectangle indicating the location of the glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence.

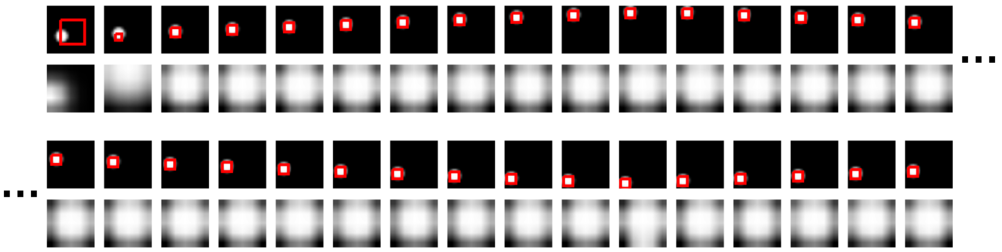


Figure 7.8 Tracking result on a test sequence from a model trained with the MSE penalty at every time step. The first row shows the first 16 frames of the sequence with a red rectangle indicating the location of the glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence.

over all frames of the model trained with the penalty on all frames.

The model trained with the penalty at every time step is able to track a bouncing ball for sequences that are much longer than the training sequences. We generated videos that are almost ten times longer (300 frames) and RATM reliably tracks the ball until the last frame. An example is uploaded as part of the supplementary material.

The dynamics in this data-set are rather limited, but as a proof-of-concept they show that the model is able to learn tracking behavior end-to-end. We describe more challenging tasks in the following sections.

7.6.2 MNIST

To increase the difficulty of the tracking task, we move to more challenging data sets, containing more than a single type of object (ten digits), each with variation. We generate videos from 28×28 MNIST images of handwritten digits (LeCun *et al.*, 1998) by placing randomly-drawn digits in a larger 100×100 canvas with black background and moving the digits from one frame to the next. We respected the same data split for training and testing as in the original MNIST data-set, i.e. digits were drawn from the training split to generate training sequences and from the test split for generation of the test sequences.

Figure 7.9 shows the schematic of RATM for the MNIST experiments. The **attention module** is similar to the one used in Section 7.6.1, except that its RNN has 100 hidden units and the size of the glimpse is 28×28 (the size of the MNIST images and the CNN input layer).

In the bouncing balls experiment we were able to generate a reliable training signal using pixel-based similarity. However, the variation in the MNIST data set requires a represen-

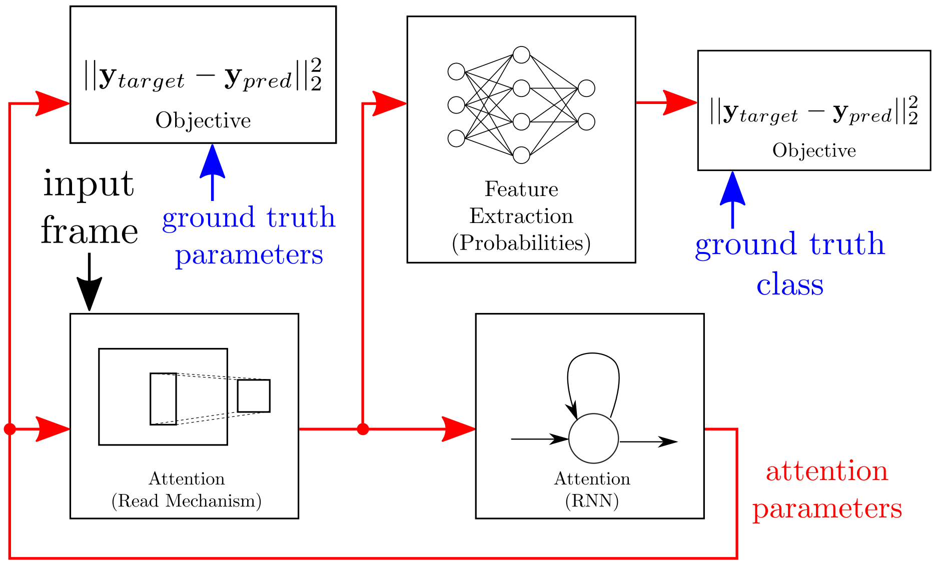

$||\mathbf{y}_{target} - \mathbf{y}_{pred}||_2^2$
Objective
input frame
ground truth parameters
Feature Extraction (Probabilities)
$||\mathbf{y}_{target} - \mathbf{y}_{pred}||_2^2$
Objective
ground truth class
Attention (Read Mechanism)
Attention (RNN)
attention parameters

Figure 7.9 The architecture used for MNIST experiments.

tation that is robust against small variations to guide the training. For this reason, our **feature-extraction module** consists of a (relatively shallow) CNN, that is pre-trained on classification of MNIST digits. Note, that this CNN is only used during training. The CNN structure has two convolutional layers with filter bank sizes of 32 × 5 × 5, each followed by a 2 × 2 maxpooling layer, 0.25 dropout (Hinton *et al.*, 2012b), and the ReLU activation function. These layers are followed by a 10-unit softmax layer for classification. The CNN was trained using SGD with a mini-batch size of 128, a learning rate 0.01, momentum of 0.9 and gradient clipping with a threshold of 5.0 to reach a validation accuracy of 99%.

This CNN classifier is used to extract class probabilities for each glimpse and its parameters remain fixed after pre-training. One of the **objective modules** computes the loss using these probabilities and the target class. Since training did not converge to a useful solution using only this loss, we first introduced an additional objective module penalizing the distances between the upper-left and lower-right bounding-box corners and the corresponding target coordinates. While this also led to unsatisfactory results, we found that replacing the bounding box objective module with one that penalized only grid center coordinates worked well. One possible explanation is, that the grid center penalty does not constrain the stride. Therefore, the glimpse is free to explore without being forced to zoom in. The two penalties on misclassification and on grid center distance, helped the model to reliably find and track the digit. The localization term helped in the early stages of training to guide RATM to track the digits, whereas the classification term encourages the model to properly zoom into the image to maximize classification accuracy.

For **learning** we use SGD with mini-batch size of 32, a learning rate of 0.001, momentum of 0.9 and gradient clipping with a threshold of 1 for $32,000$ gradient descent steps.

Single-Digit

In the first MNIST experiment, we generate videos, each with a single digit moving in a random walk with momentum. The data set consists of $100,000$ training sequences and $10,000$ test sequences. The initial glimpse roughly covers the whole frame.

Training is done on sequences with only 10 frames. The classification and localization penalties were applied at every time-step. At test time, the CNN is switched off and we let the model track test sequences of 30 frames. The fourth row of Table 7.1 shows the average IoU over all frames of the test sequences. Figure 7.10 shows one test sample.

Multi-Digit

It it interesting to investigate how robust RATM is in presence of another moving digit in the background. To this end, we generated new sequences by modifying the bouncing balls script released with Sutskever *et al.* (2009). The balls were replaced by randomly drawn MNIST digits. We also added a random walk with momentum to the motion vectors. We generated $100,000$ sequences for training and $5,000$ for testing.

Here, the bias for attention parameters is not a learn-able parameter. For each video, the bias is set such that the initial glimpse is centered on the digit to be tracked. Width and height are set to about 80% of the frame size. The model was also trained on 10 frame sequences and was able to focus on digits for at least 15 frames on test data. Figure 7.11 shows tracking results on a test sequence. The fifth row of Table 7.1 shows the average IoU

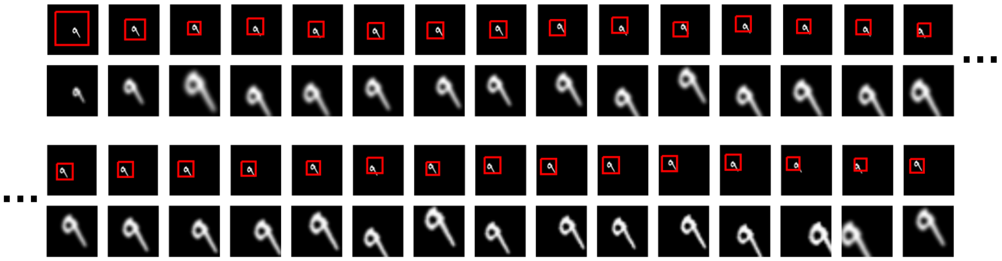


Figure 7.10 Tracking one digit. The first and second row show the sequence and corresponding extracted glimpses, respectively. The red rectangle indicates the location of the glimpse in the frame. The third and fourth row are the continuation. Prediction works well far beyond the training horizon of 10 frames.

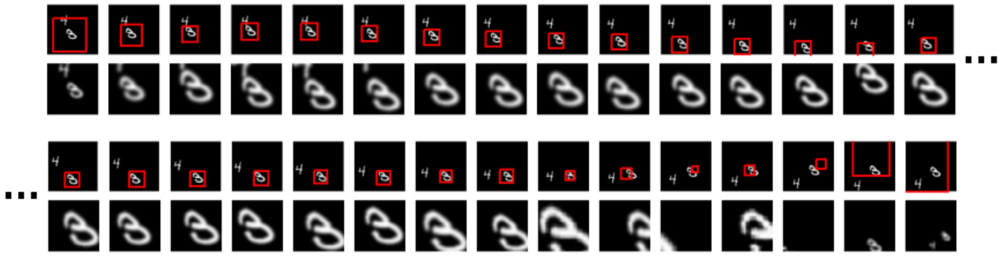


Figure 7.11 Tracking one of two digits. The first and second row show the sequence and corresponding extracted glimpses, respectively. The red rectangle indicates the location of the glimpse in the frame. The third and fourth row are the continuation. Prediction works well for sequences twice as long as the training sequences with 10 frames.

of all test sequences over 30 frames.

7.6.3 Tracking humans in video

To evaluate the performance on a real-world data set, we train RATM to track humans in the KTH action recognition data set (Schüldt *et al.*, 2004), which has a reasonably large number of sequences. We selected the three activity categories, which show considerable motion: walking, running and jogging. We used the bounding boxes provided by Jiang *et al.* (2012), which were not hand-labeled and contain noise, such as bounding boxes around the shadow instead of the subject itself.

For the **feature-extraction module** in this experiment, we trained a CNN on binary – human vs. background – classification of 28×28 grayscale patches. To generate training data for this CNN, we cropped positive patches from annotated subjects in the ETH pedestrian (Ess *et al.*, 2008) and INRIA person (Dalal and Triggs, 2005) data sets. Negative patches were sampled from the KITTI detection benchmark (Geiger *et al.*, 2012). This yielded $21,134$ positive and $29,923$ negative patches, of which we used $20,000$ per class for training. The architecture of the CNN is as follows: two convolutional layers with filter bank sizes $128 \times 5 \times 5$ and $64 \times 3 \times 3$, each followed by 2×2 max-pooling and a ReLU activation. After the convolutional layers, we added one fully-connected ReLU-layer with 256 hiddens and the output softmax-layer of size 2. For **pre-training**, we used SGD with a mini-batch size of 64, a learning rate of 0.01, momentum of 0.9 and gradient clipping with a threshold of 1. We performed early stopping with a held-out validation set sampled randomly from the combined data set.

As this real-world data set has more variation than the previous data sets, the **attention module**'s RNN can also benefit from a richer feature representation. Therefore, the ReLU activations of the second convolutional layer of the feature-extraction module are used as input to the attention module. The RNN has 32 hidden units. This low number of hidden units was selected to avoid overfitting, as the number of sequences ($1,200$ short sequences) in this data set is much lower than in the synthetic data sets. We initialize the attention parameters for the first time step with the first frame's target window. The initial and target bounding boxes are scaled up by a factor of 1.5 and the predicted bounding boxes are scaled back down with factor $\frac{1}{1.5}$ for testing. This was necessary, because the training data for the feature-extraction module had significantly larger bounding box annotations.

The inputs to the **objective module** are the ReLU activations of the fully-connected layer, extracted from the predicted window and from the target window. The computed cost is the MSE between the two feature vectors. We also tried using the cosine distance between two feature vectors, but did not observe any improvement in performance. The target window is extracted using the same read mechanism as in the attention module. Simply cropping the target bounding boxes would have yielded local image statistics that are too different from windows extracted using the read mechanism. Figure 7.12 shows the schematic of the architecture used in this experiment.

For **learning**, we used SGD with a mini-batch size of 16, a learning rate of 0.001 and gradient clipping with a threshold of 1.0. In this experiment we also added a weight-decay regularization term to the cost function that penalizes the sum of the squared Frobenius norms of the RNN weight matrices from the input to the hidden layer and from the hidden layer to the attention parameters. The squared Frobenius norm is defined as

$$||\mathbf{A}||_F^2 = \sum_i^m \sum_j^n |a_{ij}|^2, \tag{7.18}$$

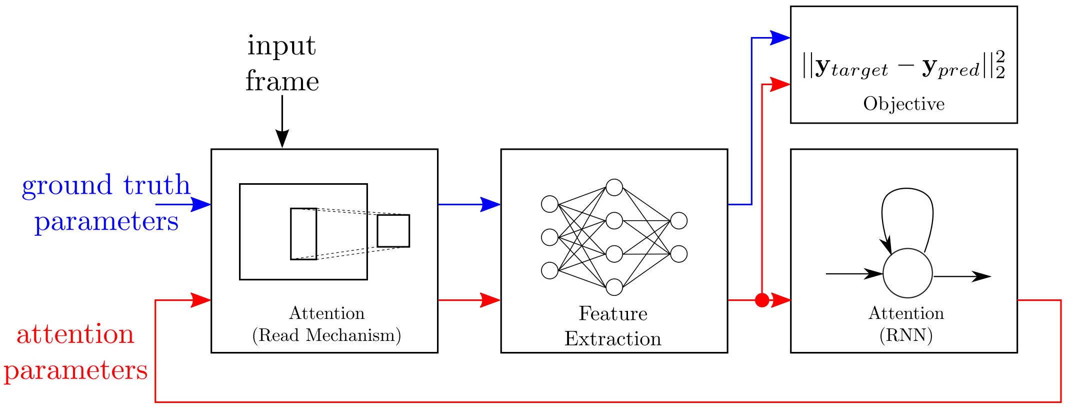


Figure 7.12 The architecture used for KTH experiments.

where a_{ij} is the element at row i, column j in matrix $\mathbf{A}$. This regularization term improved the stability during learning. As another stabilization measure, we started training with short five-frame sequences and increased the length of sequences by one frame every 160 gradient descent steps.

For evaluation, we performed a leave-one-subject-out experiment. For each of the 25 subjects in KTH, we used the remaining 24 for training and validation. A validation subject was selected randomly and used for early stopping. The reported number in the sixth row of Table 7.1 is the IoU on full-length videos of the test subject averaged over all frames of each left-out subject and then averaged over all subjects.

Figure 7.13 shows examples of test sequences for the classes *jogging* and *walking*. Note, that the region captured by the glimpses is larger than the bounding boxes, because the model internally scales the width and height by factor 1.5 and the Gaussian sampling kernels of the attention mechanism extend beyond the bounding box. An interesting observation is that RATM scales up the noisy initial bounding box in Figure 7.13 (bottom example), which covers only a small part of the subject. This likely results from pre-training the feature-extraction module on full images of persons. We observed a similar behavior for multiple other samples. Although the evaluation assumes that the target bounding boxes provided by Jiang *et al.* (2012) are accurate, RATM is able to recover from such noise.

To show how the model generalizes to unseen videos containing humans, we let it predict some sequences of the TB-100 tracking benchmark (Wu *et al.*, 2015a). For this experiment, we picked one of the 25 KTH model, that had a reasonably stable learning curve (IoU over epochs). As an example, Figure 7.14 shows every seventh predicted frame of the *Dancer* sequence and every tenth predicted frame of the sequences *Skater2*, *BlurBody* and *Human2*. For the first two examples, *Dancer* and *Skater2*, RATM tracks the subjects reliably through the whole length of the sequence. This is interesting, as the tracking model was only trained on sequences of up to 30 frames length and the variation in this data is quite different from KTH. The BlurBody and Human2 sequences are more challenging, including extreme camera motion and/or occlusions, causing the model to fail on parts of the sequence. Interestingly in some cases it seems to recover.

In general, the model shows the tendency to grow the window, when it loses a subject. This might be explained by instability of the RNN dynamics and blurry glimpses due to flat Gaussians in the attention mechanism. These challenges will be discussed further in Section 7.7.

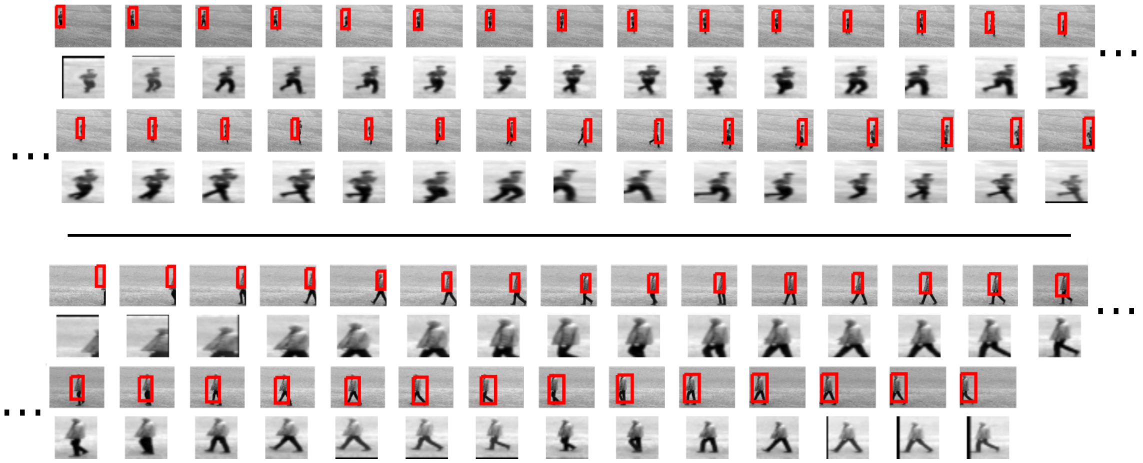


Figure 7.13 Two examples of tracking on the KTH data set. The layout for each example is as follows: the first row shows 15 frames of one test sequence with a red rectangle indicating the location of the glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence. We only show every second frame.

Table 7.1 Average Intersection-over-Union scores on test data.

Experiment	**Average IoU (over # frames)**
Bouncing Balls (training penalty only on last frame)	69.15 (1, only last frame)
Bouncing Balls (training penalty only on last frame)	54.65 (32)
Bouncing Balls (training penalty on all frames)	66.86 (32)
MNIST (single-digit)	63.53 (30)
MNIST (multi-digit)	51.62 (30)
KTH (average leave-one-subject-out)	55.03 (full length of test sequences)

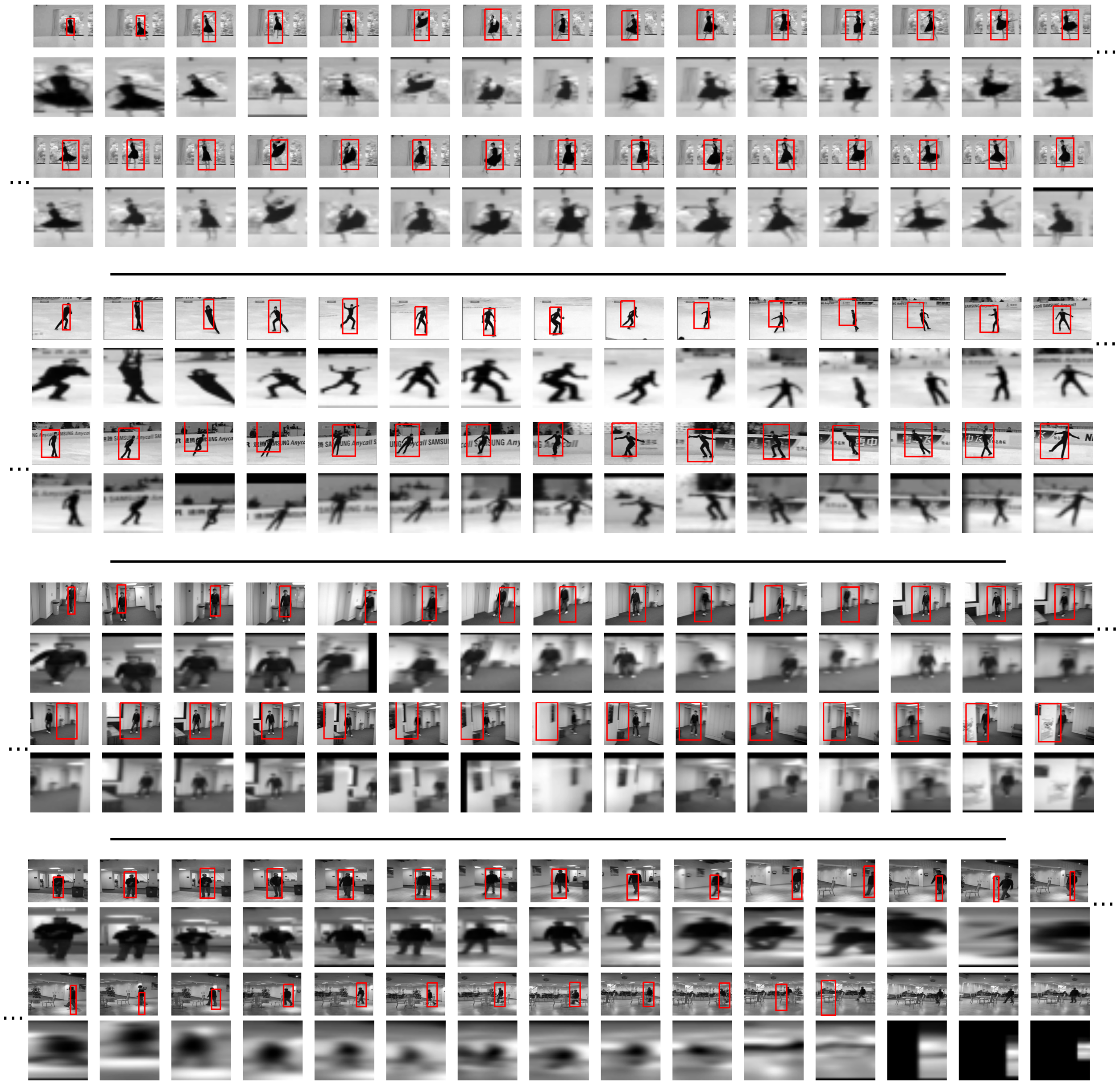


Figure 7.14 Predictions of a KTH model on sequences from the TB-100 benchmark. From top to bottom we show the sequences *Dancer*, *Skater2*, *BlurBody* and *Human2*. To save space, we only show every seventh frame of the Dancer predictions and every tenth frame of the other sequences. The layout for each sequence is as follows: The first row shows 15 frames of one test sequence with a red rectangle indicating the location of the predicted glimpse. The second row contains the extracted glimpses. The third and fourth row show the continuation of the sequence.

7.7 Discussion

We propose a novel neural framework including a soft attention mechanism for vision, and demonstrate its application to several tracking tasks. Contrary to most existing similar approaches, RATM only processes a small window of each frame. The selection of this window is controlled by a learned attentional behavior. Our experiments explore several design decisions that help overcome challenges associated with adapting the model to new data sets. Several observation in the real-world scenario in Section 7.6.3, are important for applications of attention mechanisms in computer vision in general:

- The model can be trained on noisy bounding box annotation of videos and at test time recover from noisy initialization. This might be related to the pre-training of the feature-extraction module on static images. The information about the appearance of humans is transferred to the attention module, which learns to adapt the horizontal and vertical strides among other parameters of the glimpse to match this appearance.
- The trained human tracker seems to generalize to related but more challenging data.

7.8 Directions for Future Research

The modular neural architecture is fully differentiable, allowing end-to-end training. End-to-end training allows the discovery of spatio-temporal patterns, which would be hard to learn with separate training of feature extraction and attention modules. In future work we plan to selectively combine multiple data sets from different tasks, e.g. activity recognition, tracking and detection. This makes it possible to benefit from synergies between tasks (Caruana, 1997), and can help overcome data set limitations.

One could also try to find alternatives for the chosen modules, e.g. replacing the read mechanism with *spatial transformers* (Jaderberg *et al.*, 2015). Spatial transformers offer a more general read mechanism, that can learn to align glimpses using various types of transformations. The application of Spatial Transformers in RNNs for digit recognition has been explored in Sønderby *et al.* (2015a).

Acknowledgments

The authors would like to thank the developers of Theano (Team *et al.*, 2016). We thank Kishore Konda, Jörg Bornschein and Pierre-Luc St-Charles for helpful discussions. This work was supported by an NSERC Discovery Award, CIFAR, FQRNT and the German BMBF, project 01GQ0841.

Reference

SE Kahou, V Michalski, R Memisevic, C Pal, “RATM: Recurrent Attentive Tracking Model”, Submitted to Transactions on Pattern Analysis and Machine Intelligence (TPAMI), (Kahou *et al.*, 2015b)

CHAPTER 8 GENERAL DISCUSSION

The goal of this research study was to contribute to the field of automatic emotion recognition and facial analysis by exploring ways of building an integrated pipeline. Specifically, the explored methods aimed to reduce the amount of parameter hand-tuning in favour of data-driven learning approaches. Besides requiring less manual work, deep learning algorithms in particular have been shown to adapt well in real-world scenarios (Krizhevsky *et al.*, 2012).

Each of the works included in this thesis focuses on a different part of this challenge. While Chapter 4 is an initial study, the articles presented in Chapters 5, 6 and 7, each make a significant step towards developing neural architectures for computer vision. The first and second article focus on developing and refining a neural network approach for emotion recognition in video. The third article describes how an integrated modular framework that is trainable by gradient-based optimization can be used to solve a vision task. Given enough annotated data, this framework can implement a complete pipeline for emotion recognition as will be discussed in Chapter 9.

The contributions of each presented approach and comparisons with existing work are highlighted below in separate paragraphs.

The initial study proposes an important modification of the standard approaches based on LBP features. LBPs have been shown to be strong features for face-related tasks in comparison to other descriptors (Shan *et al.*, 2009). The usual approach is to build histograms of LBPs in local non-overlapping regions which are then concatenated to represent the whole face. This work, however, skips the histogramming and extracts LBPs from overlapping regions centered on facial landmarks at different scales. One intuition for this modification is that histogramming loses information about the exact location of patterns which might contain useful hints for detecting micro-expressions. Replacing this step with PCA for noise reduction and LFDA for discriminative dimensionality reduction, i.e. guided by the emotion recognition task, is shown to yield superior performance on two standard benchmark data sets.

The main contribution of the first article is the development of deep-learning methods for a challenging close-to-real-world data set. The task on this data set is to predict emotion labels for short video clips cropped from Hollywood movies. This task is particularly difficult as achieving a good performance requires the ability to make predictions based on multiple modalities. Moreover, the amount of annotated data is low for training deep learning methods without over-fitting problems. The strongest model in this paper leverages large static image emotion recognition data sets which are combined using a noisy alignment of faces. This combined data set is used to train a deep CNN and predict per-frame probabilities for the challenge data set. The per-frame probabilities were aggregated and used as a feature input to an SVM. This model was combined with weaker models from other modalities using a modified random-search strategy to boost overall performance. To the best of our knowledge, this was the first approach for emotion recognition in video that widely benefits from a mix of deep learning methods. The proposed data set pooling and the specific way of transferring per-frame deep features to the temporal domain has since been adopted by many recent publications in the field.

Recurrent Neural Networks (RNNs) are powerful models for sequential data processing. The second article employs RNNs to address one of the drawbacks in the first article, i.e. the aggregation method used to build a video representation from per-frame features, which ignores some temporal information. A comparison showed that the RNN-based approach significantly outperforms the previous aggregation method and the naive averaging of per-frame probabilities. The application of RNNs on top of CNN features seems to be a recent trend in emotion recognition (Khorrami *et al.*, 2016; Chao *et al.*, 2015). As discussed in the paper, the explicit temporal modeling might better capture motion in video data. In general, it seems important to incorporate a motion representation in feature descriptors used for emotion recognition (Hammal and Cohn, 2014).

The third article presents a neural framework for computer vision that is fully differentiable and can be trained using backpropagation. This contribution is more technical and focuses on the tracking task that is relevant for emotion recognition in video, especially when focusing on facial features. The framework is modular and consists of feature extraction, attention and objective components that can be connected in different ways as discussed in the paper. To the best of our knowledge, the presented architecture using the recurrent soft attention module is the first approach to object tracking with differentiable attention. Such a fully differentiable framework is interesting as end-to-end training might discover useful features for the task which are hard to hand-engineer. The application of various attention mechanisms in different tasks is becoming increasingly popular (Gregor *et al.*, 2015; Mnih *et al.*, 2014; Jaderberg *et al.*, 2015). The presented experiments don't demonstrate the end-to-end training, as feature-extraction modules are pre-trained on static images and then remain fixed. This is partially due to a lack of large data sets. In future work, this can be amended by pooling over many data sets. Parallel training of an architecture on multiple tasks, such as face tracking and emotion recognition is feasible within this framework.

CHAPTER 9 CONCLUSION AND FUTURE WORK

Each of the included works provides an in-depth analysis of corresponding approaches which are mostly based on deep learning techniques such as CNNs, RNNs and attention mechanisms. These works have shown that deep learning can be applied successfully to the task of emotion recognition in more realistic scenarios. Therefore, following new developments and adapting them to the task of emotion recognition is a worthwhile direction of research. Recent advances in neural network modelling techniques and related training approaches have already closed the gap between human and computer performance in several tasks such as object recognition (Szegedy *et al.*, 2014). In the field of emotion recognition, a comparison between computer and human performance is difficult as even human experts can disagree on the annotations of facial expressions, especially of micro-expressions (Corneanu *et al.*, 2016). The high level of disagreement poses a problem for data collection and model evaluation. A low number of annotators can cause a strong bias in a dataset which usually hurts generalization performance, especially in models that learn from data.

As mentioned before, the long-term goal is to develop a complete system for facial expression recognition that can be trained end-to-end. The work in this thesis contributes towards that goal, showing that deep learning techniques can be applied to automate different steps of typical facial expression recognition pipelines. Such pipelines usually consist of detection and registration of faces followed by feature extraction and expression recognition (Corneanu *et al.*, 2016). Two of the articles included in this thesis show how the last two steps of such pipelines can be approached with neural networks. The article in Chapter 5, among other deep learning models, employs a CNN to learn rich visual features from a large set of images and uses them to build a representation for emotion recognition in dynamic videos. The next article in Chapter 6 improves upon this approach by using an RNN in the data-driven construction of video representation from per-frame features. This model significantly outperforms the previous approach which required an intermediate pooling step.

A logical follow-up project would be to automate the pre-processing steps of emotion recognition from faces in video, i.e. the detection and tracking of faces, using the neural framework proposed in the last article in Chapter 7. The first step in such a project would be to collect a large data set of videos containing faces annotated with bounding boxes. Then one can train RATM to track and extract features from facial region. If the model does not converge with end-to-end training, one can still pre-train the feature-extraction module to leverage cross-domain knowledge via transfer learning from face detection data sets. Moreover, training on several tasks together has been shown to yield improvements in performance compared to single-task learning (Caruana, 1997). For example, one can imagine a model that has one feature representation but can output both arousal-valence values as well as emotion categories. Such a model could learn a good emotion representation from multiple small- to medium-scale datasets with different parametrization.

In another direction for future research, one can evaluate how rich context descriptions can help to recognize emotion in the wild. The emergence of richly annotated image data sets such as MS COCO (Lin *et al.*, 2014) and Visual Genome (Krishna *et al.*, 2016) opens the door to explore interesting tasks in the intersection of natural language processing and computer vision. For instance Johnson *et al.* (2015) recently proposed a model composed of a CNN, an attention mechanism and an RNN for dense image captioning. This task involves generating many natural language phrases, each describing a region inside an image. This rich scene description can simplify high level tasks with limited data.

It is also important to explore joint learning of features from different modalities. There are recent publications addressing the problem of fusing modalities with recurrent networks (Chen and Jin, 2015; He *et al.*, 2015). These works suggest methods to deal with challenges arising from different temporal resolutions of features. However, the feature learning stage is separate from the rest of the model. Joint learning might improve generalization given enough data. Besides better generalization, embedding multiple modalities in the same feature space (Kiros *et al.*, 2014) is interesting from the artificial intelligence perspective. It can enable us to better capture and reason about relations between modalities such as vision and audio.

In general, applying deep learning methods to emotion recognition is still a challenge, as collecting annotated emotion data requires more effort compared to standard vision tasks such as object recognition or detection. Moreover, the statistical significance of tests performed on a low number of samples is limited. Consider the following comparison: in our GENKI experiments in Chapter 4, we performed 4-fold cross validation with 1000 test samples in each iteration for a binary classification task; whereas, in all experiments on the EmotiW datasets, the test set size is below 600 samples for a seven-class classification task. Because of the limited number of allowed submissions in the EmotiW challenge, we could not estimate error bars for classification accuracies and it was also difficult to draw conclusions from evaluation on the small test set for different models. For this reason, investing efforts in devising (semi-) automated systems for harvesting and annotating large-scale data sets using crowd-sourcing services such as Amazon Mechanical Turk might speed-up development in this field (see for example Lin *et al.*, 2014; Deng *et al.*, 2009; Krishna *et al.*, 2016).

However, simply collecting more data might not be the ultimate solution; one also has to think about the proper way of annotating emotions, especially in dynamic video data where single labels do not capture enough information. For example, in detection of psychological distress, the intensity of an emotion can be an important indicator (Stratou *et al.*, 2013). Moreover, with dynamic content, the timing of emotional events and their frequencies are highly relevant in tasks such as measuring user's reactions to multimedia content or searching in large video databases. Most of the datasets used in experiments of this thesis either contain only acted emotions or they are mixed with spontaneous emotions. These kind of datasets cannot be used to train a model that detects false emotions or deception. Some recent work suggests six basic emotions do not cover the range of facial expressions well and that composite facial expression labels e.g. happily surprised provide a more detail description (Du *et al.*, 2014). Other works parametrize emotions as a pair of real-valued numbers, expressing arousal, measuring excitement of a subject, and valence, measuring positivity or negativity of an emotion (Ringeval *et al.*, 2013).

A project that collects large-scale data for emotion recognition would have to consider most of the issues mentioned above. As an example, given a larger database of videos containing human interactions, one could collect a rich annotation using a questionnaire asking details about pose, gesture, facial expressions or whether emotions are genuine or acted. Such a dataset would be very useful for multi-task learning projects.

REFERENCES

Adelson, Edward H and Bergen, James R (1985). Spatiotemporal energy models for the perception of motion. *JOSA A*, *2*(2), 284–299.

Ahonen, T. and Hadid, A. and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *28*(12), 2037–2041.

Attneave, Fred (1954). Some informational aspects of visual perception. *Psychological review*, *61*(3), 183.

Baccouche, Moez and Mamalet, Franck and Wolf, Christian and Garcia, Christophe and Baskurt, Atilla (2011). Sequential deep learning for human action recognition. A. Salah and B. Lepri, éditeurs, *Human Behavior Understanding*, Springer Berlin Heidelberg, vol. 7065 de *Lecture Notes in Computer Science*. 29–39.

Bahdanau, Dzmitry and Cho, Kyunghyun and Bengio, Yoshua (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bartlett, Marian Stewart and Hager, Joseph C and Ekman, Paul and Sejnowski, Terrence J (1999). Measuring facial expressions by computer image analysis. *Psychophysiology*, *36*(02), 253–263.

Bartlett, Marian Stewart and Viola, Paul A and Sejnowski, Terrence J and Golomb, Beatrice A and Larsen, Jan and Hager, Joseph C and Ekman, Paul (1996). Classifying facial action. *Advances in neural information processing systems*, 823–829.

Bastien, Frédéric and Lamblin, Pascal and Pascanu, Razvan and Bergstra, James and Goodfellow, Ian and Bergeron, Arnaud and Bouchard, Nicolas and Warde-Farley, David and Bengio, Yoshua (2012). Theano: new features and speed improvements. *arXiv preprint arXiv:1211.5590*.

Bengio, Yoshua and Boulanger-Lewandowski, Nicolas and Pascanu, Razvan (2013). Advances in optimizing recurrent networks. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8624–8628.

Bengio, Yoshua and Goodfellow, Ian J and Courville, Aaron (2015). Deep learning. *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook*.

Bergstra, James and Bengio, Yoshua (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281–305.

Bergstra, James and Breuleux, Olivier and Bastien, Frédéric and Lamblin, Pascal and Pascanu, Razvan and Desjardins, Guillaume and Turian, Joseph and Warde-Farley, David and Bengio, Yoshua (2010). Theano: a cpu and gpu math expression compiler. *Proceedings of the Python for scientific computing conference (SciPy)*. Austin, TX, vol. 4, 3.

Bishop, Christopher M (2006). *Pattern recognition and machine learning*, vol. 1. springer.

Bryson, Arthur Earl (1975). *Applied optimal control: optimization, estimation and control*. CRC Press.

Pierre-Luc Carrier and Aaron Courville and Ian J. Goodfellow and Medhi Mirza and Yoshua Bengio (2013). FER-2013 Face Database. Technical report, 1365, Université de Montréal.

Caruana, Rich (1997). Multitask learning. *Machine learning*, *28*(1), 41–75.

Chang, Chih-Chung and Lin, Chih-Jen (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, *2*, 27:1–27:27.

Chao, Linlin and Tao, Jianhua and Yang, Minghao and Li, Ya and Wen, Zhengqi (2015). Long short term memory recurrent neural network based multimodal dimensional emotion recognition. *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*. ACM, 65–72.

Chen, Dong and Cao, Xudong and Wen, Fang and Sun, Jian (2013). Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '13, 3025–3032.

Chen, JunKai and Chen, Zenghai and Chi, Zheru and Fu, Hong (2014). Emotion recognition in the wild with feature fusion and multiple kernel learning. *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, 508–513.

Chen, Shizhe and Jin, Qin (2015). Multi-modal dimensional emotion recognition using recurrent neural networks. *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*. ACM, 49–56.

Chetlur, Sharan and Woolley, Cliff and Vandermersch, Philippe and Cohen, Jonathan and Tran, John and Catanzaro, Bryan and Shelhamer, Evan (2014). cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*.

Cho, Kyunghyun and Van Merriënboer, Bart and Gulcehre, Caglar and Bahdanau, Dzmitry and Bougares, Fethi and Schwenk, Holger and Bengio, Yoshua (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chopra, Sumit and Hadsell, Raia and LeCun, Yann (2005). Learning a similarity metric discriminatively, with application to face verification. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, vol. 1, 539–546.

Adam Coates and Honglak Lee and Andrew Y. Ng (2011). An Analysis of Single-Layer Networks in Unsupervised Feature Learning. *AISTATS*. 215–223.

Cohn, JF and Zlochower, AJ and Lien, JJ and Wu, YT and Kanade, T (1997). Automated face coding: A computer-vision based method of facial expression analysis. *Psychophysiology*.

Corneanu, Ciprian A and Oliu, Marc and Cohn, Jeffrey F and Escalera, Sergio (2016). Survey on rgb, 3d, thermal, and multimodal approaches for facial expression recognition: History, trends, and affect-related applications.

Cortes, Corinna and Vapnik, Vladimir (1995). Support-vector networks. *Machine learning*, *20*(3), 273–297.

Dahl, George E and Sainath, Tara N and Hinton, Geoffrey E (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. *Proc. ICASSP*. 8609–8613.

Dahmane, M. and Meunier, J. (2011). Emotion recognition using dynamic grid-based HoG features. *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*. 884–888.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. vol. 1, 886–893 vol. 1.

Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li (2009). Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.

Misha Denil and Loris Bazzani and Hugo Larochelle and Nando de Freitas (2011). Learning where to attend with deep architectures for image tracking. *CoRR*, *abs/1109.3737*.

Dhall, Abhinav and Goecke, Roland and Joshi, Jyoti and Sikka, Karan and Gedeon, Tom (2014). Emotion recognition in the wild challenge 2014: Baseline, data and protocol. *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI '14, 461–466.

Abhinav Dhall and Roland Goecke and Jyoti Joshi and Michael Wagner and Tom Gedeon (2013). Emotion recognition in the wild challenge 2013. *ACM ICMI*. 509–516.

Abhinav Dhall and Roland Goecke and Simon Lucey and Tom Gedeon (2012). Collecting large, richly annotated facial-expression databases from movies. *IEEE Multimedia*.

Dhall, Abhinav and Murthy, O. V. Ramana and Goecke, Roland and Joshi, Jyoti and Gedeon, Tom (2015). Video and image based emotion recognition challenges in the wild: Emotiw 2015. *Proceedings of the 17th ACM on International Conference on Multimodal Interaction*. ACM, ICMI '15, 423–426.

Piotr Dollár (2012). Piotr's Image and Video Matlab Toolbox (PMT). `http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html`.

Donahue, Jeffrey and Anne Hendricks, Lisa and Guadarrama, Sergio and Rohrbach, Marcus and Venugopalan, Subhashini and Saenko, Kate and Darrell, Trevor (2015). Long-term recurrent convolutional networks for visual recognition and description. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2625–2634.

Du, Shichuan and Tao, Yong and Martinez, Aleix M (2014). Compound facial expressions of emotion. *Proceedings of the National Academy of Sciences*, *111*(15), E1454–E1462.

Ebrahimi Kahou, Samira and Michalski, Vincent and Konda, Kishore and Memisevic, Roland and Pal, Christopher (2015). Recurrent neural networks for emotion recognition in video. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 467–474.

Peter Eisert and Bernd Girod (1998). Analyzing facial expressions for virtual conferencing. *IEEE Computer Graphics and Applications*, 70–78.

Ekman, Paul (1992). Are there basic emotions? *Psychological Review*, 550–553.

Ekman, Paul and Friesen, Wallace V (1977). Facial action coding system.

Ekman, Paul and Keltner, Dacher (1970). Universal facial expressions of emotion. *California Mental Health Research Digest*, *8*(4), 151–158.

A. Ess and B. Leibe and K. Schindler and and L. van Gool (2008). A mobile vision system for robust multi-person tracking. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*. IEEE Press.

Everingham, Mark and Van Gool, Luc and Williams, Christopher KI and Winn, John and Zisserman, Andrew (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, *88*(2), 303–338.

Eyben, Florian and Wöllmer, Martin and Schuller, Björn (2010). Opensmile: the munich versatile and fast open-source audio feature extractor. *Proceedings of the international conference on Multimedia*. ACM, 1459–1462.

Florian Eyben and Martin Wöllmer and Björn Schuller (2009). openear - introducing the munich open-source emotion and affect recognition toolkit. *ACII*. 576–581.

Fogel, Itzhak and Sagi, Dov (1989). Gabor filters as texture discriminator. *Biological cybernetics*, *61*(2), 103–113.

Fragopanagos, N and Taylor, John G (2005). Emotion recognition in human–computer interaction. *Neural Networks*, *18*(4), 389–405.

Freund, Yoav and Schapire, Robert E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. *Proceedings of the Second European Conference on Computational Learning Theory*. Springer-Verlag, London, UK, UK, EuroCOLT '95, 23–37.

Fukushima, Kunihiko (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, *36*(4), 193–202.

Gehrig, Tobias and Ekenel, Hazım Kemal (2013). Why is facial expression analysis in the wild challenging? *Proceedings of the 2013 on Emotion recognition in the wild challenge and workshop*. ACM, 9–16.

Andreas Geiger and Philip Lenz and Raquel Urtasun (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. *Conference on Computer Vision and Pattern Recognition (CVPR)*.

GENKI-4K (2008). *http://mplab.ucsd.edu, The MPLab GENKI Database, GENKI-4K Subset.*

Google (2013). The Google picasa face detector. [accessed 1-Aug-2013].

Gratch, Jonathan and Lucas, Gale M and King, Aisha Aisha and Morency, Louis-Philippe (2014). It's only a computer: the impact of human-agent interaction in clinical interviews. *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 85–92.

Graves, Alex and Mayer, Christoph and Wimmer, Matthias and Schmidhuber, J and Radig, Bernd (2008). Facial expression recognition with recurrent neural networks.

Graves, Alan and Mohamed, Abdel-rahman and Hinton, Geoffrey (2013). Speech recognition with deep recurrent neural networks. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 6645–6649.

Graves, Alex and Schmidhuber, Jürgen (2009). Offline handwriting recognition with multidimensional recurrent neural networks. *Advances in Neural Information Processing Systems*. 545–552.

Karol Gregor and Ivo Danihelka and Alex Graves and Daan Wierstra (2015). DRAW: A recurrent neural network for image generation. *CoRR*, *abs/1502.04623*.

Hamel, Philippe and Lemieux, Simon and Bengio, Yoshua and Eck, Douglas (2011). Temporal pooling and multiscale learning for automatic annotation and ranking of music audio. *ISMIR*. 729–734.

Hammal, Zakia and Cohn, Jeffrey F (2014). Intra-and interpersonal functions of head motion in emotion communication. *Proceedings of the 2014 Workshop on Roadmapping the Future of Multimodal Interaction Research including Business Opportunities and Challenges*. ACM, 19–22.

He, Lang and Jiang, Dongmei and Yang, Le and Pei, Ercheng and Wu, Peng and Sahli, Hichem (2015). Multimodal affective dimension prediction using deep bidirectional long short-term memory recurrent neural networks. *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*. ACM, 73–80.

Heusch, Guillaume and Cardinaux, Fabien and Marcel, Sébastien (2005). Lighting normalization algorithms for face verification. *IDIAP Communication Com05-03*.

Hinton, Geoffrey and Deng, Li and Yu, Dong and Dahl, George E and Mohamed, Abdel-Rahman and Jaitly, Navdeep and Senior, Andrew and Vanhoucke, Vincent and Nguyen, Patrick and Sainath, Tara N and others (2012a). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Sig. Proc. Magazine,*, *29*(6), 82–97.

Hinton, Geoffrey and Osindero, Simon and Teh, Yee-Whye (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527–1554.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hinton, Geoffrey E and Srivastava, Nitish and Krizhevsky, Alex and Sutskever, Ilya and Salakhutdinov, Ruslan (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*.

Hjortsjö, Carl-Herman (1969). *Man's face and mimic language*. Studen litteratur.

Hochreiter, Sepp (1991). Untersuchungen zu dynamischen neuronalen netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, München*.

Hochreiter, S. and Bengio, Y. and Frasconi, P. and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. Kremer and Kolen, éditeurs, *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press. 237–243.

Hochreiter, Sepp and Schmidhuber, Jürgen (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hubel, David H and Wiesel, Torsten N (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, *160*(1), 106–154.

Hyvärinen, Aapo and Hurri, Jarmo and Hoyer, Patrick O (2009). *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.*, vol. 39. Springer Science & Business Media.

Jaderberg, Max and Simonyan, Karen and Zisserman, Andrew and others (2015). Spatial transformer networks. *Advances in Neural Information Processing Systems*. 2008–2016.

Jain, Varun and Crowley, James (2013). Smile Detection Using Multi-scale Gaussian Derivatives. *12th WSEAS International Conference on Signal Processing, Robotics and Automation*.

Jeni, L.A. and Takacs, D. and Lorincz, A. (2011). High quality facial expression recognition in video streams using shape related information only. *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. 2168–2174.

Jiang, Zhuolin and Lin, Zhe and Davis, Larry S (2012). Recognizing human actions by learning and matching shape-motion prototype trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *34*(3), 533–547.

Johnson, Justin and Karpathy, Andrej and Fei-Fei, Li (2015). Densecap: Fully convolutional localization networks for dense captioning. *arXiv preprint arXiv:1511.07571.*

Jones, Judson P and Palmer, Larry A (1987). An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex. *Journal of neurophysiology*, *58*(6), 1233–1258.

Kahou, Samira Ebrahimi and Bouthillier, Xavier and Lamblin, Pascal and Gulcehre, Caglar and Michalski, Vincent and Konda, Kishore and Jean, Sébastien and Froumenty, Pierre and Dauphin, Yann and Boulanger-Lewandowski, Nicolas and Chandias Ferrari, Raul and Mirza, Mehdi and Warde-Farley, David and Courville, Aaron and Vincent, Pascal and Memisevic, Roland and Pal, Christopher and Bengio, Yoshua (2015a). Emonets: Multimodal deep learning approaches for emotion recognition in video. *Journal on Multimodal User Interfaces*, 1–13.

Kahou, Samira Ebrahimi and Froumenty, Pierre and Pal, Christopher (2014). Facial expression analysis based on high dimensional binary features. *ECCV Workshop on Computer Vision with Local Binary Patterns Variants.* Zurich, Switzerland, 135–147.

Kahou, Samira Ebrahimi and Michalski, Vincent and Memisevic, Roland (2015b). RATM: recurrent attentive tracking model. *arXiv preprint arXiv:1510.08660.*

Kahou, Samira Ebrahimi and Pal, Christopher and Bouthillier, Xavier and Froumenty, Pierre and Gulcehre, Caglar and Memisevic, Roland and Vincent, Pascal and Courville, Aaron and Bengio, Yoshua and Ferrari, Raul Chandias and Mirza, Mehdi and Jean, Sébastien and Carrier, Pierre-Luc and Dauphin, Yann and Boulanger-Lewandowski, Nicolas and Aggarwal, Abhishek and Zumer, Jeremie and Lamblin, Pascal and Raymond, Jean-Philippe and Desjardins, Guillaume and Pascanu, Razvan and Warde-Farley, David and Torabi, Atousa and Sharma, Arjun and Bengio, Emmanuel and Côté, Myriam and Konda, Kishore Reddy and Wu, Zhenzhou (2013). Combining modality specific deep neural networks for emotion recognition in video. *Proceedings of the 15th ACM on International Conference on Multimodal Interaction.* ACM, ICMI '13, 543–550.

Kalchbrenner, Nal and Grefenstette, Edward and Blunsom, Phil (2014). A convolutional neural network for modelling sentences. *arXiv:1404.2188.*

Kanade, T. and Cohn, J.F. and YingLi Tian (2000). Comprehensive database for facial expression analysis. *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on.* 46–53.

Khorrami, Pooya and Paine, Thomas and Huang, Thomas (2015). Do deep neural networks learn facial action units when doing expression recognition? *Proceedings of the IEEE International Conference on Computer Vision Workshops.* 19–27.

Khorrami, Pooya and Paine, Tom Le and Brady, Kevin and Dagli, Charlie and Huang, Thomas S (2016). How deep neural networks can improve emotion recognition on video data. *arXiv preprint arXiv:1602.07377.*

Kim, Bo-Kyeong and Lee, Hwaran and Roh, Jihyeon and Lee, Soo-Young (2015). Hierarchical committee of deep cnns with exponentially-weighted decision fusion for static facial expression recognition. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction.* ACM, 427–434.

Kiros, Ryan and Salakhutdinov, Ruslan and Zemel, Richard S (2014). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539.*

Kokiopoulou, Effrosini and Chen, Jie and Saad, Yousef (2011). Trace optimization and eigenproblems in dimension reduction methods. *Numerical Linear Algebra with Applications*, *18*(3), 565–602.

Konda, Kishore and Memisevic, Roland and Krueger, David (2014a). Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337.*

Kishore Reddy Konda and Roland Memisevic and Vincent Michalski (2014b). Learning to encode motion using spatio-temporal synchrony. *Proceedings of ICLR.*

Krishna, Ranjay and Zhu, Yuke and Groth, Oliver and Johnson, Justin and Hata, Kenji and Kravitz, Joshua and Chen, Stephanie and Kalantidis, Yannis and Li, Li-Jia and Shamma, David A and Bernstein, Michael and Fei-Fei, Li (2016). Visual genome: Connecting language and vision using crowdsourced dense image annotations.

Alex Krizhevsky (2009). Learning multiple layers of features from tiny images.

Krizhevsky,Alex (2012). Cuda-convnet Google code home page. `https://code.google.com/p/cuda-convnet/`.

Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 1097–1105.

Larochelle, Hugo and Hinton, Geoffrey E (2010). Learning to combine foveal glimpses with a third-order boltzmann machine. *Advances in neural information processing systems*. 1243–1251.

Le, Q.V. and Zou, W.Y. and Yeung, S.Y. and Ng, A.Y. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *CVPR*. 3361–3368.

Le, Quoc V and Jaitly, Navdeep and Hinton, Geoffrey E (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

LeCun, Yann and Bottou, Léon and Bengio, Yoshua and Haffner, Patrick (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lien, James J and Kanade, Takeo and Cohn, Jeffrey F and Li, Ching-Chung (1998). Automated facial expression recognition based on facs action units. *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*. IEEE, 390–395.

Lien, James J and Kanade, Takeo and Zlochower, Adena J and Cohn, Jeffrey F and Li, Ching-chung (1997). Automatically recognizing facial expressions in spatio-temporal domain using hidden markov models. *Proceedings of the Workshop on Perceptual User Interfaces*. 94–97.

Lin, Tsung-Yi and Maire, Michael and Belongie, Serge and Hays, James and Perona, Pietro and Ramanan, Deva and Dollár, Piotr and Zitnick, C Lawrence (2014). Microsoft coco: Common objects in context. *Computer Vision–ECCV 2014*, Springer. 740–755.

Liu, Mengyi and Li, Shaoxin and Shan, Shiguang and Chen, Xilin (2013a). Enhancing expression recognition in the wild with unlabeled reference data. *Proceedings of the 11th Asian Conference on Computer Vision - Volume Part II*. Springer-Verlag, Berlin, Heidelberg, ACCV'12, 577–588.

Liu, Mengyi and Wang, Ruiping and Huang, Zhiwu and Shan, Shiguang and Chen, Xilin (2013b). Partial least squares regression on grassmannian manifold for emotion recognition. *Proceedings of the 15th ACM on International conference on multimodal interaction*. ACM, 525–530.

Liu, Mengyi and Wang, Ruiping and Li, Shaoxin and Shan, Shiguang and Huang, Zhiwu and Chen, Xilin (2014). Combining multiple kernel methods on riemannian manifold for emotion recognition in the wild. *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, New York, NY, USA, ICMI '14, 494–501.

Lowe, D.G. (1999). Object recognition from local scale-invariant features. *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. vol. 2, 1150–1157 vol.2.

C. Lu and X. Tang (2014). Surpassing human-level face verification performance on LFW with gaussianface. *Technical report, arXiv:1404.3840*.

Lucey, P. and Cohn, J.F. and Kanade, T. and Saragih, J. and Ambadar, Z. and Matthews, I. (2010). The extended cohn-kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. 94–101.

Matthews, Iain and Baker, Simon (2004). Active appearance models revisited. *International Journal of Computer Vision*, *60*(2), 135–164.

McCulloch, Warren S and Pitts, Walter (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115–133.

Mehrabian, Albert (1971). *Silent messages*. Wadsworth.

Michalski, Vincent (2013). Neural networks for motion understanding: Diploma thesis.

Mnih, Volodymyr and Heess, Nicolas and Graves, Alex and others (2014). Recurrent models of visual attention. *Advances in Neural Information Processing Systems*. 2204–2212.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective*. MIT press.

Nagadomi (2014). Github: kaggle-cifar10-torch7. `https://github.com/nagadomi/kaggle-cifar10-torch7/`.

Nair, Vinod and Hinton, Geoffrey E (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 807–814.

Hyeonseob Nam and Bohyung Han (2015). Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, *abs/1510.07945*.

Neverova, Natalia and Wolf, Christian and Taylor, Graham W and Nebout, Florian (2014). Moddrop: adaptive multi-modal gesture recognition. *arXiv:1501.00102*.

Ng, Hong-Wei and Nguyen, Viet Dung and Vonikakis, Vassilios and Winkler, Stefan (2015). Deep learning for emotion recognition on small datasets using transfer learning. *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*. ACM, 443–449.

Nicolaou, Mihalis A and Gunes, Hatice and Pantic, Maja (2011). Continuous prediction of spontaneous affect from multiple cues and modalities in valence-arousal space. *Affective Computing, IEEE Transactions on*, *2*(2), 92–105.

Ojala, Timo and Pietikäinen, Matti and Harwood, David (1996). A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, *29*(1), 51–59.

Pascanu, Razvan and Mikolov, Tomas and Bengio, Yoshua (2012). On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Polyak, Boris T (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, *4*(5), 1–17.

Prince, Simon JD (2012). *Computer vision: models, learning, and inference*. Cambridge University Press.

Ranzato, Marc'Aurelio (2014). On learning where to look. *arXiv preprint arXiv:1405.5488*.

Rifai, Salah and Bengio, Yoshua and Courville, Aaron and Vincent, Pascal and Mirza, Mehdi (2012). Disentangling factors of variation for facial expression recognition. *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*. Springer-Verlag, Berlin, Heidelberg, ECCV'12, 808–822.

Rifai, Salah and Vincent, Pascal and Muller, Xavier and Glorot, Xavier and Bengio, Yoshua (2011). Contractive auto-encoders: Explicit invariance during feature extraction. *Proceedings of the 28th international conference on machine learning (ICML-11)*. 833–840.

Ringeval, Fabien and Sonderegger, Andreas and Sauer, Jens and Lalanne, Denis (2013). Introducing the recola multimodal corpus of remote collaborative and affective interactions. *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, 1–8.

Romero, Adriana and Ballas, Nicolas and Kahou, Samira Ebrahimi and Chassang, Antoine and Gatta, Carlo and Bengio, Yoshua (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Rosenblatt, Frank (1961). Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document.

Rumelhart, David E and Hinton, Geoffrey E and Williams, Ronald J (1988). Learning representations by back-propagating errors. *Cognitive modeling*, *5*(3), 1.

Rush, Alexander M and Chopra, Sumit and Weston, Jason (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Samal, Ashok and Iyengar, Prasana A (1992). Automatic recognition and analysis of human faces and facial expressions: A survey. *Pattern recognition*, *25*(1), 65–77.

Scherer, Stefan and Stratou, Giota and Mahmoud, Mohamed and Boberg, Jill and Gratch, Jonathan and Rizzo, Alessandro and Morency, Louis-Philippe (2013). Automatic behavior descriptors for psychological disorder analysis. *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE, 1–8.

Schölkopf, Bernhard and Smola, Alexander J (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schüldt, Christian and Laptev, Ivan and Caputo, Barbara (2004). Recognizing human actions: a local svm approach. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. IEEE, vol. 3, 32–36.

Schuller, Björn and Valstar, Michel and Eyben, Florian and McKeown, Gary and Cowie, Roddy and Pantic, Maja (2011). Avec 2011–the first international audio/visual emotion challenge. *Affective Computing and Intelligent Interaction*, Springer. 415–424.

Sermanet, Pierre and Frome, Andrea and Real, Esteban (2014). Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*.

Shaker, Noor and Asteriadis, Stylianos and Yannakakis, Georgios N and Karpouzis, Kostas (2011). A game-based corpus for analysing the interplay between game context and player experience. *Affective Computing and Intelligent Interaction*, Springer. 547–556.

Shan, Caifeng (2012). Smile detection by boosting pixel differences. *Trans. Img. Proc.*, *21*(1), 431–436.

Shan, Caifeng and Gong, Shaogang and McOwan, Peter W. (2009). Facial expression recognition based on local binary patterns: A comprehensive study. *Image Vision Comput.*, *27*(6), 803–816.

Shen, Liping and Wang, Minjuan and Shen, Ruimin (2009). Affective e-learning: Using "emotional" data to improve learning in pervasive learning environment. *Journal of Educational Technology & Society*, *12*(2), 176–189.

Sikka, Karan and Dykstra, Karmen and Sathyanarayana, Suchitra and Littlewort, Gwen and Bartlett, Marian (2013). Multiple kernel learning for emotion recognition in the wild. *Proceedings of the 15th ACM on International conference on multimodal interaction.* ACM, 517–524.

Sikka, Karan and Wu, Tingfan and Susskind, Josh and Bartlett, Marian (2012). Exploring bag of words architectures in the facial expression domain. *Proceedings of the European Conference on Computer Vision.* Springer-Verlag, Berlin, Heidelberg, ECCV'12, 250–259.

Karen Simonyan and Andrew Zisserman (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, *abs/1409.1556*.

Smeulders, Arnold WM and Chu, Dung M and Cucchiara, Rita and Calderara, Simone and Dehghan, Afshin and Shah, Mubarak (2014). Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *36*(7), 1442–1468.

Sønderby, Søren Kaae and Sønderby, Casper Kaae and Maaløe, Lars and Winther, Ole (2015a). Recurrent spatial transformer networks. *arXiv preprint arXiv:1509.05329*.

Sønderby, Søren Kaae and Sønderby, Casper Kaae and Nielsen, Henrik and Winther, Ole (2015b). Convolutional lstm networks for subcellular localization of proteins. *Algorithms for computational biology*, Springer. 68–80.

Srivastava, Nitish and Mansimov, Elman and Salakhutdinov, Ruslan (2015). Unsupervised learning of video representations using lstms. *arXiv preprint arXiv:1502.04681*.

Stratou, Giota and Scherer, Stefan and Gratch, Jonathan and Morency, Louis-Philippe (2013). Automatic nonverbal behavior indicators of depression and ptsd: Exploring gender differences. *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on.* IEEE, 147–152.

Štruc, Vitomir and Pavešić, Nikola (2009). Gabor-based kernel partial-least-squares discrimination features for face recognition. *Informatica*, *20*(1), 115–138.

Sugiyama, Masashi (2007). Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *The Journal of Machine Learning Research*, *8*, 1027–1061.

Sun, Bo and Li, Liandong and Zuo, Tian and Chen, Ying and Zhou, Guoyan and Wu, Xuewen (2014). Combining multimodal features with hierarchical classifier fusion for emotion recognition in the wild. *Proceedings of the 16th International Conference on Multimodal Interaction*. ACM, 481–486.

Sun, Yi and Wang, Xiaogang and Tang, Xiaoou (2013). Deep convolutional network cascade for facial point detection. *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Washington, DC, USA, CVPR '13, 3476–3483.

Joshua Susskind and Adam Anderson and Geoffrey Hinton (2010). The toronto face database. Technical report, UTML TR 2010-001, University of Toronto.

Sutskever, Ilya and Hinton, Geoffrey E and Taylor, Graham W (2009). The recurrent temporal restricted boltzmann machine. *Advances in Neural Information Processing Systems*. 1601–1608.

Sutskever, Ilya and Martens, James and Dahl, George and Hinton, Geoffrey (2013). On the importance of initialization and momentum in deep learning. *Proceedings of the 30th international conference on machine learning (ICML-13)*. 1139–1147.

Sutskever, Ilya and Vinyals, Oriol and Le, Quoc VV (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*. 3104–3112.

Sutton, Richard S and Barto, Andrew G (1998). *Reinforcement learning: An introduction*. MIT press Cambridge.

Suwa, Motoi and Sugie, Noboru and Fujimora, Keisuke (1978). A preliminary note on pattern recognition of human emotional expression. *International joint conference on pattern recognition*. vol. 1978, 408–410.

Szegedy, Christian and Liu, Wei and Jia, Yangqing and Sermanet, Pierre and Reed, Scott and Anguelov, Dragomir and Erhan, Dumitru and Vanhoucke, Vincent and Rabinovich, Andrew (2014). Going deeper with convolutions. *arXiv:1409.4842*.

Taigman, Yaniv and Yang, Ming and Ranzato, Marc'Aurelio and Wolf, Lior (2014). Deepface: Closing the gap to human-level performance in face verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1701–1708.

Yichuan Tang (2013). Deep learning using support vector machines. *CoRR*, *abs/1306.0239*.

Taylor, Graham W. and Fergus, Rob and LeCun, Yann and Bregler, Christoph (2010). Convolutional learning of spatio-temporal features. *Proceedings of the 11th European conference on Computer vision: Part VI*. ECCV'10, 140–153.

Team, The Theano Development and Al-Rfou, Rami and Alain, Guillaume and Almahairi, Amjad and Angermueller, Christof and Bahdanau, Dzmitry and Ballas, Nicolas and Bastien, Frédéric and Bayer, Justin and Belikov, Anatoly and others (2016). Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Urban, Gregor and Geras, Krzysztof J and Kahou, Samira Ebrahimi and Aslan, Ozlem and Wang, Shengjie and Caruana, Rich and Mohamed, Abdelrahman and Philipose, Matthai and Richardson, Matt (2016). Do deep convolutional nets really need to be deep (or even convolutional)? *arXiv preprint arXiv:1603.05691*.

Vincent, Pascal and Larochelle, Hugo and Lajoie, Isabelle and Bengio, Yoshua and Manzagol, Pierre-Antoine (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, *11*, 3371–3408.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. vol. 1, I–511–I–518 vol.1.

V. Štruc and N. Pavešić (2011). Photometric normalization techniques for illumination invariance. *Advances in Face Image Analysis: Techniques and Technologies*, 279–300.

Heng Wang and Muhammad Muneeb Ullah and Alexander Kläser and Ivan Laptev and Cordelia Schmid (2009). Evaluation of local spatio-temporal features for action recognition. *BMVC*. 1–11.

Wang, Naiyan and Yeung, Dit-Yan (2013). Learning a deep compact image representation for visual tracking. *Advances in neural information processing systems*. 809–817.

Werbos, Paul (1974). Beyond regression: New tools for prediction and analysis in the behavioral sciences. *Ph.D. thesis, Harvard University*.

Werbos, Paul J (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Jacob Whitehill and Gwen Littlewort and Ian Fasel and Marian Bartlett and Javier Movellan (2009). Toward practical smile detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(11), 2106–2111.

Wöllmer, Martin and Metallinou, Angeliki and Eyben, Florian and Schuller, Björn and Narayanan, Shrikanth S (2010). Context-sensitive multimodal emotion recognition from speech and facial expression using bidirectional lstm modeling. *INTERSPEECH*. 2362–2365.

Wu, Yi and Lim, Jongwoo and Yang, Ming-Hsuan (2015a). Object tracking benchmark. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *37*(9), 1834–1848.

Wu, Zuxuan and Wang, Xi and Jiang, Yu-Gang and Ye, Hao and Xue, Xiangyang (2015b). Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *arXiv preprint arXiv:1504.01561*.

Xu, Kelvin and Ba, Jimmy and Kiros, Ryan and Courville, Aaron and Salakhutdinov, Ruslan and Zemel, Richard and Bengio, Yoshua (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Yosinski, Jason and Clune, Jeff and Bengio, Yoshua and Lipson, Hod (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*. 3320–3328.

Zabih, Ramin and Woodfill, John (1994). Non-parametric local transforms for computing visual correspondence. *Proceedings of the Third European Conference on Computer Vision (Vol. II)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, ECCV '94, 151–158.

Xiangxin Zhu and Ramanan, D. (2012). Face detection, pose estimation, and landmark localization in the wild. *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2879–2886.

APPENDIX A LIST OF PUBLICATIONS

This thesis is based on the following publications:

- **Chapter 4**
  SE Kahou, P Froumenty, C Pal
  "Facial Expression Analysis Based on High Dimensional Binary Features"
  In proceedings of the ECCV 2014 workshop on computer vision with local binary patterns, *best paper award* (Kahou *et al.*, 2014)

- **Chapter 5**
  SE Kahou, X Bouthillier, P Lamblin, C Gulcehre, V Michalski, K Konda, S Jean, P Froumenty, Y Dauphin, N Boulanger-Lewandowski, RC Ferrari, M Mirza, D Warde-Farley, A Courville, P Vincent, R Memisevic, C Pal, Y Bengio
  "EmoNets: Multimodal deep learning approaches for emotion recognition in video"
  In Journal on Multimodal User Interfaces (JMUI) (Kahou *et al.*, 2015a)

- **Chapter 6**
  SE Kahou, V Michalski, K Konda, R Memisevic, C Pal
  "Recurrent Neural Networks for Emotion Recognition in Video"
  In proceedings of the 17th ACM International Conference on Multimodal Interaction (ICMI '15), *challenge second runner-up* (Ebrahimi Kahou *et al.*, 2015)

- **Chapter 7**
  SE Kahou, V Michalski, R Memisevic, C Pal
  "RATM: Recurrent Attentive Tracking Model"
  Submitted to Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (Kahou *et al.*, 2015b)

The article in Chapter 5 is an extended version with slight modifications in the approach and new experiments of the following work:

- **SE Kahou**, C Pal, X Bouthillier, P Froumenty, Ç Gülçehre, [1], R Memisevic, P Vincent, A Courville, Y Bengio
  "Combining modality specific deep neural networks for emotion recognition in video"

[1] Please see the additional authors section for a complete author list.

In proceedings of the 15th ACM International Conference on Multimodal Interaction (ICMI '13), *challenge winner* (Kahou *et al.*, 2013)

Besides emotion recognition, I have also been working on model compression and knowledge distillation for neural networks. This work contributed to the following publications:

- A Romero, N Ballas, **SE Kahou**, A Chassang, C Gatta, Y Bengio
  "FitNets: Hints for Thin Deep Nets"
  In International Conference on Learning Representations (ICLR '15) (Romero *et al.*, 2014)

- G Urban, KJ Geras, **SE Kahou**, O Aslan, S Wang, R Caruana, A Mohamed, M Philipose, M Richardson
  "Do Deep Convolutional Nets Really Need to be Deep (Or Even Convolutional)?"
  In International Conference on Learning Representations Workshop (ICLR '16) (Urban *et al.*, 2016)