Models for Supervised Learning on Sequence Data

Models for Supervised Learning on Sequence Data

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Wednesday 13, June 2018 at 15:00 o'clock

by

Wenjie Pei

Master of Science in Computer Science and Engineering,
Eindhoven University of Technology, Netherlands,
Born in Shanxi, China.

This dissertation has been approved by

promotor: prof. dr. ir. M. J. T. Reinders and
copromotor: dr. D. M. J. Tax

Composition of the doctoral committee:

Rector Magnificus,	
Prof. dr. ir. M. J. T. Reinders,	Delft University of Technology, promotor
Dr. D. M. J. Tax,	Delft University of Technology, copromotor
Independent members:	
Prof. dr. A. Hanjalic,	Delft University of Technology
Prof. dr. E. O. Postma,	Tilburg University
Prof. dr. B. van Ginneken,	Radboud University Medical Center
Prof. dr. C. G. M. Snoek,	University of Amsterdam
Prof. dr. C. M. Jonker,	Delft University of Technology, reserve member
Other members:	
Dr. L. v. d. Maaten,	Facebook AI Research

Part of this thesis is supported by AAL SALIG++.

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 392.

ISBN 978-94-6186-930-2

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

Cover images are generated based on the datasets used in this thesis: Columbia Consumer Video database, MCYT signature dataset, Arabic spoken digit dataset and Stanford Sentiment TreeBank.

In loving memory of my grandparents

To my family

and

To my fiancee Xiangrong Wang

CONTENTS

1 INTRODUCTION

One important characteristic of human intelligence is that it is able to learn from known examples and further generalize the learned knowledge to unseen objects. For instance, a child can distinguish apples and pears by himself after being shown several examples of apples and pears. This is because he/she can summarize (or extract) the discriminative features between apples and pears, and then apply this knowledge to unseen examples.

Inspired by the learning manner of humans, artificial intelligence aims to build machine learning models to enable machines to learn from examples. In machine learning, the process of learning from known examples (termed as training data) to make predictions on unseen examples is called supervised learning. The training data comprises pairs of input data (the representative features for describing examples) and target values. From it, the supervised learning algorithm aims to infer a mapping function (from the input data to the target values) which can be generalized to predict the target values for new input data. For instance, in the task of face recognition, the training data consists of a set of face images (input data) labeled with the associated person names (target values). We can train a supervised learning model to learn the underlying mapping between faces and labeled names, then the trained model can be used to predict the name of a newly observed face. In this example, we aim to classify a data sample (a face image) into one of a finite number of candidate categories (person names) and the target value is the corresponding category label (person name). Such supervised learning problems are typically referred to as classification problems. Another classical type of supervised learning problems is called regression, in which the target value is a continuous value that indicates the desired property of the object. For instance, we aim to estimate the age of a person from face images in the task of human age estimation, then the target value is the age which is a continuous value.

The input data for a supervised learning model is not always a single observation. Instead, it could be a sequence of observations, in which the order matters. Figure 1.1 presents an example of a sequence $\mathbf{x}_{1,\ldots,T}$ and its associated target value y. The sequence consists of T observations in which $\mathbf{x}_t \in \mathbb{R}^D$ denotes the representative features for describing the observation at the t-th position. The observations in the sequence can be

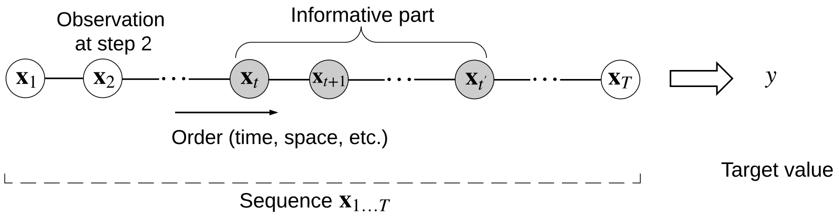


Figure 1.1: The graphical visualization of a sequence $\mathbf{x}_{1,\dots,T}$. It consists of T observations in which $\mathbf{x}_t \in \mathbb{R}^D$ denotes the observation at the t-th position ($\mathbf{x}_t$ can be raw redundant features or extracted features). The informative subsequence which is useful for the task of discrimination is indicated by gray circles. y is its associated target value which is to be predicted using a trained supervised learning model.

arranged either in temporal order (also referred to as time series) or in any other sequential order such as spatial order. In real life, extensive forms of information in a vast variety of human activities can be represented as sequence data. Well-known examples include audio signals in speech, pen trajectories in handwriting or signature, recorded videos as a sequence of images and text documents as a sequence of words.

A potential straightforward way to deal with sequence data is to concatenate all the observations of the sequence into one observation and feed it to traditional supervised learning models. However, it is cumbersome due to several reasons. First of all, the length of sequence samples may be variable instead of a fixed value. For example, signatures from the same person (Figure 1.2) can have varying lengths. The concatenation of all observations in a sequence would result in different lengths of feature vectors for sequences with different lengths, which is hardly handled by traditional supervised learning models. Secondly, the input sequence data may contain irrelevant segments (Figure 1.1) which create potential interference in the learning process. In this scenario, the learner needs to automatically distinguish the informative parts in the sequence. Generally, this is done by adding a feature selection to the learner but most custom approaches implement this as an extra independent step from the learning (prediction) step which introduces redundancies. Consequently, a more optimal approach would be to integrate the step of distinguishing the informative parts in the sequence into the learner. The third reason relates to the specific nature of sequence data, i.e. the observations at different positions are typically not independent. As they generally exhibit some kind of temporally shift-invariant correlations. For example, suppose a signature shown in Figure 1.2 is represented by a sequence of three measurements over time: pen pressure and pen movement in x-direction and y-direction. Obviously, these observations are temporally related, changing this order will inadmissibly result in a signature with different appearance. Hence, temporal dependency is essential to label signatures. While, in principle, these temporal dependencies can be captured by current learners when the observations are presented in an ordered way, they potentially require a lot more training examples than when these temporal dependencies are explicitly modeled in the learner.

This thesis focuses on developing learners that can explicitly deal with sequence data. We will first give a brief introduction to supervised learning on sequence data.

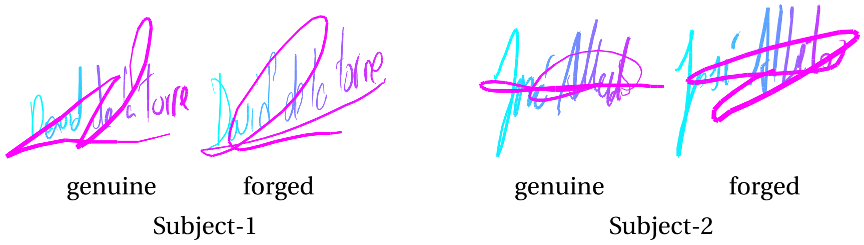


Figure 1.2: Signature examples from MCYT_Signature corpus [1]. We visualize two groups of signatures from two subjects, each group includes one genuine and one forged signature. For each signature, the pen moves from cyan color to magenta color. Wider line indicates higher pressure.

Then, we will elaborate on the research questions investigated in this thesis, finally the contributions of this thesis will be summarized.

1.1. SUPERVISED LEARNING ON SEQUENCE DATA

Given a sequence as input, the goal of supervised learning on sequence data is to predict the target value for this sequence by training a model. Formally, suppose we are given an input sequence $\mathbf{x}_{1,\ldots,T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ of length T in which $\mathbf{x}_t \in \mathbb{R}^D$ denotes the D-dimensional feature vector used to describe the observation at the t-th time step. We are interested in building a model to predict the target value y for the sequence. To this end, a mapping function f is learned from the input sequence to the target value using a training set $\boldsymbol{D} = \{\mathbf{x}^{(n)}_{1,\ldots,T}, y^{(n)}\}_{n=1,\ldots,N}$, which contains N pairs of sequences and their associated target values (the length T of sequences can vary). Typically this is achieved by estimating the model parameters $\boldsymbol{\Theta}$ to maximally fit the mapping function f to the training set $\boldsymbol{D}$. Mathematically, the model is trained by minimizing a specific loss function with respect to the parameters $\boldsymbol{\Theta}$:

$$\min_{\boldsymbol{\Theta}} \sum_{n=1}^{N} L(y^{(n)}, f(\mathbf{x}^{(n)}_{1,\ldots,T})). \tag{1.1}$$

Herein, L is the loss function which quantifies the extent to which the prediction $f(\mathbf{x}^{(n)}_{1,\ldots,T})$ deviates from the true target value $y^{(n)}$ for training sample $\mathbf{x}^{(n)}_{1,\ldots,T}$. For classification tasks, a commonly used loss function is the negative log-likelihood function, which is designed to maximize the predicted probability of the true label. In this case, the mapping function f predicts a probability distribution $\mathbf{P} \in \mathbb{R}^K$ over all K classes instead of predicting a single class given the input $\mathbf{x}^{(n)}_{1,\ldots,T}$:

$$P(\mathbf{y}|\mathbf{x}^{(n)}_{1,\ldots,T}) = f(\mathbf{x}^{(n)}_{1,\ldots,T}), \quad \sum_{i=1}^{K} P(y_i|\mathbf{x}^{(n)}_{1,\ldots,T}) = 1, \tag{1.2}$$

where $P(y_i|\mathbf{x}^{(n)}_{1,\ldots,T})$ denotes the predicted probability (termed as conditional likelihood) for the i-th class. We seek to maximize the conditional likelihood for the true label $P(y^{(n)}|\mathbf{x}^{(n)}_{1,\ldots,T})$, which is equivalent to minimizing the negative value:

$$L(y^{(n)}, f(\mathbf{x}^{(n)}_{1,\ldots,T})) = -\log(P(y^{(n)}|\mathbf{x}^{(n)}_{1,\ldots,T})). \tag{1.3}$$

Here we work with log likelihood for mathematical convenience. This is also known as Maximum Likelihood Estimation (MLE).

Key for supervised learning on sequence data is the modeling of the mapping function $f(\mathbf{x}^{(n)}_{1,...,T})$, which should capture the discriminative information in the sequence as well as exploiting the ordering relationships between the observations in the sequence.

1.1.1. SUPERVISED LEARNING ON SINGLE SEQUENCE

There is substantial amount of prior work on sequence classification, which can be categorized into two types: generative and discriminative modeling. A generative model focuses on modeling the joint probability distribution $P(\mathbf{x}_{1,...,T}, y)$ of input sequence data $\mathbf{x}_{1,...,T}$ and target values y. It can be applied for classification by obtaining the conditional probability $P(y|\mathbf{x}_{1,...,T})$ via Bayes' rule:

$$P(y|\mathbf{x}_{1,...,T}) = \frac{P(\mathbf{x}_{1,...,T}, y)}{P(\mathbf{x}_{1,...,T})} \tag{1.4}$$

A well-known example of a generative model is the hidden Markov model (HMM) [2], which models sequence data using hidden states. HMM can be employed for sequence classification by modeling the data distribution for each class $P(\mathbf{x}_{1,...,T}|y)$ and then the class priors $P(y)$ can be integrated via Bayes' rule to calculate the posterior probability:

$$P(y|\mathbf{x}_{1,...,T}) = \frac{P(\mathbf{x}_{1,...,T}, y)}{P(\mathbf{x}_{1,...,T})} = \frac{P(\mathbf{x}_{1,...,T}|y) \cdot P(y)}{P(\mathbf{x}_{1,...,T})} \tag{1.5}$$

A potential downside of these HMMs-based models is that they first construct features based on generative models (HMMs) and then perform classification, which is a two-stage process. The constructed features may not be well suited for the classification task at hand. In this thesis, we focus on discriminative models, which model the conditional distribution $P(y|\mathbf{x}_{1,...,T})$ and perform classification directly.

A example of a discriminative model for sequence labeling is Conditional Random Fields (CRFs) [3], which model the temporal dependencies by a chain structure (Figure 1.3). While standard CRFs perform well on high-dimensional data, the linear nature of most CRF models limits their modeling capability to learn complex non-linear decision boundaries. To address this limitation, several models extended from CRFs are proposed to deal with non-linear scenarios. A prominent example is the hidden-state conditional random field (HCRF) [4], which introduces a chain of k-nomial hidden variables to model the non-linear latent structure (Figure 1.3). The HCRF has been successfully applied in many tasks such as gesture recognition [5] and action recognition [6]. A potential weakness of the HCRF is that the number of latent states the model can represent (for modeling the decision boundary) grows linearly with the number of hidden variables. This implies that the training of complex models with a large number of hidden variables is very prone to overfitting, while models with smaller numbers of hidden variables may be too simple to represent a good classification function. This downside motivated us to pose our first research question:

- **Research Question 1:** *can we propose a sequence classification model which is able to model complex decision boundaries with a limited number of latent variables?*

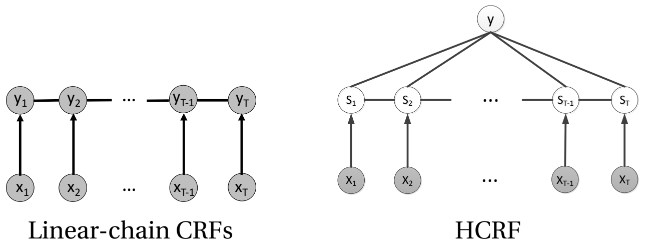


Figure 1.3: The architectures of linear-chain CRFs and HCRF.

Most existing sequence classification models are designed for well-segmented sequences which have been pre-processed to remove irrelevant parts. Therefore, these methods cannot be directly applied to unsegmented sequence data, reducing their applicability. As a result, there is much attention for sequence classification models that are able to ignore the irrelevant parts automatically. A popular approach is the gated recurrent network such as Gated Recurrent Units (GRU) [7] or Long Short-Term Memory (LSTM) [8]. They employ gates (e.g., the input gate, forget gate and output gate in LSTM shown in Figure 1.4) to control the information flow for each time step and filter out the irrelevant information. However, these gates are performed on each of the hidden states and hence cannot interpret the relevance of each observation of the input sequence. This limitation brings us a new research question:

- **Research Question 2:** *can we propose a sequence classification model which is able to deal with unsegmented sequences and meanwhile interpret the relevance of each time step of the input sequence to the classification task?*

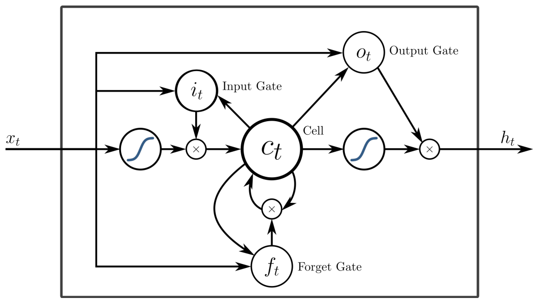


Figure 1.4: Long Short-Term Memory (LSTM) employs three gates to control the information flow for each time step. The figure is adapted from Graves et al. [9].

Typical supervised learning models on sequence data take as input pre-defined features describing each observation of the input sequence (either handcrafted or automatically extracted with some procedure). The benefit is that the sequence model can then focus on modeling the temporal information contained in the sequence. However, the potential drawback of these approaches is that the feature extraction and the supervised learning are performed as a two-state process. Consequently, the extracted features may not be well suited for the supervised task at hand. Alternatively, one can learn features simultaneously with the supervised learning. In such a one-stage approach, features would match the sequence-based model, but the question is whether this is feasible since two tasks need to be performed at the same time. This prompted us for the following research question:

- **Research Question 3:** *can we propose a one-stage supervised sequence model which learns features together with the supervised learning task?*

1.1.2. MODELING SIMILARITY BETWEEN PAIRED SEQUENCES

Apart from the supervised learning on a single sequence, some supervised learning tasks involve paired sequences. Take the example of signature verification shown in Figure 1.2, a typical way to validate the authenticity of a signature is to compare it to a genuine signature in the database and measure their similarity as a confidence score. In this scenario, we aim to build a supervised learning model that takes a pair of sequences as input and measure the similarity between them as output, which is depicted in Figure 1.5. Note that both sequences can have different lengths T_1 and T_2, and the time samples $\mathbf{x}_t^{(1)}$ and $\mathbf{x}_t^{(2)}$ may not correspond to the same time or spatial location. Accordingly, the mapping function f is defined on the paired sequences and corresponding loss function in Equation 1.1 becomes:

$$\min_{\boldsymbol{\Theta}} \sum_{([n,1],[n,2])\in \boldsymbol{D}} L(y^{([n,1],[n,2])}, f(\mathbf{x}_{1,\dots,T_1}^{[n,1]}, \mathbf{x}_{1,\dots,T_2}^{[n,2]})), \tag{1.6}$$

Herein $([n,1],[n,2])$ is a pair of sequences in the training set $\boldsymbol{D}$, and $y^{([n,1],[n,2])}$ is the target value indicating the similarity between two sequences. The challenges in supervised learning on paired sequences lie not only in the modeling of each single sequence, but in modeling the interdependencies between them.

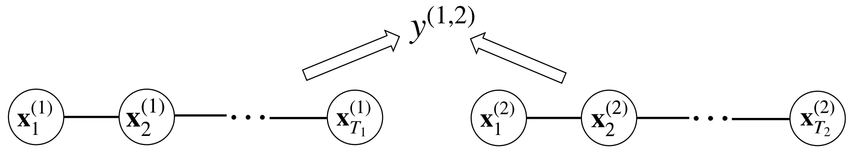


Figure 1.5: The graphical visualization of a pair of sequences $\mathbf{x}_{1,\dots,T_1}^{(1)}$ and $\mathbf{x}_{1,\dots,T_2}^{(2)}$ together with the target value $y^{(1,2)}$ which indicates the association to be investigated between these two sequences. The supervised learning on paired sequences aims to build models to learn a mapping function from the input paired sequences $(\mathbf{x}_{1,\dots,T_1}^{(1)}, \mathbf{x}_{1,\dots,T_2}^{(2)})$ to the target value $y^{(1,2)}$.

Traditional techniques for measuring similarity between sequences are based on handcrafted similarity measures like Dynamic Time Warping (DTW) [11]. DTW is good at aligning two sequences which share a similar shape but may vary in speed (Figure 1.6). Consequently, DTW is good at identifying, for example, signatures from the same person. However, because DTW exploits the signal shape so explicitely, it is difficult for

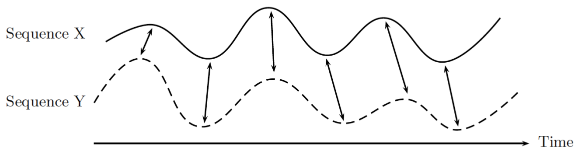


Figure 1.6: Sequence X and sequence Y are aligned by Dynamic Time Warping. The aligned points are indicated by the arrows. The figure is adapted from Müller [10].

DTW to recognize, for example, whether two speech signals with different content are uttered from the same person for voice verification. To address this shortcoming, similarity measures were proposed to employ a generative model to model the data distribution. For instance, Fisher kernels [12] have been widely adopted in Computer Vision [13]. However, these methods model the data distribution in an unsupervised way, which cannot exploit the class information effectively and may result in irrelevant features for the underlying similarity structure. Those findings motivate us to investigate the following research question:

- **Research Question 4:** *can we propose a sequence model that learns a good similarity measure in a supervised way?*

1.1.3. MODELING ASSOCIATION BETWEEN TWO SEQUENCES

This thesis also attempts to model some kind of association between two input sequences. The two sequences contain different types of features (with different dimensionalities) that describe distinct objects (shown in Figure 1.7). Specifically, we seek to model the preference of a user over an item in a recommendation system, In this scenario, the user can be described by a sequence of all the items purchased by this user. Whereas the other input sequence describes an item by recording the representations of all users who have purchased this item. On the one hand, the two sequences should be modeled separately

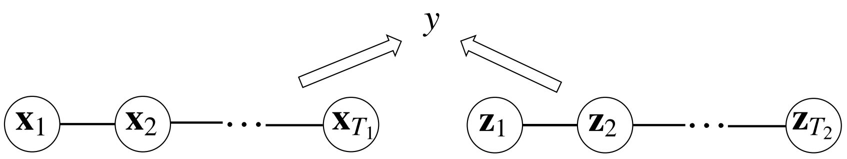


Figure 1.7: Given two sequences $\mathbf{x}_{1,\dots,T_1}$ and $\mathbf{z}_{1,\dots,T_2}$, we aim to model the association y between these two sequences. The two sequences contain different types of features that describe distinct objects.

using individual modules since they own different properties, which is in contrast to the same type of sequences modeled with two same modules in the case of modeling similarity. On the other hand, it is crucial to model the dependencies between them since they affect each other mutually. This specific topic gives rise to an interesting research question that is worth to explore:

- **Research Question 5:** *Given two input sequences representing a pair of historic user and item data, how to model the preference of the user over the item, which takes* *into account not only the information contained in each individual sequence, but also the interdependencies between them?*

1.2. CONTRIBUTIONS

The main contributions of this thesis are that we propose models for supervised learning on sequence data to address the research questions posed in Section 1.1. Each of these questions is specifically approached in an individual chapter in the remainder of this thesis.

Our first contribution is a sequence classification model, called the hidden-unit logistic model (HULM), which uses binary stochastic hidden units to model latent structure in sequence data. The hidden units are connected in a chain structure that models temporal dependencies in the input sequence. Notably, compared with the prior models for sequence classification such as Hidden Conditional Random Fields (HCRFs), our proposed model is able to model much more complex decision boundaries since our model can represent an exponential number of hidden states with the same number of hidden variables as HCRF model. This work answers ***Research Question 1*** and is presented in Chapter 2.

Chapter 3 investigates ***Research Question 2***. Specifically, it introduces a sequence classification model, referred to as Temporal Attention-gated Model (TAGM), which integrates ideas from attention models and gated recurrent networks to better deal with noisy or unsegmented sequences. We extend the concept of the attention model to measure the relevance of each time step of a sequence. We then use a novel gated recurrent network to learn a latent representation for the final classification. An important advantage of this model is that the learned attention weights provide a meaningful interpretation for the salience of each time step in the input sequence.

The third contribution of this thesis is an end-to-end architecture for age estimation from facial expression videos, which attempts to combine the feature learning and sequence supervised learning together as an integrated system, discussed in ***Research Question 3***. Our proposed model is able to simultaneously learn both the appearance features in each frame of the input video as well as the temporal dynamics. More importantly, we propose to leverage attention models for salience detection in both the spatial domain for each single image and the temporal domain for the whole video. We design a specific spatially-indexed attention mechanism to extract the salient facial regions in each individual image, and a temporal attention layer to assign attention weights to frames. This approach not only improves the performance by allowing the model to focus on informative frames and facial areas, but it also offers an interpretable correspondence between the spatial facial regions as well as temporal frames, and the task of age estimation. This work is presented in Chapter 4.

Chapter 5 proposes an model named Siamese Recurrent Networks (SRNs) to address ***Research Question 4*** and learns a similarity measure between two input sequences in a supervised way. In particular, our model learns a latent vectorial representation for each input sequence in such a way that similar sequences are modeled by similar representations, while dissimilar sequences are modeled by a dissimilar representations.

Chapter 6 explores ***Reserach Question 5*** by developing Interacting Attention-gated Recurrent Networks (IARN) to model the interaction between users and items in recommendation systems. In this scenario, two sequences represent a pair of user and item history data and the goal is to predict the preference of the user over the paired item. Our approach not only models the dynamics in both user's and item's information jointly, but also measures the relevance of each time step of two sequences in an interacting way by a novel attention scheme integrated in the recurrent networks. The resulting latent representations of both the user and item are used to predict the preference of user over the item.

Chapter 7 concludes the thesis and presents possible research directions for future work.

2
Multivariate Time Series Classification using the Hidden-Unit Logistic Model

We present a new model for multivariate time series classification, called the hidden-unit logistic model, that uses binary stochastic hidden units to model latent structure in the data. The hidden units are connected in a chain structure that models temporal dependencies in the data. Compared to the prior models for time series classification such as the hidden conditional random field, our model can model very complex decision boundaries because the number of latent states grows exponentially with the number of hidden units. We demonstrate the strong performance of our model in experiments on a variety of (computer vision) tasks, including handwritten character recognition, speech recognition, facial expression, and action recognition. We also present a state-of-the-art system for facial action unit detection based on the hidden-unit logistic model.

This chapter is published as:
Wenjie Pei, Hamdi Dibeklioğlu, David M. J. Tax and Laurens van der Maaten. Multivariate Time-Series Classification Using the Hidden-Unit Logistic Model. IEEE Transactions on Neural Networks and Learning Systems, 2017.

2.1. Introduction

Time series classification is the problem of assigning a single label to a sequence of observations (i.e., to a time series). Time series classification has a wide range of applications in computer vision. A state-of-the-art model for time series classification problem is the hidden-state conditional random field (HCRF) [4], which models latent structure in the data using a chain of k-nomial latent variables. The HCRF has been successfully used in, amongst others, gesture recognition [5], object recognition [4], and action recognition [6]. An important limitation of the HCRF is that the number of model parameters grows linearly with the number of latent states in the model. This implies that the training of complex models with a large number of latent states is very prone to overfitting, whilst models with smaller numbers of parameters may be too simple to represent a good classification function. In this chapter, we propose to circumvent this problem of the HCRF by replacing each of the k-nomial latent variables by a collection of H binary stochastic hidden units. To keep inference tractable, the hidden-unit chains are conditionally independent given the time series and the label. Similar ideas have been explored before in discriminative RBMs [14] for standard classification problems and in hidden-unit CRFs [15] for sequence labeling. The binary stochastic hidden units allow the resulting model, which we call the hidden-unit logistic model (HULM), to represent 2^H latent states using only $O(H)$ parameters. This substantially reduces the amount of data needed to successfully train models without overfitting, whilst maintaining the ability to learn complex models with exponentially many latent states. Exact inference in our proposed model is tractable, which makes parameter learning via (stochastic) gradient descent very efficient. We show the merits of our hidden-unit logistic model in experiments on computer-vision tasks ranging from online character recognition to activity recognition and facial expression analysis. Moreover, we present a system for facial action unit detection that, with the help of the hidden-unit logistic model, achieves state-of-the-art performance on a commonly used benchmark for facial analysis.

The remainder of this chapter is organized as follows. Section 2 reviews prior work on time series classification. Section 3 introduces our hidden-unit logistic model and describes how inference and learning can be performed in the model. In section 4, we present the results of experiments comparing the performance of our model with that of state-of-the-art time series classification models on a range of classification tasks. In section 5, we present a new state-of-the-art system for facial action unit detection based on the hidden-unit logistic model. Section 6 concludes the chapter.

2.2. Related Work

There is a substantial amount of prior work on multivariate time series classification. Much of this work is based on the use of (kernels based on) dynamic time warping (*e.g.,* [16]) or on hidden Markov models (HMMs) [2]. The HMM is a generative model that models the time series data in a chain of latent k-nomial features. Class-conditional HMMs are commonly combined with class priors via Bayes' rule to obtain a time series classification models. Alternatively, HMMs are also frequently used as the base model for Fisher kernel [17], which constructs a time series representation that consists of the gradient a particular time series induces in the parameters of the HMM; the resulting

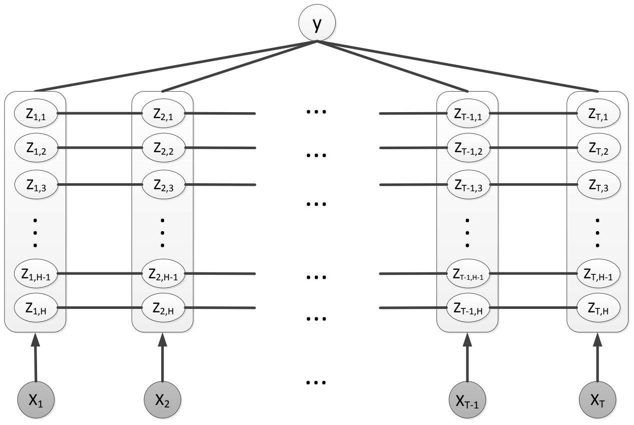


Figure 2.1: Graphical model of the hidden-unit logistic model.

representations can be used on standard classifiers such as SVMs. Some recent work has also tried to learn the parameters of the HMM in such a way as to learn Fisher kernel representations that are well-suited for nearest-neighbor classification [18]. HMMs have also been used as the base model for probability product kernels [19], which fit a single HMM on each time series and define the similarity between two time series as the inner product between the corresponding HMM distributions. A potential drawback of these approaches is that they perform classification based on (rather simple) generative models of the data that may not be well suited for the discriminative task at hand. By contrast, we opt for a discriminative model that does not waste model capacity on features that are irrelevant for classification. In contrast to HMMs, conditional random fields (CRFs [3]) are discriminative models that are commonly used for sequence labeling of time series using so-called linear-chain CRFs. Whilst standard linear-chain CRFs achieve strong performance on very high-dimensional data (e.g., in natural language processing), the linear nature of most CRF models limits their ability to learn complex decision boundaries. Several sequence labeling models have been proposed to address this limitation, amongst which are latent-dynamic CRFs [20], conditional neural fields [21], neural conditional random fields [22], and hidden-unit CRFs [15]. These models introduce stochastic or deterministic hidden units that model latent structure in the data, allowing these models to represent nonlinear decision boundaries. As these prior models were designed for sequence labeling (assigning a label to each frame in the time series), they cannot readily be used for time series classification (assigning a single label to the entire time series). Our hidden-unit logistic model may be viewed as an adaptation of sequence labeling models with hidden units to the time series classification problem. As such, it is closely related to the hidden CRF model [4]. The key difference between our hidden-unit logistic model and the hidden CRF is that our model uses a collection of binary stochastic hidden units instead of a single k-nomial hidden unit, which allows our model to represent exponentially more states with the same number of parameters.

An alternative approach to expanding the number of hidden states of the HCRF is the infinite HCRF (iHCRF), which employs a Dirichlet process to determine the number of hidden states. Inference in the iHCRF can be performed via collapsed Gibbs sampling [23] or variational inference [24]. Whilst theoretically facilitating infinitely many states, the modeling power of the iHCRF is, however, limited to the number of "represented" hidden states. Unlike our model, the number of parameters in the iHCRF thus still grows linearly with the number of hidden states.

2.3. Hidden-Unit Logistic Model

The hidden-unit logistic model is a probabilistic graphical model that receives a time series as input, and is trained to produce a single output label for this time series. Like the hidden-state CRF, the model contains a chain of hidden units that aim to model latent temporal features in the data, and that form the basis for the final classification decision. The key difference with the HCRF is that the latent features are model in H binary stochastic hidden units, much like in a (discriminative) RBM. These hidden units $\mathbf{z}_t$ can model very rich latent structure in the data: one may think about them as carving up the data space into 2^H small clusters, all of which may be associated with particular clusters. The parameters of the temporal chains that connect the hidden units may be used to differentiate between features that are "constant" (i.e., that are likely to be presented for prolonged lengths of time) or that are "volatile" (i.e., that tend to rapidly appear and disappear). Because the hidden-unit chains are conditionally independent given the time series and the label, they can be integrated out analytically when performing inference or learning.

Suppose we are given a time series $\mathbf{x}_{1,\ldots,T} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ of length T in which the observation at the t-th time step is denoted by $\mathbf{x}_t \in \mathbb{R}^D$. Conditioned on this time series, the hidden-unit logistic model outputs a distribution over vectors $\mathbf{y}$ that represent the predicted label using a 1-of-K encoding scheme (i.e., a one-hot encoding): $\forall k : y_k \in \{0,1\}$ and $\sum_k y_k = 1$.

Denoting the stochastic hidden units at time step t by $\mathbf{z}_t \in \{0,1\}^H$, the hidden-unit logistic model defines the conditional distribution over label vectors using a Gibbs distribution in which all hidden units are integrated out:

$$p(\mathbf{y}|\mathbf{x}_{1,\ldots,T}) = \frac{\sum_{\mathbf{z}_{1,\ldots,T}} \exp\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}_{1,\ldots,T}, \mathbf{y})\}}{Z(\mathbf{x}_{1,\ldots,T})}. \tag{2.1}$$

Herein, $Z(\mathbf{x}_{1,\ldots,T})$ denotes a partition function that normalizes the distribution, and is given by:

$$Z(\mathbf{x}_{1,\ldots,T}) = \sum_{\mathbf{y}'} \sum_{\mathbf{z}'_{1,\ldots,T}} \exp\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}'_{1,\ldots,T}, \mathbf{y}')\}. \tag{2.2}$$

The energy function of the hidden-unit logistic model is defined as:

$$\begin{gathered} E(\mathbf{x}_{1,\ldots,T}, \mathbf{z}_{1,\ldots,T}, \mathbf{y}) = \mathbf{z}_1^\top \boldsymbol{\pi} + \mathbf{z}_T^\top \boldsymbol{\tau} + \mathbf{c}^\top \mathbf{y} + \\ \sum_{t=2}^{T} \mathbf{z}_{t-1}^\top \mathrm{diag}(\mathbf{A}) \mathbf{z}_t + \sum_{t=1}^{T} \left[\mathbf{z}_t^\top \mathbf{W} \mathbf{x}_t + \mathbf{z}_t^\top \mathbf{V} \mathbf{y} + \mathbf{z}_t^\top \mathbf{b} \right]. \end{gathered} \tag{2.3}$$

The graphical model of the hidden-unit logistic model is shown in Fig. 2.1.

Next to a number of bias terms, the energy function in (2.3) consists of three main components: (1) a term with parameters $\mathbf{W}$ that measures to what extent particular latent features are present in the data; (2) a term parametrized by $\mathbf{A}$ that measures the compatibility between corresponding hidden units at time step $t-1$ and t; and (3) a prediction term with parameters $\mathbf{V}$ that measures the compatibility between the latent features $\mathbf{z}_{1,\dots,T}$ and the label vector $\mathbf{y}$. Please note that hidden units in consecutive time steps are connected using a chain structure rather than fully connected; we opt for this structure because exact inference is intractable when consecutive hidden units are fully connected. Intuitively, the hidden-unit logistic model thus assigns a high probability to a label (for a particular input) when there are hidden unit states that are both "compatible" with the observed data and with a particular label. As the hidden units can take 2^H different states, this leads to a model that can represent highly nonlinear decision boundaries. The following subsections describe the details of inference and learning in the hidden-unit logistic model. The whole process is summarized in Algorithm 1.

Algorithm 1 The inference and learning of *HULM*.

```
Input: A time series $\mathbf{x}_{1,\dots,T} = \{\mathbf{x}_1, \dots, \mathbf{x}_T\}$ and the associated labels $\mathbf{y}$.
Output:
  • The conditional distribution over predicted labels $p(\mathbf{y}|\mathbf{x}_{1,\dots,T})$ (inference);
  • The conditional log-likelihood of the training data: $\mathcal{L}(\Theta)$ (inference);
  • The gradient of $\mathcal{L}(\Theta)$ with respect to each parameter $\theta \in \Theta$: $\frac{\partial \mathcal{L}}{\partial \theta}$ (learning).
1: Compute the potential functions $\Psi_{t,h}(\mathbf{x}_t, z_{t-1,h}, z_{t,h}, \mathbf{y})$ for each hidden unit $h$ ($1 \le h \le H$) at each time step $t$ ($1 \le t \le T$) as indicated in Equation 2.5.
2: for $t = 1 \to T$ do
3:     Calculate the forward message $\alpha_{t,h,k}$ with $k \in \{0,1\}$ by Equation 2.9.
4: end for
5: for $t = T \to 1$ do
6:     Compute the backward message $\beta_{t,h,k}$ by Equation 2.10.
7: end for
8: Compute the intermediate term $M(\mathbf{x}_{1,\dots,T}, \mathbf{y}) = \sum_{\mathbf{z}_{1,\dots,T}} \exp\{E(\mathbf{x}_{1,\dots,T}, \mathbf{z}_{1,\dots,T}, \mathbf{y})$ either with $\alpha_{T,h,k}$ or with $\beta_{1,h,k}$ by Equation 2.11.
9: Compute the partition function $Z(\mathbf{x}_{1,\dots,T}) = \sum_{\mathbf{y}'} M(\mathbf{x}_{1,\dots,T}, \mathbf{y}')$.
10: The conditional distribution over predicted labels is calculated by $p(\mathbf{y}|\mathbf{x}_{1,\dots,T}) = \frac{M(\mathbf{x}_{1,\dots,T}, \mathbf{y})}{Z(\mathbf{x}_{1,\dots,T})}$.
11: The conditional log-likelihood of the training data $\mathcal{L}(\Theta)$ is calculated by Equation 3.7.
12: Compute the marginal distribution over a chain edge $\xi_{t,h,k,l} = P(z_{t,h} = k, z_{t+1,h} = l|\mathbf{x}_{1,\dots,T}, \mathbf{y})$ by Equation 2.13 using forward and backward messages.
13: The gradient of $\mathcal{L}(\Theta)$ with respect to each parameter $\theta \in \Theta$: $\frac{\partial \mathcal{L}}{\partial \theta}$ is calculated by Equation 2.15 and 2.16 using marginal distribution $\xi_{t,h,k,l}$.
```

2.3.1. INFERENCE

The main inferential problem given an observation $\mathbf{x}_{1,\ldots,T}$ is the evaluation of predictive distribution $p(\mathbf{y}|\mathbf{x}_{1,\ldots,T})$. The key difficulty in computing this predictive distribution is the sum over all $2^{H\times T}$ hidden unit states:

$$M(\mathbf{x}_{1,\ldots,T},\mathbf{y}) = \sum_{\mathbf{z}_{1,\ldots,T}} \exp\{E(\mathbf{x}_{1,\ldots,T},\mathbf{z}_{1,\ldots,T},\mathbf{y})\}. \tag{2.4}$$

The chain structure of the hidden-unit logistic model allows us to employ a standard forward-backward algorithm that can compute $M(\cdot)$ in computational time linear in T.

Specifically, defining potential functions that contain all terms that involve time t and hidden unit h:

$$\Psi_{t,h}(\mathbf{x}_t, z_{t-1,h}, z_{t,h}, \mathbf{y}) = \exp\{\mathbf{z}_{t-1,h}\mathbf{A}_h z_{t,h} + z_{t,h}\mathbf{W}_h\mathbf{x}_t + z_{t,h}\mathbf{V}_h\mathbf{y} + z_{t,h}b_h\} \tag{2.5}$$

ignoring bias terms, and introducing virtual hidden units $\mathbf{z}_0 = \mathbf{0}$ at time $t = 0$, we can rewrite $M(\cdot)$ as:

$$\begin{aligned}
M(\cdot) &= \sum_{\mathbf{z}_{1,\ldots,T}} \prod_{t=1}^{T}\prod_{h=1}^{H} \Psi_{t,h}(\mathbf{x_t}, z_{t-1,h}, z_{t,h}, \mathbf{y}) \\
&= \prod_{h=1}^{H}\left[\sum_{z_{1,h},\ldots,z_{T,h}} \prod_{t=1}^{T} \Psi_{t,h}(\mathbf{x}_t, z_{t-1,h}, z_{t,h}, \mathbf{y})\right] \\
&= \prod_{h=1}^{H}\left[\sum_{z_{T-1,h}} \Psi_{T,h}(\mathbf{x}_T, z_{T-1,h}, z_{T,h}, \mathbf{y}) \sum_{z_{T-2,h}} \Psi_{T-1,h}(\mathbf{x}_{T-1}, z_{T-2,h}, z_{T-1,h}, \mathbf{y}) \ldots\right].
\end{aligned} \tag{2.6}$$

In the above derivation, it should be noted that the product over hidden units h can be pulled outside the sum over all states $\mathbf{z}_{1,\ldots,T}$ because the hidden-unit chains are conditionally independent given the data $\mathbf{x}_{1,\ldots,T}$ and the label $\mathbf{y}$. Subsequently, the product over time t can be pulled outside the sum because of the (first-order) Markovian chain structure of the temporal connections between hidden units.

In particular, the required quantities can be evaluated using the forward-backward algorithm, in which we define the forward messages $\alpha_{t,h,k}$ with $k \in \{0,1\}$ as:

$$\alpha_{t,h,k} = \sum_{z_{1,h},\ldots,z_{t-1,h}} \prod_{t'=1}^{t} \Psi_{t',h}(\mathbf{x}_{t'}, z_{t'-1,h}, z_{t',h} = k, \mathbf{y}) \tag{2.7}$$

and the backward messages $\beta_{t,h,k}$ as:

$$\beta_{t,h,k} = \sum_{z_{t+1,h},\ldots,z_{T,h}} \prod_{t'=t+1}^{T} \Psi_{t',h}(\mathbf{x}_{t'+1}, z_{t',h} = k, z_{t'+1,h}, \mathbf{y}). \tag{2.8}$$

These messages can be calculated recursively as follows:

$$\alpha_{t,h,k} = \sum_{i\in\{0,1\}} \Psi_{t,h}(\mathbf{x}_t, z_{t-1,h} = i, z_{t,h} = k, \mathbf{y})\alpha_{t-1,h,i} \tag{2.9}$$

$$\beta_{t,h,k} = \sum_{i\in\{0,1\}} \Psi_{t+1,h}(\mathbf{x}_{t+1}, z_{t,h} = k, z_{t+1,h} = i, \mathbf{y})\beta_{t+1,h,i}. \tag{2.10}$$

The value of $M(\mathbf{x}_{1,\dots,T},\mathbf{y})$ can readily be computed from the resulting forward messages or backward messages:

$$M(\mathbf{x}_{1,\dots,T},\mathbf{y}) = \prod_{h=1}^{H}\left(\sum_{k\in\{0,1\}}\alpha_{T,h,k}\right) = \prod_{h=1}^{H}\left(\sum_{k\in\{0,1\}}\beta_{1,h,k}\right). \tag{2.11}$$

To complete the evaluation of the predictive distribution, we compute the partition function of the predictive distribution by summing $M(\mathbf{x}_{1,\dots,T},\mathbf{y})$ over all K possible labels: $Z(\mathbf{x}_{1,\dots,T}) = \sum_{\mathbf{y}'} M(\mathbf{x}_{1,\dots,T},\mathbf{y}')$. Indeed, inference in the hidden-unit logistic model is linear in both the length of the time series T and in the number of hidden units H.

Another inferential problem that needs to be solved during parameter learning is the evaluation of the marginal distribution over a chain edge:

$$\xi_{t,h,k,l} = P(z_{t,h}=k, z_{t+1,h}=l|\mathbf{x}_{1,\dots,T},\mathbf{y}). \tag{2.12}$$

Using a similar derivation, it can be shown that this quantity can also be computed from the forward and backward messages:

$$\xi_{t,h,k,l} = \frac{\alpha_{t,h,k}\cdot\Psi_{t+1,h}(\mathbf{x}_{t+1}, z_{t,h}=k, z_{t+1,h}=l, y)\cdot\beta_{t+1,h,l}}{\sum_{k\in\{0,1\}}\alpha_{T,h,k}}. \tag{2.13}$$

2.3.2. PARAMETER LEARNING

Given a training set $\mathscr{D} = \{(\mathbf{x}^(n)_{1,\dots,T},\mathbf{y}^{(n)})\}_{n=1,\dots,N}$ containing N pairs of time series and their associated label. We learn the parameters $\mathbf{\Theta} = \{\pi,\tau,\mathbf{A},\mathbf{W},\mathbf{V},\mathbf{b},\mathbf{c}\}$ of the hidden-unit logistic model by maximizing the conditional log-likelihood of the training data with respect to the parameters:

$$\begin{aligned}\mathscr{L}(\Theta) &= \sum_{n=1}^{N}\log p\left(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}_{1,\dots,T}\right)\\ &= \sum_{n=1}^{N}\left[\log M\left(\mathbf{x}^{(n)}_{1,\dots,T},\mathbf{y}^{(n)}\right) - \log\sum_{\mathbf{y}'} M\left(\mathbf{x}^{(n)}_{1,\dots,T},\mathbf{y}'\right)\right].\end{aligned} \tag{2.14}$$

We augment the conditional log-likelihood with L2-regularization terms on the parameters $\mathbf{A}$, $\mathbf{W}$, and $\mathbf{V}$. As the objective function is not amenable to closed-form optimization (in fact, it is not even a convex function), we perform optimization using stochastic gradient descent on the negative conditional log-likelihood. The gradient of the conditional log-likelihood with respect to a parameter $\theta\in\Theta$ is given by:

$$\frac{\partial\mathscr{L}}{\partial\theta} = \mathbb{E}\left[\frac{\partial E(\mathbf{x}_{1,\dots,T},\mathbf{z}_{1,\dots,T},\mathbf{y})}{\partial\theta}\right]_{P(\mathbf{z}_{1,\dots,T}|\mathbf{x}_{1,\dots,T},\mathbf{y})} - \mathbb{E}\left[\frac{\partial E(\mathbf{x}_{1,\dots,T},\mathbf{z}_{1,\dots,T},\mathbf{y})}{\partial\theta}\right]_{P(\mathbf{z}_{1,\dots,T},\mathbf{y}|\mathbf{x}_{1,\dots,T})}. \tag{2.15}$$

where we omitted the sum over training examples for brevity. The required expectations can readily be computed using the inference algorithm described in the previous subsection.

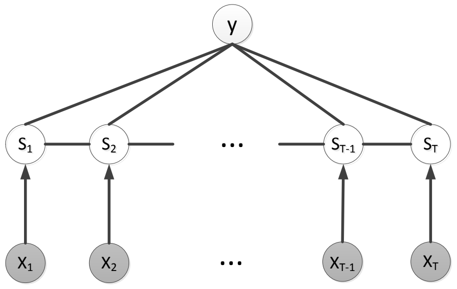


Figure 2.2: Graphical model of the HCRF.

For example, defining $r(\Theta) = z_{t-1,h}\mathbf{A}_h z_{t,h} + z_{t,h}\mathbf{W}_h\mathbf{x}_t + z_{t,h}\mathbf{V}_h y + z_{t,h} b_h$ for notational simplicity, the first expectation can be computed as follows:

$$\begin{aligned}
&\mathbb{E}\left[\frac{\partial E(\mathbf{x}_{1,\dots,T},\mathbf{z}_{1,\dots,T},\mathbf{y})}{\partial\theta}\right]_{P(\mathbf{z}_{1,\dots,T}|\mathbf{x}_{1,\dots,T},\mathbf{y})}\\
&= \sum_{\mathbf{z_{1,\dots,T}}} P(\mathbf{z}_{1,\dots,T}|\mathbf{x}_{1,\dots,T},\mathbf{y})\left(\sum_{t=1}^{T}\sum_{h=1}^{H}\frac{\partial r(\Theta)}{\partial\theta}\right)\\
&= \sum_{t=1}^{T}\sum_{k\in\{0,1\}}\sum_{l\in\{0,1\}}\left(\xi_{t-1,h,k,l}\cdot\frac{\partial r(\Theta)}{\partial\theta}\right).
\end{aligned} \tag{2.16}$$

The second expectation is simply an average of these expectations over all K possible labels $\mathbf{y}$.

2.3.3. Comparison with HCRF

The hidden-state CRF's graphical model, shown in Figure 2.2, is similar to that of the hidden-unit logistic model (HULM). They are both discriminative models which employ hidden variables to model the latent structures. The key difference between the two models is in the way the hidden units are defined: whereas the hidden-unit logistic model uses a large number of (conditionally independent) binary stochastic hidden units to represent the latent state, the HCRF uses a single multinomial unit (much like a hidden Markov model). As a result, there are substantial differences in the distributions that the HCRF and HULM can model. In particular, the HULM is a *product of experts* model[1], whereas the HCRF is a *mixture of experts* model [25, 26]. A potential advantage of product distributions over mixture distributions is in the "sharpness" of the distributions [25]. Consider, for instance, two univariate Gaussian distributions with equal variance but different means: whereas a mixture those distributions will have higher variance than each of the individual Gaussians, a product of the distribution will have lower variance and, therefore, model a much sharper distribution. This can be a substantial

[1]The expression of $M(\cdot)$ presented earlier clearly shows that HULM models a distribution that is a product over H experts.

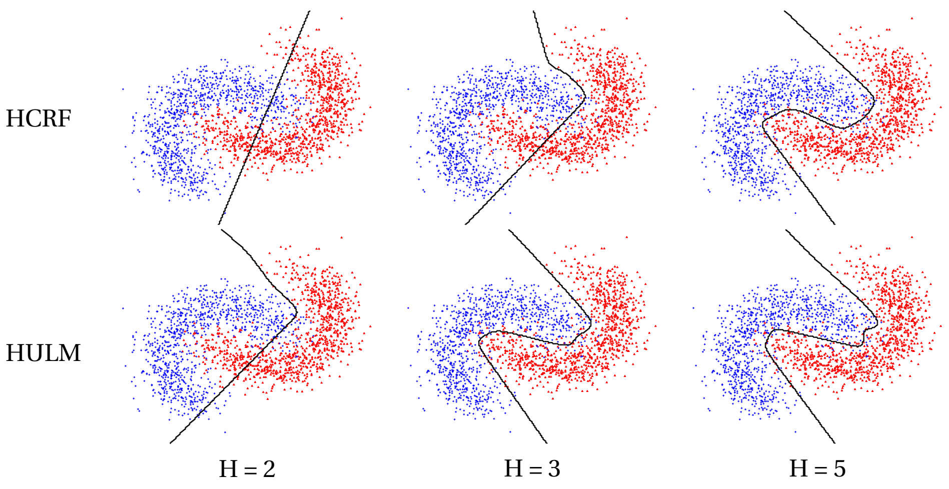


Figure 2.3: Comparison of HCRF and HULM for binary classification on the banana dataset (ignoring the time series aspect of the models) with the same number of hidden units H. The black lines show the decision boundaries learned by both models.

advantage when modeling high-dimensional distributions in which much of the probability mass tends to be lost in the tails. There also appear to be differences in the total number of modes that can be modeled by product and mixture distributions in high-dimensional spaces (although it is hitherto unknown how many modes a mixture of unimodal distributions maximally contains [27]). Indeed, theoretical results suggest that product distributions have more modeling power with the same number of parameters than mixture distributions; for certain distributions, mixture distributions even require exponentially more parameters than their product counterparts [28].

To empirically explore these differences, we performed a simple experiment in which we ignore the temporal component of the HULM and HCRF models (to facilitate visualizations), and train the models on a binary two-dimensional classification problem. Fig. 2.3 shows the decision boundaries learned by HULM and HCRF models with the same number of hidden parameters on our test dataset. Indeed, the results suggest that the HULM can model more complex decision boundaries than HCRFs with the same number of parameters.

In our experiments, we also observed that HULM models can be trained faster than HCRF models. We illustrate this in Fig. 2.4, which shows the training time of both models (with the same experimental configuration) on a facial expression dataset. Whilst these differences in training speed may be partly due to implementation differences, they are also the result of the constraint we introduce that the transition matrix between hidden units in consecutive time steps is diagonal. As a result, the computation of the forward message α in Eqn. 2.7 and backward message β in Eqn. 2.8 is linear in the number of hidden units H. Consequently, the quantities $M(\mathbf{x}_{1,\ldots,T}, \mathbf{y})$ in Eqn. 2.11 and marginal distribution $\xi_{t,h,k,l}$ in Eqn. 2.12 can be calculated in $O(THD)$. Taking into account the number of label classes Y, the overall computational complexity of HULM is $O(TH(D+Y))$. By contrast, the complexity of HCRF is $O(TH^2(D+Y))$ [4]. This difference facilitates the

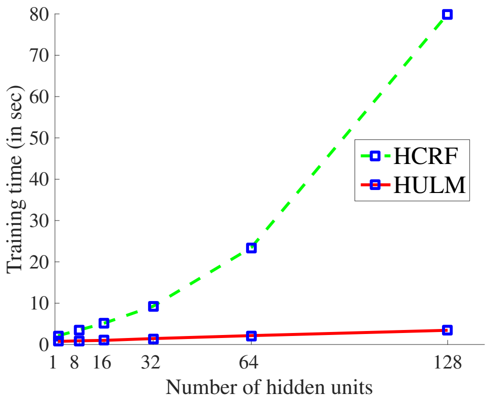


Figure 2.4: Running time of a single training epoch of the HULM and HCRF model on the facial expression data (CK+) described in Sec. 2.4.1 as a function of the number of hidden units. We used stochastic gradient descent with the same configuration to train both the HULM and the HCRF.

use or larger numbers of hidden units H in the HULM model than in the HCRF model. (Admittedly, it is straightforward to develop a diagonal version of the HCRF model, also.)

2.4. Experiments

To evaluate the performance of the hidden-unit logistic model, we conducted classification experiments on eight different problems involving seven time series data sets. Since univariate times series can be considered as a special case of multivariate time series, we first performed experiments on two univariate time series data sets introduced by UCR Archive [29]: (1) Synthetic Control and (2) Swedish Leaf, subsequently we evaluated our models on five multivariate time series data sets : (1) an online handwritten character data set (OHC) [30]; (2) a data set of Arabic spoken digits (ASD) [31]; (3) the Cohn-Kanade extended facial expression data set (CK+) [32]; (4) the MSR Action 3D data set (Action) [33]; and (5) the MSR Daily Activity 3D data set (Activity) [34]. The seven data sets are introduced in 2.4.1, the experimental setup is presented in 2.4.2, and the results of the experiments are in 2.4.3.

2.4.1. Data Sets

Univariate Time Series Data Sets

We performed experiments on two univariate UCR data sets: *Synthetic Control* and *Swedish Leaf*. *Synthetic Control* is a relatively easy data set containing 300 training samples and 300 test samples grouped into 6 classes. All samples in it have the identical length of time series equaling to 60. We enrich the univariate feature by windowing 10 frames into 1 frame resulting in the 10 dimensions for each frame. *Swedish Leaf* is a challenging data set which consists of 500 training samples and 625 test samples with the length of 128 frames spreading in 15 classes. Similarly, we pre-process the data by windowing the features of 30 frames into 1 frame with 30-dimension feature.

MULTIVARIATE TIME SERIES DATA SETS

The online handwritten character dataset [30] is a pen-trajectory time series data set that consists of three dimensions at each time step, *viz.*, the pen movement in the x-direction and y-direction, and the pen pressure. The data set contains 2858 time series with an average length of 120 frames. Each time series corresponds to a single handwritten character that has one of 20 labels. We pre-process the data by windowing the features of 10 frames into a single feature vector with 30 dimensions.

The Arabic spoken digit dataset contains 8800 utterances [31], which were collected by asking 88 Arabic native speakers to utter all 10 digits ten times. Each time series consists of 13-dimensional MFCCs which were sampled at 11,025Hz, 16-bits using a Hamming window. We enrich the features by windowing 3 frames into 1 frames resulting in the 13×3 dimensions for each frame of the features while keeping the same length of time series. We use two different versions of the spoken digit dataset: (1) a *digit* version in which the uttered digit is the class label and (2) a *voice* version in which the speaker of a digit is the class label.

The Cohn-Kanade extended facial expression data set [32] contains 593 image sequences (videos) from 123 subjects. Each video shows a single facial expression. The videos have an average length of 18 frames. A subset of 327 of the videos, which have validated label corresponding to one of seven emotions (anger, contempt, disgust, fear, happiness, sadness, and surprise), are used in our experiments. We adopt the publicly available shape features used in [35] as the feature representation for our experiments. These features represent each frame by the variation of 68 feature point locations (x, y) with respect to the first frame [32], which leads to 136-dimensional feature representation for each frame in the video.

The MSR Action 3D data set [33] consists of RGB-D videos of people performing certain actions. The data set contains 567 videos with an average length of 41 frames. Each video should be classified into one of 20 actions such as “high arm wave”, “horizontal arm wave”, and “hammer”. We use the real-time skeleton tracking algorithm of [36] to extract the 3D joint positions from the depth sequences. We use the 3D joint position features (pairwise relative positions) proposed in [34] as the feature representation for the frames in the videos. Since we track a total of 20 joints, the dimensionality of the resulting feature representation is $3 \times \binom{20}{2} = 570$, where $\binom{20}{2}$ is the number of pairwise distances between joints and 3 is dimensionality of the (x, y, z) coordinate vectors. It should be noted that we only extract the joints features to evaluate performances of different time series classification models mentioned in this chapter rather than pursue state-of-the-art action-recognition performance, hence it is not fair to compare the reported results in Table 1 directly to the performance of the ad-hoc action-recognition methods which employ 2D/3D appearance features.

The MSR Daily Activity 3D data set [34] contains RGB-D videos of people performing daily activities. The data set also contains 3D skeletal joint positions, which are extracted using the Kinect SDK. The videos need to be classified into one of 16 activity types, which include “drinking”, “eating”, “reading book”, *etc.* Each activity is performed by 10 subjects in two different poses (namely, while sitting on a sofa and while standing), which leads to a total of 320 videos. The videos have an average length of 193 frames. To represent each frame, we extract 570-dimensional 3D joint position features.

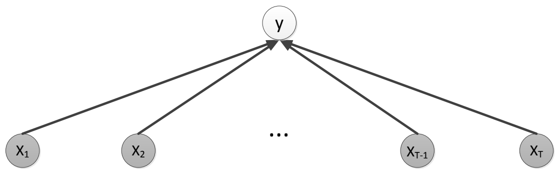


Figure 2.5: Graphical model of the naive logistic model.

2.4.2. Experimental Setup

In our experiments, the model parameters $\mathbf{A}, \mathbf{W}, \mathbf{V}$ of the hidden-unit logistic model were initialized by sampling them from a Gaussian distribution with a variance of 10^{-3}. The initial-state parameter $\boldsymbol{\pi}$, final-state parameter $\boldsymbol{\tau}$ and the bias parameters $\mathbf{b}, \mathbf{c}$ were initialized to 0. Training of our model is performed using a standard stochastic gradient descent procedure; the learning rate is decayed during training. We set the number of hidden units H to 100. The L2-regularization parameter λ was tuned by minimizing the error on a small validation set. Code reproducing the results of our experiments is available on *https://github.com/wenjiepei/HULM*.

We compare the performance of our hidden-unit logistic model with that of three other time series classification models: (1) the naive logistic model shown in Fig. 2.5, (2) the popular HCRF model [4], and (3) Fisher kernel learning model [18]. Details of these models are given below.

Naive logistic model The naive logistic model is a linear logistic model that shares parameters between all time steps, and makes a prediction by summing (or equivalently, averaging) the inner products between the model weights and feature vectors over time before applying the softmax function. Specifically, the naive logistic model defined the following conditional distribution over the label y given the time series data $\mathbf{x}_{1,\ldots,T}$:

$$p(\mathbf{y}|\mathbf{x}_{1,\ldots,T}) = \frac{\exp\{E(\mathbf{x}_{1,\ldots,T}, \mathbf{y})\}}{Z(\mathbf{x}_{1,\ldots,T})},$$

where the energy function is defined as

$$E(\mathbf{x}_{1,\ldots,T}, \mathbf{y}) = \sum_{t=1}^{T} (\mathbf{y}^T \mathbf{W} \mathbf{x}_t) + \mathbf{c}^T \mathbf{y}.$$

The corresponding graphical model is shown in Fig. 2.5. We include the naive logistic model in our experiments to investigate the effect of adding hidden units to models that average energy contributions over time.

Hidden CRF The Hidden-state CRF model is similar to HULM and thereby an important baseline. We performed experiments using the hidden CRF implementation of [37]. Following [4], we trained HCRFs with 10 latent states on all data sets. (We found it was computationally infeasible to train HCRFs with more than 10 latent states.) We tune the L2-regularization parameter of the HCRF on a small validation set.

Fisher kernel learning In addition to comparing with HCRFs, we compare the performance of our model with that of the recently proposed Fisher kernel learning (FKL) model [18]. We selected the FKL model for our experiments because [18] reports strong performance on a range of time series classification problems. We trained FKL models based on hidden Markov models with 10 hidden states (the number of hidden states was set identical to that of the hidden CRF). Subsequently, we computed the Fisher kernel representation and trained a linear SVM on the resulting features to obtain the final classifier. The slack parameter C of the SVM is tuned on a small validation set.

2.4.3. RESULTS

We perform two sets of experiments with the hidden-unit logistic model: (1) a set of experiments in which we evaluate the performance of the model (and of the hidden CRF) as a function of the number of hidden units and (2) a set of experiments in which we compare the performance of all models on all data sets. The two sets of experiments are described separately below.

EFFECT OF VARYING THE NUMBER OF HIDDEN UNITS.

We first conduct experiments on the ASD data set to investigate the performance of the hidden-unit logistic model as a function of the number of hidden units. The results of these experiments are shown in Fig. 2.6. The results presented in the figure show that the error initially decreases when the number of hidden unit increases, because adding hidden units adds complexity to the model that allows it to better fit the data. However, as the hidden unit number increases further, the model starts to overfit on the training data despite the use of L2-regularization.

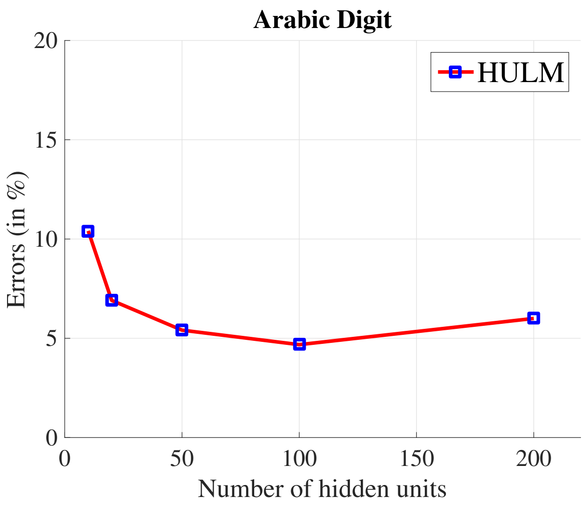


Figure 2.6: Generalization error (in %) of the hidden-unit logistic model on the Arabic speech data set as a function of the number of hidden units.

We performed a similar experiment on the CK+ facial expression data set, in which we also performed comparisons with the hidden CRF for a range of values for the number of hidden states. Fig. 2.7 presents the results of these experiments. On the CK+ data set, there are no large fluctuations in the errors of the HULM as the hidden parameter number increases. The figure also shows that the hidden-unit logistic model outperforms the hidden CRF irrespective of the number of hidden units. For instance, a hidden-unit logistic model with 10 hidden units outperforms even a hidden CRF with 100 hidden parameters. This result illustrates the potential merits of using models in which the number of latent states grows exponentially with the number of parameters.

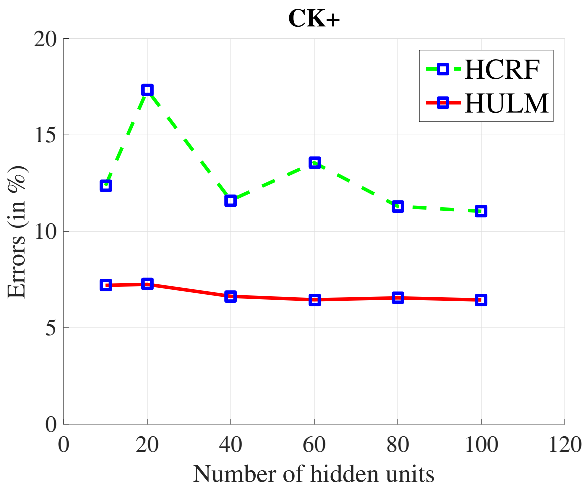


Figure 2.7: Generalization error (in %) of the hidden-unit logistic model and the hidden CRF on the CK+ data set as a function of the number of hidden units.

Comparison with Modern Time Series Classifiers.

In a second set of experiments, we compare the performance of the hidden-unit logistic model with that of the naive logistic model, Fisher kernel learning, and the hidden CRF on all eight problems. In our experiments, the number of hidden units in the hidden-unit logistic model was set to 100; following [4], the hidden CRF used 10 latent states. The results of our experiments are presented in Table 2.1, and are discussed for each data set separately below.

Synthetic Control *Synthetic Control* is a simple univariate time-series classification problem from the UCR time series classification archive [29]. Table 2.1 shows the generalization errors by four time series classification models mentioned above. HULM model achieves the best performance with 1.33%, which is close to the state-of-the-art performance on this dataset (0.7%) reported in [29]. This is an encouraging result, in particular, because the HULM method is not at all tuned towards solving univariate time-series classification problems.

Table 2.1: Generalization errors (%) on all eight problems by four time series classification models: the naive logistic model (NL), Fisher kernel learning (FKL), the hidden CRF (HCRF), and the hidden-unit logistic model (HULM). The best performance on each data set is boldfaced. See text for details.

Dataset	Dim.	Classes	Model			
			NL	FKL	HCRF	HULM
Synthetic Control	1×10	6	20.00	2.33	1.67	**1.33**
Swedish Leaf	1×30	15	52.64	10.24	12.80	**10.08**
OHC	3×10	20	23.67	**0.97**	1.58	1.30
ASD-digit	13×3	10	25.50	6.91	**3.68**	4.68
ASD-voice	13×3	88	36.91	6.36	20.40	**5.45**
CK+	136	7	9.20	10.81	11.04	**6.44**
Action	570	20	40.40	40.74	**34.68**	35.69
Activity	570	16	59.38	**43.13**	62.50	45.63
Avg. rank	–	–	3.50	2.38	2.63	**1.50**

Swedish Leaf *Swedish Leaf* is a much more challenging univariate time-series classification problem. Whereas the naive logistic model performs very poorly on this data set, all other three models achieves good performance, with the HULM slightly outperforming the other methods. It is worth mentioning that all three methods outperform the dynamic time warping approach that achieves 15.4% on this dataset reported in [29]. We surmise the strong performance of our models is due to the non-linear features transformations these models perform. The state-of-the-art performance (6.24%) on this dataset is obtained by the recursive edit distance kernels (REDK) [38] which aims to embed (univariate) time series in time-warped Hilbert spaces while preserving the properties of elastic measure.

Online handwritten character dataset (OHC) Following the experimental setup in [18], we measure the generalization error of all four models on the online handwritten character dataset using 10-fold cross validation. The average generalization error of each model is shown in Table 2.1. Whilst the naive logistic model performs very poorly on this data set, all three other methods achieve very low error rates. The best performance is obtained by FKL, but the differences between the models are very small on this data set, presumably, due to a ceiling effect.

Arabic spoken digits dataset (ASD-digit) Following [31], the error rates for the Arabic spoken digits data set with *digit* as the class label in Table 2.1 were measured using a fixed training/test division: 75% of samples are used for training and left 25% of samples compose test set. The best performance on this data set is obtained by the hidden CRF model (3.68%), whilst our model has a slightly higher error of 4.68%, which in turn is better than the performance of FKL. It should be noted that the performance of the hidden

CRF and the hidden-unit logistic model are better than the error rate of 6.88% reported in [31] (on the same training/test division).

Arabic spoken digits dataset (ASD-voice) In the experiment setup in which the speaker of a digit is the class label for the ASD data set, the classification problem becomes much harder than the *digit* version due to much more classes involved (88 subjects). Table 2.1 shows that HULM achieves the best performance and FKL also performs very well. While the naive logistic model unsurprisingly performs very poorly, it should be noted that HULM significantly outperforms HCRF which reveals the advantage of HULM in the case of challenging classification problem.

Facial expression dataset (CK+) Table 2.1 presents generalization errors measured using 10-fold cross-validation. Folds are constructed in such a way that all videos by the same subject are in the same fold (the subjects appearing in test videos were not present in the training set). On the CK+ data set, the hidden-unit logistic model substantially outperforms the hidden CRF model, obtaining an error of 6.44%. Somewhat surprisingly, the naive logistic model also outperforms the hidden CRF model with an error of 9.20%. A possible explanation for this result is that the classifying these data successfully does not require exploitation of temporal structure: many of the expressions can also be recognized well from a single frame. As a result, the naive logistic model may perform well even though it simply averages over time. This result also suggests that the hidden CRF model may perform poorly on high-dimensional data (the CK+ data is 136-dimensional) despite performing well on low-dimensional data such as the handwritten character data set (3-dimensional) and the Arabic spoken data set (13-dimensional).

MSR Action 3D data set (Action) To measure the generalization error of the time series classification models on the MSR Action 3D dataset, we followed the experimental setup of [34]: we used all videos of the five subjects for training, and used the videos of the remaining five subjects for testing. Table 2.1 presents the average generalization error on the videos of the five test subjects. The four models perform quite similarly, although the hidden CRF and the hidden-unit logistic model do appear to outperform the other two models somewhat. The state-of-the-art performance on this dataset is achieved by [39], which performs temporal down-sampling associated to elastic kernel machine learning. Nevertheless, it performs cross-validation on the all possible (252) combinations of training/test subject divisions. Hence the direct comparison with our model is not straightforward.

MSR Daily Activity 3D data set (Activity) On the MSR Daily Activity data set, we use the same experimental setup as on the action data set: five subjects are used for training and five for testing. The results in Table 2.1 show that the hidden-unit logistic model substantially outperforms the hidden CRF on this challenging data set (but FKL performs slightly better).

In terms of the average rank over all data sets, the hidden-unit logistic model performs very strongly. Indeed, it substantially outperforms the hidden CRF model, which illustrates that using a collection of (conditionally independent) hidden units may be a more effective way to represent latent states than a single multinomial unit. FKL also performs quite well in our experiments, although its performance is slightly worse than that of the hidden-unit logistic model. However, it should be noted here that FKL scales poorly to large data sets: its computational complexity is quadratic in the number of time series, which limits its applicability to relatively small data sets (with fewer than, say, 10,000 time series). By contrast, the training of hidden-unit logistic models scales linearly in the number of time series and, moreover, can be performed using stochastic gradient descent.

2.5. Application to Facial AU Detection

In this section, we present a system for facial action unit (AU) detection that is based on the hidden-unit logistic model. We evaluate our system on the Cohn-Kanade extended facial expression database (CK+) [32], evaluating its ability to detect 10 prominent facial action units: namely, AU1, AU2, AU4, AU5, AU6, AU7, AU12, AU15, AU17, and AU25. We compare the performance of our facial action unit detection system with that of state-of-the-art systems for this problem. Before describing the results of these experiments, we first describe the feature extraction of our AU detection system and the setup of our experiments.

2.5.1. Facial Features

We extract two types of features from the video frames in the CK+ data set: (1) shape features and (2) appearance features. Our features are identical to the features used by the system described in [35]; the features are publicly available online. For completeness, we briefly describe both types of features below.

The *shape features* represent each frame by the vertical/horizontal displacements of facial landmarks with respect to the first frame. To this end, automatically detected/tracked 68 landmarks are used to form 136-dimensional time series. All landmark displacements are normalized by removing rigid transformations (translation, rotation, and scale).

The *appearance features* are based on grayscale intensity values. To capture the change in facial appearance, face images are warped onto a base shape, where feature points are in the same location for each face. After this shape normalization procedure, the grayscale intensity values of the warped faces can be readily compared. The final appearance features are extracted by subtracting the warped textures from the warped texture in the first frame. The dimensionality of the appearance feature vectors is reduced using principal components analysis as to retain 90% of the variance in the data. This leads to 439-dimensional appearance feature vectors, which are combined with the shape features to form the final feature representation for the video frames. For further details on the feature extraction, we refer to [35].

2.5.2. Experimental Setup

To gauge the effectiveness of the hidden-unit logistic model in facial AU detection, we performed experiments on the CK+ database [32]. The database consists of 593 image sequences (videos) from 123 subjects with an average length of 18.1 frames. The videos show expressions from neutral face to peak formation, and include annotations for 30

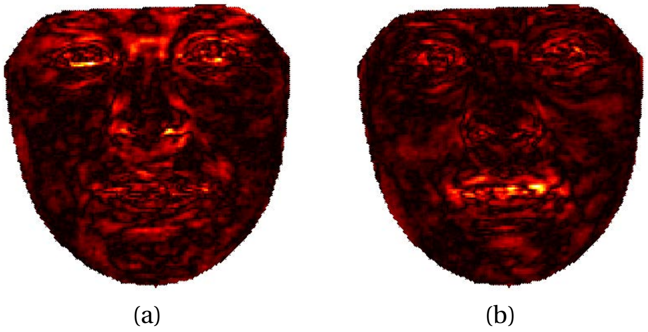


Figure 2.8: Visualization of |**W**| for (a) AU4 and (b) AU25. Brighter colors correspond to image regions with higher weights.

action units. In our experiments, we only consider the 10 most frequent action units.

Our AU detection system employs 10 separate binary classifiers for detecting action units in the videos. In other words, we train a separate HULM for each facial action unit. An individual model thus distinguishes between the presence and non-presence of the corresponding action unit. We use a 10-fold cross-validation scheme to measure the performance of the resulting AU detection system: we randomly select one test fold containing 10% of the videos, and use remaining nine folds are used to train the system. The folds are constructed such that there is no subject overlap between folds: *i.e.*, subjects appearing in the test data were not present in the training data.

2.5.3. Results

We ran experiments using the HULM on three feature sets: (1) shape features, (2) appearance features, and (3) a concatenation of both feature vectors. We measure the performance of our system using the area under ROC curve (AUC). Table 2.2 shows the results for HULM, and for the baseline in [35]. The results show that the HULM outperforms the CRF baseline of [35], with our best model achieving an AUC that is approximately 0.03 higher than the best result of [35].

Table 2.2: AUC of the HULM and the CRF baseline in [35] for three feature sets. *In [35], the combined feature set also includes SIFT features.

Method	Feature Set		
	Shape	Appearance	Combination
HULM	0.9101	0.9197	0.9253
[35]	0.8902	0.8971	0.8647*

To obtain insight in what features are modeled by the HULM hidden units, we visualized a single column of |**W**| in Fig. 2.8 for the AU4 and AU25 models that were trained on appearance features. Specifically, we selected the hidden unit with the highest corresponding **V**-value for visualization, as this hidden unit apparently models the most discriminative features. The figure shows that the appearance of the eyebrows is most important in the AU4 model (brow lowerer), whereas the mouth region is most important in the AU25 model (lips part).

Table 2.3: Average F1-scores of our system and seven state-of-the-art systems on the CK+ data set. The F1 scores for all methods were obtained from the literature. Note that the averages are not over the same AUs, and cannot readily be compared. The best performance for each condition is boldfaced.

AU	HULM	[40]	[41]	[42]	[43]	[44]	[45]
1	**0.91**	0.87	0.83	0.66	0.78	0.76	0.88
2	0.85	0.90	0.83	0.57	0.80	0.76	**0.92**
4	0.76	0.73	0.63	0.71	0.77	0.79	**0.89**
5	0.63	**0.80**	0.60	–	0.64	–	–
6	0.69	0.80	0.80	**0.94**	0.77	0.70	0.93
7	0.57	0.47	0.29	**0.87**	0.62	0.63	–
12	0.88	0.84	0.84	0.88	**0.90**	0.87	**0.90**
15	0.72	0.70	0.36	**0.84**	0.70	0.71	0.73
17	**0.89**	0.76	–	0.79	0.81	0.86	0.76
25	**0.96**	**0.96**	0.75	–	0.88	–	0.73
Avg.	0.79	0.78	0.66	0.78	0.77	0.76	0.84

In Table 2.3, we compare the performance of our AU detection system with that of seven other state-of-the-art systems in terms of the more commonly used F1-score. (Please note that the averages are not over the same AUs, and cannot readily be compared.) The results in the table show that our system achieves the best F1 scores for AU1, AU17, and AU25. It performs very strongly on most of the other AUs, illustrating the potential of the hidden-unit logistic model. Note that the state-of-the-art methods used in this comparison have specifically designed and optimized for AU detection task, while our approach is a direct application of the proposed hidden-unit logistic model.

Detailed performance analysis of the proposed hidden-unit logistic model (HULM), using combined features, is given in Table 2.4, where accuracy (ACC), recall (RC), precision (PR), F1, AUC measures, and number of positive samples are given for each AU.

2.6. CONCLUSIONS

In this chapter, we presented the hidden-unit logistic model (HULM), a new model for the single-label classification of time series. The model is similar in structure to the popular hidden CRF model, but it employs binary stochastic hidden units instead of multinomial hidden units between the data and label. As a result, the HULM can model exponentially more latent states than a hidden CRF with the same number of parameters. The results of our experiments with HULM on several real-world datasets show that this may result in improved performance on challenging time-series classification tasks. In

Table 2.4: Performance of HULM for different AUs using combined features. P shows the number of positive samples. ACC, RC, and denote detection accuracy, recall, and precision, respectively.

AU	P	ACC	RC	PR	F1	AUC
1	175	0.95	0.88	0.93	0.91	0.96
2	117	0.94	0.84	0.86	0.85	0.96
4	194	0.86	0.71	0.83	0.76	0.90
5	102	0.88	0.62	0.64	0.63	0.88
6	123	0.88	0.63	0.77	0.69	0.92
7	121	0.82	0.58	0.56	0.57	0.81
12	131	0.95	0.88	0.89	0.88	0.95
15	95	0.91	0.75	0.70	0.72	0.92
17	203	0.92	0.91	0.87	0.89	0.97
25	324	0.95	0.95	0.97	0.96	0.97
Avg.	-	0.91	0.77	0.80	0.79	0.93

particular, the HULM performs very competitively on complex computer-vision problems such as facial expression recognition.

In future work, we aim to explore more complex variants of our hidden-unit logistic model. In particular, we intend to study variants of the model in which the simple first-order Markov chains on the hidden units are replaced by more powerful, higher-order temporal connections. Specifically, we intend to implement the higher-order chains via a similar factorization as used in neural autoregressive distribution estimators [46]. The resulting models will likely have longer temporal memory than our current model, which will likely lead to stronger performance on complex time series classification tasks. A second direction for future work we intend to explore is an extension of our model to multi-task learning. Specifically, we will explore multi-task learning scenarios in which sequence labeling and time series classification is performed simultaneously (for instance, simultaneous recognition of short-term actions and long-term activities, or simultaneous optical character recognition and word classification). By performing sequence labeling and time series classification based on the same latent features, the performance on both tasks may be improved because information is shared in the latent features.

3 TEMPORAL ATTENTION-GATED MODEL FOR ROBUST SEQUENCE CLASSIFICATION

Typical techniques for sequence classification are designed for well-segmented sequences which have been edited to remove noisy or irrelevant parts. Therefore, such methods cannot be easily applied on noisy sequences expected in real-world applications. In this chapter, we present the Temporal Attention-Gated Model (TAGM) which integrates ideas from attention models and gated recurrent networks to better deal with noisy or unsegmented sequences. Specifically, we extend the concept of attention model to measure the relevance of each observation (time step) of a sequence. We then use a novel gated recurrent network to learn the hidden representation for the final prediction. An important advantage of our approach is interpretability since the temporal attention weights provide a meaningful value for the salience of each time step in the sequence. We demonstrate the merits of our TAGM approach, both for prediction accuracy and interpretability, on three different tasks: spoken digit recognition, text-based sentiment analysis and visual event recognition.

This chapter is published as:
Wenjie Pei, Tadas Baltrusaitis, David M.J. Tax and Louis-Philippe Morency. Temporal Attention-Gated Model for Robust Sequencce Classification. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

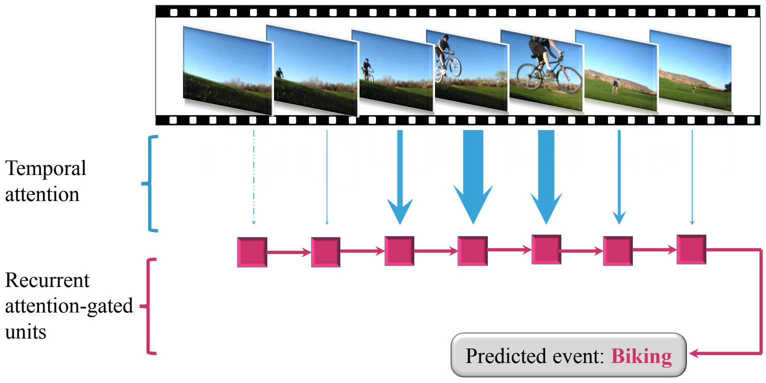


Figure 3.1: Our proposed model first employs an attention module to extract the salient frames from the noisy raw input sequences, and then learns an effective hidden representation for the top classifier. The wider the arrow is, the more the information is incorporated into the hidden representation. The dashed line represents no transfer of information.

3.1. INTRODUCTION

Sequence classification is posed as a problem of assigning a label to a sequence of observations. Sequence classification models have extensive applications ranging from computer vision [47] to natural language processing [48]. Most existing sequence classification models are designed for well segmented sequences and do not explicitly model the fact that irrelevant (noisy) parts may be present in the sequence. To reduce the interference of these irrelevant parts, researchers will often manually pre-process the dataset to remove irrelevant subsequences. This manual pre-processing can be very time consuming and reduce applicability in real-world scenarios.

A popular approach for sequence classification is gated recurrent networks like Gated Recurrent Units (GRU) [7] and Long Short-Term Memory (LSTM) [8]. They employ gates (e.g., the input gate in the LSTM model) to balance between current and previous time steps when memorizing the temporal information flow. However, these vectorial gates are applied individually to each dimension of the information flow, thus it is hard to interpret the relative importance of the input time observations (i.e., time steps). What subset of sequential observations is the most salient for the classification task? Another way to balance the information flow, as we do in this work, is the adoption of attention-based mechanism, which applies individual attention scores to each observation (time step), allowing for better interpretability.

In this section, we introduce the Temporal Attention-Gated Model (TAGM) which extends the idea of attention-based mechanism to sequence classification tasks (see overview in Figure 3.1). TAGM's attention module automatically localizes the salient observations which are relevant to the final decision and ignore the irrelavant (noisy) parts of the input sequence. We created a new recurrent neural unit that can learn a better sequence hidden representation based on the attention scores. Consequently, TAGM's classification decision is made based on the selected relevant segments, improving accuracy over the conventional models that take into account the whole input sequence.

Notably, compared to conventional sequence classification models, TAGM benefits from the following advantages:

- It is able to automatically capture salient parts of the input sequences thereby leading to better performance.
- The inferred attention (scalar) scores provide a meaningful interpretation for the informativeness of each observation in the sequence.
- Compared to conventional gated recurrent models such as LSTM, our model reduces the number of parameters which leads to faster training and inference and better generalizability with less training data.
- The proposed model is able to generalize to tasks in computer vision, speech recognition, and natural language processing.

3.2. RELATED WORK

While a full review of previous sequence classification models is beyond the scope of this chapter, in this section we summarize approaches most relevant to our proposed approach, grouping them in three areas: sequence classification, attention models and recurrent networks.

Sequence Classification. The conventional sequence classification models can be divided roughly into two categories: generative and discriminative models.

The first category focuses on learning an effective intermediate representation based on generative models. These methods are typically based on the Hidden Markov Models (HMMs) [2]. The HMM is a generative model which can be extended to class-conditional HMMs for sequence classification by combining class priors via Bayes' rule. HMM can also be used as the base model for Fisher Kernel [17] to learn a sequence representation.

The second category is the discriminative graphical models which model the distribution over all class labels conditioned on the input data. Conditional random fields (CRF) [3] are discriminative models for sequence labeling which aims to assign one label for each sequence observation. A potential drawback of common CRFs is that the linear mapping between observations and labels cannot model complex decision boundaries, which gives rise to many non-linear CRF-variants (e.g., latent-dynamic CRFs [20], conditional neural fields [21], neural conditional random fields [22] and hidden-unit CRF model [15]). Hidden-state CRF (HCRF) [4] employs a chain of k-nomial latent variables to model the latent structure and has been successfully used in the sequence labeling. Similarly, hidden unit logistic model (HULM) [49] utilizes binary stochastic hidden units to represent the exponential hidden states so as to model more complex latent decision boundaries.

Aforementioned works are specifically designed for well segmented sequences and hence cannot cope well with noisy or unsegmented sequences.

Attention Models. Inspired by the attention scheme of human foveal vision, attention model was proposed to focus selectively on certain relevant parts of the input by measuring the sensitivity of output to variances of the input. Doing so can not only improve the performance of the model but can also result in better interpretability [50]. Attention models have been applied to image and video captioning [50–53], machine translation [48, 54, 55], depth-based person identification [56] and speech recognition [57]. To the best of our knowledge, our TAGM is the first end-to-end recurrent neural network to employ the attention mechanism in the temporal domain of sequences, with the added advantage of interpretability of its temporal salience indicators (i.e., temporal attention) at each time step (sequence observation). Our work is different from prior work focused on spatial domain (e.g., images) such as the model proposed by Sharma et al. [58].

Recurrent Networks. Recurrent Neural Networks (RNN) learn a representation for each time step by taking into account both the observation at current time step and the representation in the previous one [59]. The biggest advantage of recurrent neural networks lies in their capability of preserving information over time by the recurrent mechanism. Recurrent networks have been successfully applied to various tasks including language modeling [60], image generation [61] and online handwriting generation [62]. To address the gradient vanishing problem of plain-RNN when dealing with long sequences, LSTM [8] and GRU [7] were proposed. They are equipped with the gates to balance the information flow from the previous time step and current time step dynamically. Inspired by this setup, our TAGM model also employs a gate to filter out the noisy time steps and preserve the salient ones. The difference from the LSTM and GRU is that the gate value in our model is fed from the attention module which focuses on learning the salience at each time step.

3.3. TEMPORAL ATTENTION-GATED MODEL

Given as input an unsegmented sequence of possibly noisy observations, our goal is to: (1) calculate a salience score for each time step observation in our input sequence, and (2) construct a hidden representation based on the salience scores, best suited for the sequence classification task. To achieve these goals, we propose the Temporal Attention-Gated Model (TAGM) which consists of two modules: temporal attention module, and recurrent attention-gated units. Our TAGM model can be trained in an end-to-end manner efficiently. The graphical structure of the model is illustrated in Figure 4.1.

3.3.1. RECURRENT ATTENTION-GATED UNITS

The goal of the recurrent attention-gated units is to learn a hidden sequence representation which integrates the attention scores (inferred from the temporal attention module that will be discussed in the next section). In order to integrate the attention scores in the recurrent network units, we define an attention gate to control how much information is incorporated from the input of the current time step based on the salience and relevance to the final task.

Formally, given an input sequence $\mathbf{x}_{1,\dots,T} = \{\mathbf{x}_1,\dots,\mathbf{x}_T\}$ of length T in which $\mathbf{x}_t \in \mathbb{R}^D$ denotes the observation at the t-th time step, the attention score at time step t is denoted as a_t, which is a scalar value that indicates the salience of current time step to the final decision. For this purpose, we define our core recurring process where the hidden state $\mathbf{h}_t$ at time step t is modeled as a convex summation:

$$\mathbf{h}_t = (1 - a_t) \cdot \mathbf{h}_{t-1} + a_t \cdot \mathbf{h}'_t \tag{3.1}$$

Wherein, $\mathbf{h}_{t-1}$ is the previous hidden state and $\mathbf{h}'_t$ is the candidate hidden state value which fully incorporates the input information $\mathbf{x}_t$ in the current time step:

$$\mathbf{h}'_t = g(\mathbf{W} \cdot \mathbf{h}_{t-1} + \mathbf{U} \cdot \mathbf{x}_t + \mathbf{b}) \tag{3.2}$$

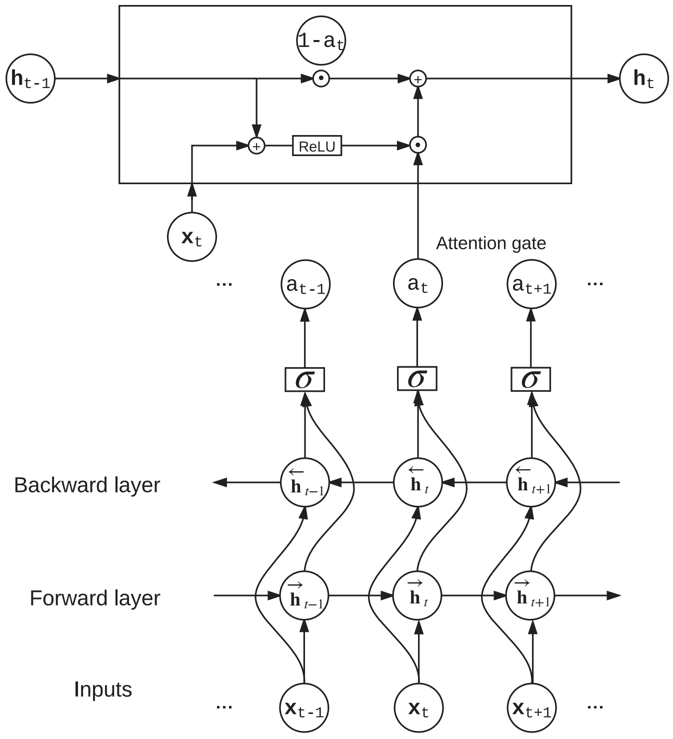


Figure 3.2: The graphical representation of our Temporal Attention-Gated Model (TAGM). The top part of the figure is the Recurrent Attention-Gated Units and the bottom is the Temporal Attention Module. Note that a_t is the saliency score represented as a scalar value instead of a vector, hence $\odot$ in the figure means multiplication between a scalar and a vector.

Herein, **W** and **U** are respectively the linear transformation parameters for previous and current time steps while **b** is the bias term. We use the rectified linear unit (ReLU)[63] as the activation function g. Equation 3.1 uses attention score a_t to balance the information flow between current candidate hidden state $\mathbf{h}'_t$ and previous hidden state $\mathbf{h}_{t-1}$. High attention value will push the model to focus more on the current hidden state $\mathbf{h}'_t$ and input feature $\mathbf{x}_t$, while low attention value would make the model ignore the current input feature and inherit more information from previous time steps.

The learned hidden representation at the last time step $\mathbf{h}_T$ of the sequence is further fed into the final classifier, often a softmax function, to perform a classification task, which calculates the probability of a predicted label y_k among K classes as:

$$P(y_k|\mathbf{h}_T) = \frac{\exp\{\mathbf{W}_k^\top \mathbf{h}_T + b_k\}}{\sum_{i=1}^K \exp\{\mathbf{W}_i^\top \mathbf{h}_T + b_i\}} \tag{3.3}$$

where $\mathbf{W}_i^\top$ and b_i refer to the parameters calculating the linear mapping score for the i-th class.

3.3.2. TEMPORAL ATTENTION MODULE

The goal of this module is to estimate the saliency and relevance of each sequence observation. This saliency score should not only be based on the input observation at the current time step, but also take into consideration information from neighboring observations in both directions. To model this neighborhood influence, we infer the attention score a_t in Equation 3.1 using a bi-directional RNN:

$$a_t = \sigma(\mathbf{m}^\top(\overrightarrow{h}_t; \overleftarrow{h}_t) + b) \tag{3.4}$$

Herein, **m** is the weight vector of our fusion layer which integrates both directional layers of our bi-directional RNN and b is the bias term. A sigmoid function is employed as the activation function σ at the top layer of the attention module in Equation 3.4 to constraint the attention weight to lie between $[0,1]$. $\overrightarrow{h}_t$ and $\overleftarrow{h}_t$ are the hidden representations of a bi-directional RNN model:

$$\overrightarrow{h}_t = g(\overrightarrow{\mathbf{W}}\mathbf{x}_t + \overrightarrow{\mathbf{U}}\overrightarrow{h}_{\mathbf{t-1}} + \overrightarrow{\mathbf{b}}) \tag{3.5}$$

$$\overleftarrow{h}_t = g(\overleftarrow{\mathbf{W}}\mathbf{x}_t + \overleftarrow{\mathbf{U}}\overleftarrow{h}_{\mathbf{t+1}} + \overleftarrow{\mathbf{b}}) \tag{3.6}$$

The ReLU functions are used as the activation functions g. Our choice of using plain bi-directional RNN model is motivated by the design goal of reducing the number of parameters in our model.

The learned attention weights a_t serve as the attention gate for Recurrent Attention-Gated Units to control the involved information flow. Furthermore, another important role the learned attention weights play is to provide an interpretability about the degree of salience of each time step.

3.3.3. END-TO-END PARAMETER LEARNING

Suppose we are given a training set $\mathscr{D} = \{(\mathbf{x}^{(n)}_{1,\dots,T}, y^{(n)})\}_{n=1,\dots,N}$ containing N sequences of length T and their associated labels $y^{(n)}$. $\mathbf{x}^{(n)}_t \in \mathbb{R}^D$ denotes the observation at the t-th time step of the n-th sample and T can differ from sequence to sequence. We learn jointly the two TAGM modules (temporal attention module and recurrent attention-gated units) and the final sequence classifier by minimizing the conditional negative log-likelihood of the training data with respect to the parameters:

$$\mathscr{L} = -\sum_{n=1}^{N} \log P\left(\mathbf{y}^{(n)} | \mathbf{x}^{(n)}_{1,\dots,T}\right) \tag{3.7}$$

Since all three modules (including the final sequence classifier) are analytically differentiable, our TAGM model can be readily trained in an end-to-end manner. The loss is back-propagated through top recurrent attention-gated units and temporal attention module successively using back-propagation through time algorithm [64].

3.3.4. COMPARISON WITH LSTM AND GRU

While our model is similar to RNN variants like GRU and LSTM, it is specifically designed with salience detection in mind and has four key differences when compared to them:

- We only focus on one scalar attention score to measure the relevance of the current time step instead of generally modeling gate's multi-dimensional values for each hidden unit as done by GRU and LSTM. In this way, we can obtain an interpretable salience detection (demonstrated on three tasks in Section 6.4).

- We separate the attention modeling and recurrent hidden representation learning as two independent modules to decrease the degree of coupling. One of the advantages of this is our ability to customize the specific recurrent structure for each module with different complexity according to the requirements (eg., different size of hidden units in two modules of TAGM in Table 3.1).
- We employ a bi-directional RNN to take into account both the preceding and the following information of the sequence in the temporal attention module. It helps to model the temporal smoothness of the sequence of salience scores (demonstrated in Figure 3.4). It should be noted that it is different from the design of the gates in the bi-directional LSTM model since the latter just concatenates the hidden representations of two unidirectional LSTMs, which does not remedy the downside that all vectorial gates are still calculated by considering only one-directional information.
- Our model only contains one scalar gate, namely the attention gate, rather than 2 vectorial gates in GRU and 3 gates in LSTM. Doing so enforces the attention gate to take full responsibility of modeling all the salience information. In addition, the model contains fewer parameters (compared to LSTM) and simpler gate structure with less redundancy (compared to GRU and LSTM). It eases the training procedure and can alleviate the potential over-fitting and has better generalization given small amount of training data, which is demonstrated in Section 3.4.1.

3.4. EXPERIMENTS

We performed experiments with TAGM on three publicly available datasets , selected to show generalization across different tasks and modalities: (1) speech recognition on an audio dataset, (2) sentiment analysis on a text dataset, and (3) event recognition on a video dataset.

Experimental setup shared across experiments. For all the recurrent networks mentioned in this work (TAGM, GRU, LSTM and plain-RNN), the number of hidden units is tuned by selecting the best configuration from the option set $\{64, 128, 256\}$ using a validation set. The dropout value is validated from the option set $\{0.0, 0.25, 0.5\}$ to avoid potential overfitting. We employ RMSprop as the gradient descent optimization algorithm with gradient clipping between -5 and 5 [65].

We validate the learning rate for parameters $\mathbf{m}$ and b in Equation 3.4 to make the effective region of the sigmoid function of TAGM model adaptive to the specific data. Larger learning rate leads to sharper distribution of attention weights. Code reproducing the results of our experiments is available [1].

3.4.1. SPEECH RECOGNITION EXPERIMENTS

We first conduct preliminary experiments on a modified dataset constructed from the Arabic spoken digit dataset [31] to (1) evaluate the effectiveness of the two main modules of TAGM; (2) compare the generalizability of three different gate-setup recurrent models

[1]https://github.com/wenjiepei/TAGM

(TAGM, GRU and LSTM) with the varying size of the training data.

DATASET

The Arabic spoken digit dataset contains 8800 utterances, which were collected by asking 88 Arabic native speakers to utter all 10 digits ten times. Each sequence consists of 13-dimensional Mel Frequency Cepstral Coefficents (MFCCs) which were sampled at 11,025Hz, 16-bits using a Hamming window. We append white noise to the beginning and the end of each sample to simulate the problem with unsegmented sequences. The length of the unrelated sub-sequences before and after the original audio clips is randomized to ensure that the model does not learn to just focus on the middle of the sequence.

EXPERIMENTAL SETUP

We use the same data division as Hammami and Bedda [31]: 6600 samples as training set and 2200 samples as test set. We further set aside 1100 samples from training set as the validation set. There is no subject overlap in the three sets.

We compare the performance of our TAGM with three types of baseline models:
Attention Module + Neural Network (AM-NN). To study the impact of our recurrent attention-gated unit, we include a baseline model which employs a feed-forward network directly on top of the temporal attention module. In this AM-NN model, **v** is defined as the weighted sum of input features:

$$\mathbf{v} = \sum_{t=1}^{T} a_t \cdot \mathbf{x}_t, \quad \mathbf{h} = g(\mathbf{W} \cdot \mathbf{v} + \mathbf{b}) \tag{3.8}$$

Sequence classification is performed by passing **h** into a softmax layer, as done for our TAGM (see Equation 3.3).
Discriminative Graphical Models. HCRF and HULM are both extensions of CRF [3] by inserting hidden layers to model the non-linear latent structure in the data. The difference lies in the structure of hidden layers: HCRF uses a chain of k-nomial latent variables while HULM utilizes k binary stochastic hidden units.
Recurrent neural networks. Since our model is a recurrent network equipped with a gate mechanism, we compare it with other recurrent networks: plain-RNN, GRU, LSTM. We also investigate the bi-directional variant of our TAGM model (referred as Bi-TAGM), which employs the bi-directional recurrent configuration in the recurrent attention-gated units.

In our experiments, we also evaluate the generalizability when varying size of training data: from 1,100 to 5,500 training samples. During these experiments, the optimal configuration is selected automatically during validation from the option set {64,128,256}.

RESULTS AND DISCUSSION

Evaluation of Classification Performance Table 3.1 presents the classification performance of several sequence classifiers on Arabic dataset. In order to investigate the effect of the manually added noise information, we perform experiments on both clean and noisy versions of data.

Table 3.1: Classification accuracy (%) on Arabic spoken digit dataset by different sequence classification models. Asterisked models (∗) are trained and evaluated on the clean version of data. Note that we can customize separately the complexity of TAGM's two modules. This design advantage is shown when looking at the optimal TAGM model (after validation) which has 128 dimensions for the Temporal Attention Module, and 64 dimensions for the Recurrent Attention-Gated Units.

Model	#Hidden units	#Parameters	Accuracy
HULM* [49]	–	–	95.32
HCRF* [49]	–	–	96.32
HULM	–	–	88.27
HCRF	–	–	90.41
Plain-RNN*	256	75 K	94.95
Plain-RNN	256	75 K	10.95
GRU	128	61 K	97.05
LSTM	128	81 K	95.91
NN	64	2.4 K	65.50
AM-NN	128-64	43 K	85.59
TAGM	128-64	47 K	**97.64**
Bi-GRU	64	37 K	97.68
Bi-LSTM	256	587 K	97.45
Bi-TAGM	128-128	83 K	**97.91**

While the Plain-RNN is unable to recognize spoken digits in a noisy setting, other three recurrent models with gate-setup do not suffer from the noise and obtain comparable performance with the result achieved by HCRF on clean data. Our model achieves the best results among all classifiers with single-directional recurrent configuration. This probably results from better generalization of our model on the relatively small dataset due to the simpler gate setup and also the attention mechanism. We also perform experiments with the bi-directional version of GRU, LSTM and TAGM, in which our Bi-TAGM performs best. Bi-GRU achieves its best performance with 64 hidden units. It is worth mentioning that our (single-directional) TAGM using 47 K parameters already achieves comparable result with the Bi-LSTM and Bi-GRU, which indicates that the bi-directional mechanism in the attention module of TAGM enables it to capture most bi-directional information in the attention layer alone.

Comparison of generalizability with the varying size of training data. We first conduct experiments to compare the generalizability of TAGM to GRU and LSTM by varying the size of training data on the noisy Arabic dataset. Figure 3.3 presents the experimental results. It can be seen that TAGM exhibits better generalizability than GRU and LSTM on smaller training data sizes, which we believe is caused by the need to learn fewer model parameters, avoiding overfitting.

Sequence Salience Detection. In order to evaluate the performance of sequence salience detection by our TAGM model, we visualize the attention weights of our model trained on the noisy Arabic dataset, which is illustrated in Figure 3.4.a. It shows that the atten-

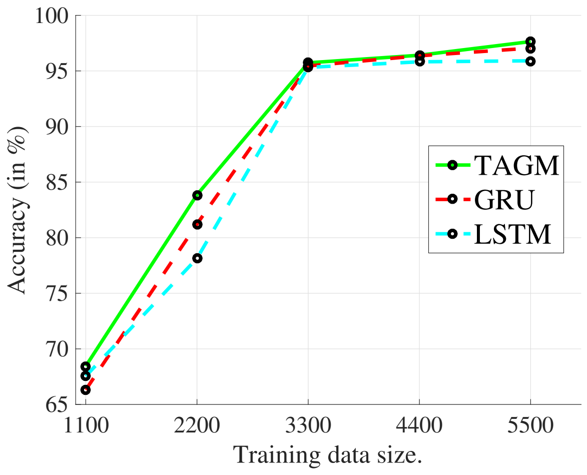


Figure 3.3: The classification accuracy on the noisy Arabic speech dataset as a function of the size of training data. Note that our TAGM model outperforms GRU and LSTM when less training data is available.

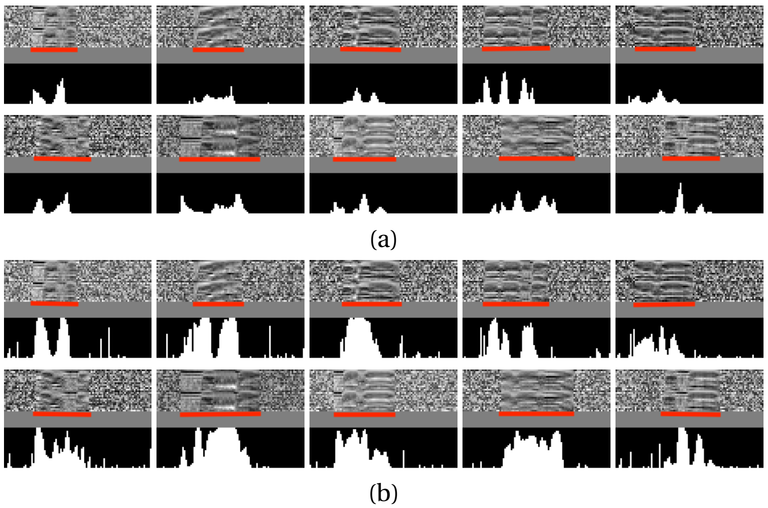


Figure 3.4: The visualization of attention weights of TAGM in Figure a and Attention module+NN in Figure b (the weighted features are fed into Feed-forward Neural Networks) on 10 samples (one sample for each digit). For each subfigure, the top subplot shows the spectrogram of the original sequence data, the bottom subplot shows the attention values a_t over time. The red lines indicate the ground-truth of salient segments. Note that TAGM attention weights result in a cleaner attention representation.

tion model can correctly detect the informative section of the raw signal.

To investigate the effect of the temporal information contained in the hidden representation, we also visualize the attention weight of the Attention module + Neural Network classifier, which is shown in Figure 3.4.b. It shows that the TAGM results in a cleaner and smoother attention weight profile, also notice the spiky behavior, which is mainly

Table 3.2: Classification accuracy (%) on Stanford Sentiment TreeBank dataset when training with only the sentence-level labels. We conduct experiments on both binary and fine-grained (5-class) classification tasks. Note that our model outperforms all others in the task.

	Model	Binary	Fine-grained
Graphical models	HULM	81.3	44.1
	HCRF	84.8	45.3
Syntactic compositions	DAN-ROOT [69]	85.7	46.9
Recurrent models	Plain-RNN	83.9	42.3
	GRU	85.4	46.7
	LSTM	85.9	47.2
Our model	TAGM	**86.2**	**48.0**

achieved by the bi-directional RNN in our temporal attention module.

3.4.2. SENTIMENT ANALYSIS EXPERIMENTS

Sentiment analysis is a popular research topic in the field of natural language processing (NLP) which aims to identify the viewpoint(s) underlying a text span [66]. We conduct experiments for sentiment analysis to evaluate the performance of our TAGM model on the text modality.

DATASET

The Stanford Sentiment Treebank (SST) [67] is a data corpus of movie review excerpts. It consists of 11,855 sentences each of which is assigned a score to indicate the sentimental attitude towards the movie reviews. The dataset offers two types of annotations, sentiment annotations at the sentence level (with a total of 11,855 sentences) and at the phrase level (with a total of 215,154 phrases). The sentence-level and phrase-level labels are provided with two resolutions: binary-classification task (positive or negative) and fine-grained task (5-level classes).

EXPERIMENTAL SETUP

Following previous work [67], we utilize 300-d *Glove* word vectors (300 dimensions) pretrained over the Common Crawl [68] as the features for each word of the sentences. Our model is well suited to perform sentiment analysis using sentence-level labels. Nevertheless, we also perform experiments with phrase-level labels so as to have a fair and intuitive comparison with state-of-the-art baselines.

We follow the same data split as described by Socher et al. [67]: 8544/1101/2210 samples are used for training, validation and testing respectively in the 5-class task. The corresponding splits in the binary classification task are 6920/872/1821.

RESULTS AND DISCUSSION

Evaluation of Classification Performance We conduct two sets of experiments to evaluate the performance of our model in comparison with the baseline models. Since our

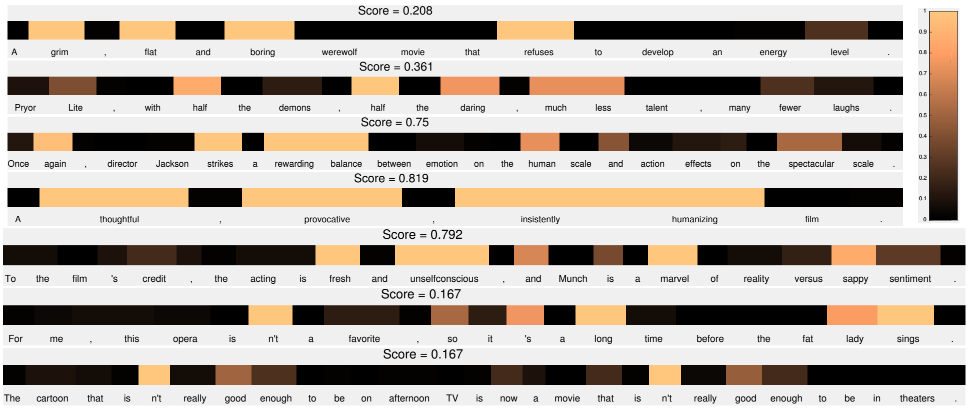
Table 3.3: Classification accuracy (%) on Stanford Sentiment TreeBank dataset when training with both phrase-level and sentence-level labels. Our TAGM achieves the best overall result.

	Model	Binary	Fine-grained	Overall Performance
Unordered compositions	NBOW-RAND [69]	81.4	42.3	123.7
	NBOW [69]	83.6	43.6	127.2
	BiNB [69]	83.1	41.9	125.0
Syntactic compositions	RecNN [70]	82.4	43.2	125.6
	RecNTN [67]	85.4	45.7	131.1
	DRecNN [71]	86.6	49.8	136.4
	DAN [69]	86.3	47.7	134.0
	TreeLSTM [72]	86.9	**50.6**	137.5
	CNN-MC [73]	**88.1**	47.4	135.5
	PVEC [74]	87.8	48.7	136.5
Our model	TAGM	87.6	50.1	**137.7**

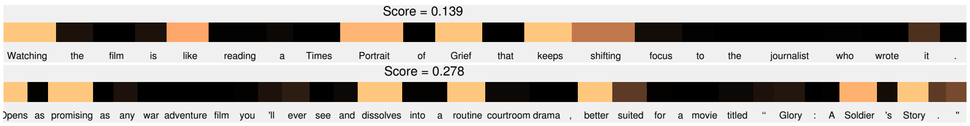
model is designed for unsegmented and possibly noisy sequences modeling, it is more suitable to only use sentence-level labels, although phrase-level labels are also provided in SST dataset. Table 3.2 shows the experimental results of several sequential models trained with only sentence-level labels. Our model achieves the best result in both binary classification task and fine-grained (5-class) task. LSTM and GRU outperform plain-RNN model due to the information-filtering capability performed by additional gates. It is worth mentioning that our model achieves better performance than LSTM with only half the hidden parameters.

To have a fair comparison with the existing sentiment analysis models, we conduct the second set of experiments with both sentence-level and phrase-level labels. The results are presented in Table 3.3. It shows that our model outperforms most of the existing models and achieves comparable accuracy with the state-of-the-art results. Our TAGM model actually obtains overall best results considering both binary and fine-grained cases. This is an encouraging result, in particular, since our model is not specifically designed towards NLP tasks.

Sequence Salience Detection In order to investigate the performance of salience detection by our TAGM model on Sentiment dataset (SST), we visualize the calculated attention weights for each word in the test sentences. Group (a) in Figure 3.5 presents a number of examples that are predicted correctly by our model in the binary-classification task. It shows that our model is able to successfully capture the key sentimental words and omit irrelevant words, even for the sentences with complicated syntax. We also test the examples that include negated expressions. As shown in the last two sentences of group (a), our model can deal with them very well. We also investigate the samples our model fails to predict the correct sentiment label (see Figure 3.5b).



(a) Correct predictions.



(b) Wrong predictions.

Figure 3.5: The visualization of attention weights of Recurrent Attention Model: (a) correct predictions and (b) wrong predictions. The scores displayed are the groundtruth label indicating the writer's overall sentiment for this review. Darker color indicates smaller scores.

3.4.3. EVENT RECOGNITION EXPERIMENTS

We subsequently conduct experiments for video event recognition to evaluate our model on the visual modality.

DATASET

Columbia Consumer Video (CCV) Database [75] is an unconstrained video database collected from YouTube videos without any post-editing. It consists of 9317 web videos with average duration of 80 seconds (210 hours in total). Except for some negative background videos, each video is manually annotated into one or more of 20 semantic categories such as 'basketball', 'ice skating', 'biking', 'birthday' and so on. It is a very challenging database due to the many noisy and irrelevant segments contained inside these videos.

EXPERIMENTAL SETUP

Following Jiang et al [75], we use the same split for training and test sets: 4659 videos as the training set and 4658 as the test set. We compare our model with the baseline method [76] on this dataset, which performs classification separately with Support Vector Machine (SVM) models trained on the bag-of-words representations for several popular features separately and then combines the results using late fusion. Its experimental results show that Convolutional Neural networks (CNNs) features perform best among all features they tried, hence we choose to use CNN features with the same setup, i.e., the outputs (4,096 dimensions) of the seventh fully-connected layer of a pre-trained AlexNet

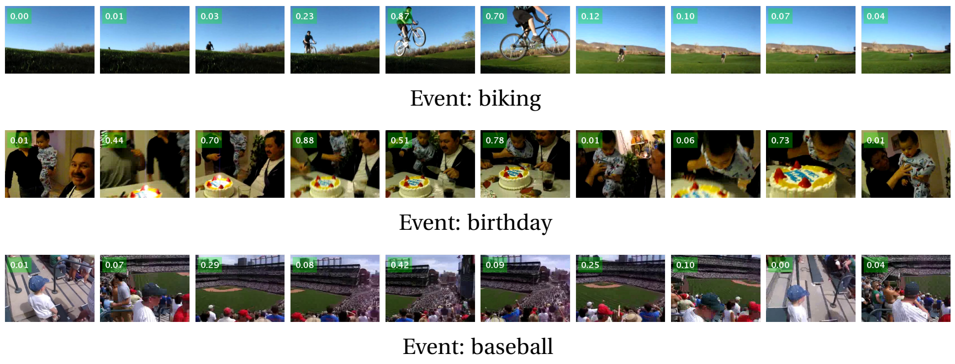


Figure 3.6: The calculated attention weights of our TAGM model for examples from test set of CCV database. The attention weight is indicated for representative frames. Our TAGM is able to capture the action of 'riding bike' for the event 'biking', 'cake' for the event 'birthday' and 'infield zone' for 'baseball'. A video containing these three complete sample sequences is presented in the supplementary material.

model [77]. For the sake of computational efficiency, we extract CNN features with a sampling rate 1/8 (one out of every eight frame).
We adopt mean Average Precision (mAP) as the evaluation metric, which is typically used for CCV dataset [75, 76]. Since more than one event (correct label) can happen in a sample, we perform binary classification for each category but train them jointly, hence the prediction score for each category is calculated by a sigmoid function instead of softmax Equation 3.3:

$$P(y_k = 1|\mathbf{h}_T) = \frac{1}{1+\exp\{-(\mathbf{W}_k^\top \mathbf{h}_T + b_k)\}} \tag{3.9}$$

and joint binary cross-entropy over K categories is minimized:

$$\mathcal{L} = -\sum_{n=1}^{N}\sum_{k=1}^{K}\Big[\log P(y_k = 1|\mathbf{h}_T) + \log(1 - P(y_k = 0|\mathbf{h}_T))\Big]$$

Results and Discussion

Evaluation of Classification Performance. We compare our model with the event recognition system proposed by dataset authors [76]. Table 3.4 presents the performance of several models for event recognition, in which our TAGM outperforms the other recurrent models by a large margin. The baseline BOW+SVM employs the one-vs-all strategy to train a separate classifier for each event while our model trains all events jointly in a single classifier. Our model still shows encouraging results since it is quite a challenging task for TAGM to capture salient sections for 20 events with complex scenes simultaneously. Moreover, our TAGM can provide a meaningful interpretation which the baseline models cannot do.

Sequence Salience Detection. Salience detection for CCV database is a difficult but appealing task due to complex and long scenes in videos. Figure 3.6 shows some exam-

Table 3.4: Mean Average Precision (mAP) of our TAGM model and baseline models on CCV dataset.

Model	Training strategy	Feature	mAP
BOW+SVM +late average fusion	Separately (one-vs-all)	SIFT	0.52
		STIP	0.45
		SIFT+STIP	0.55
		CNN	0.67
Plain-RNN	Jointly	CNN	0.45
GRU	Jointly	CNN	0.56
LSTM	Jointly	CNN	0.55
TAGM	Jointly	CNN	0.63

ples where TAGM correctly locates the salient subsequences by the attention weights. Our model is able to capture the relevant action, object and scene to the event, e.g., the action of riding bike for the event ‘biking’, cake for the event ‘birthday’ and baseball playground for the event ‘baseball’. It is interesting to note that the frame with the score 0.42 in event ‘baseball’ achieves the high score probably because of the real-time screen in the top right corner.

3.5. CONCLUSION

In this work, we presented the Temporal Attention-Gated Model (TAGM), a new model for classifying noisy and unsegmented sequences. The model is inspired by attention models and gated recurrent networks and is able to detect salient parts of the sequence while ignoring irrelevant and noisy ones. The resulting hidden representation suffers less from the effect of noise and and thus leads to better performance. Furthermore, the learned attention scores provide a physically meaningful interpretation of relevance of each time step observation for the final decision. We showed the generalization of our approach on three very different datasets and sequence classification tasks. As future work, our model could be extended to help with document or video summarization.

4

ATTENDED END-TO-END ARCHITECTURE FOR AGE ESTIMATION FROM FACIAL EXPRESSION VIDEOS

The main challenges of age estimation from facial expression videos lie not only in the modeling of the static facial appearance, but also in the capturing of the temporal facial dynamics. Traditional techniques to this problem focus on constructing handcrafted features to explore the discriminative information contained in facial appearance and dynamics separately. This relies on sophisticated feature-refinement and framework-design. In this chapter, we present an end-to-end architecture for age estimation which is able to simultaneously learn both the appearance and dynamics of age from raw videos of facial expressions. Specifically, we employ convolutional neural networks to extract effective latent appearance representations and feed them into recurrent networks to model the temporal dynamics. More importantly, we propose to leverage attention models for salience detection in both the spatial domain for each single image and the temporal domain for the whole video as well. We design a specific spatially-indexed attention mechanism among the convolutional layers to extract the salient facial regions in each individual image, and a temporal attention layer to assign attention weights to each frame. This two-pronged approach not only improves the performance by allowing the model to focus on informative frames and facial areas, but it also offers an interpretable correspondence between the spatial facial regions as well as temporal frames, and the task of age estimation. We demonstrate the strong performance of our model in experiments on a large, gender-balanced database with 400 subjects with ages spanning from 8 to 76 years.

This chapter is submitted for publication as:
Wenjie Pei, Hamdi Dibeklioğlu, Tadas Baltrušaitis and David M.J. Tax. Attended End-to-End Architecture for Age Estimation from Facial Expression Videos. IEEE Transactions on Image Processing.

Experiments reveal that our model exhibits significant superiority over the state-of-the-art methods given sufficient training data.

4.1. INTRODUCTION

Human age estimation from faces is an important research topic due to its extensive applications ranging from surveillance monitoring [78, 79] to forensic art [80, 81] and social networks [82, 83]. The widely-used discriminative features for age estimation are appearance-related, such as wrinkles in the face, skin texture and luster, hence plenty of prevalent methods focus on modeling the appearance information from the static face [84, 85]. Recent studies [86, 87] also indicate that the dynamic information in facial expressions like temporal properties of a smile can be leveraged to significantly improve the performance of age estimation. It is reasonable since there are intuitive temporal dynamics involved in facial expressions which are relevant to the age. For instance, the exhibited facial movement like wrinkles in the smiling process is different for people of different ages.

The traditional approaches to age estimation from facial expression videos focus on constructing handcrafted features to explore the discriminative information contained in static appearance and temporal dynamics separately, and then combining them into an integrated system [86, 87]. However, these kinds of methods rely on sophisticated feature design. In this work, we propose a novel end-to-end architecture for age estimation from facial expression videos, which is able to automatically learn the static appearance in each single image and the temporal dynamics contained in the facial expression simultaneously. In particular, we employ convolutional neural networks (CNNs) to model the static appearance due to its successful representation learning in the image domain. The learned latent appearance features for each image are subsequently fed into recurrent networks to model the temporal dynamics. In this way, both static appearance and temporal dynamics can be integrated seamlessly in an end-to-end manner. A key benefit of this design is that the learned static appearance features and temporal dynamic features are optimized jointly, which can lead to better performance for the final task of age estimation. Additionally, the end-to-end manner of our method avoids separating feature design from the age regression as a two-stage procedure, which is the typical way of the methods using handcrafted features.

Attention models have been proposed to let models learn by themselves to pay attention to specific regions in an image [50] or different segments in a sequence [48, 88] according to the relevance to the aimed task. Likewise, different facial parts (in each single image) and different phases of the expression (in the inspected video) may exhibit varying degrees of salience to age estimation. To incorporate attention, we customize a specific attention module for spatial facial salience detection. To detect temporal salience in the sequential expression, we mount a temporal attention module upon the recurrent networks. It serves as a filtering layer to determine the amount of information of each frame to be incorporated into final age regression task. All functional modules including convolutional networks for learning appearance, recurrent networks for learning dynamics, two attention modules as well as the final age regression module can be trained jointly, without any manual intervention.

Extensive experiments on a real-world database demonstrate the substantial superiority of our model over the state-of-the-art methods. Our model has the capacity to learn and it could do even better on more data while other models potentially saturate and do not get better no matter how much data you give them. Notably, larger training data tends to explore more potential of our model and expand its advantages compared to other methods.

4.2. Related Work

In this study, we propose to effectively learn spatial and temporal patterns of aging in an attended end-to-end manner for a more reliable age estimation. To comprehend the related concepts, in this section, the literature on automatic age estimation will be summarized, and an overview of neural attention models will be given.

4.2.1. Automatic Age Estimation

Based on the fact that age information is crucial to understand requirements or preferences of people, automatic estimation of age from face images has quite a few real-life applications, and thus, it has been a very active area of research over the last two decades. Biometric search/authentication in facial image databases, denying purchase of unhealthful products (e.g. alcohol and tobacco) by underage customers, and personalization/adaptation of interactive systems to the users (displaying personalized advertisements) can be counted as some examples of the use of age estimation.

One of the major requirements in facial age estimation is the capturing/modeling of facial properties that change by age. These patterns are related to craniofacial development [89] and alteration in facial skin (e.g. wrinkles) [90]. While the majority of earlier studies in the area focus on describing such patterns using engineered (handcrafted) representations such as local binary patterns (LBP) [91] and its variations [92], Gabor filter based biologically-inspired aging features (BIF) [93], and shape-based features [94], some studies propose to capture aging patterns through learning-based approaches such as subspace learning [78, 95–98], PCA-tree-based encoding [99], and metric learning [100, 101].

The increase in the number and size of facial age databases, and the recent dramatic improvements in the field of deep learning have shifted the focus towards deep architectures to learn complex (nonlinear) patterns of facial aging from a large collection of face images. For instance, [102] presents the first exploration of employing CNNs for age estimation, where representations obtained from different layers of CNN are used. In [103], a multi-task CNN architecture is proposed to optimize the facial representation jointly for age and gender estimation. [104] models the features extracted from a pre-rained CNN (VGG-Face [105]) using the kernel extreme learning machines. [106] uses VGG-16 CNN architecture [107] (pre-trained on ImageNet images) and fine-tunes the model using a multi-class classifier for age estimation. Then, softmax-normalized output probabilities are used for the final prediction. Differently from conventional methods, [106] solely employs face detection for face alignment rather than using facial landmark detection, leading to a more accurate age estimation. In [108], Agustsson *et al.* complement [106] with their proposed Anchored Regression Network (rather than employing softmax classifier on top the VGG-16 CNN architecture), enhancing the reliability. In a recent study [109], Xing *et al.* have analyzed different loss functions and CNN architectures for age estimation as well as employing a joint optimization together with race and gender classification tasks.

In contrast to regression and multi-class classification, some studies approach age estimation as an ordinal ranking problem. For instance, [110] presents a deep (category-based) ranking model that combines deep scattering transform and ordinal ranking. [111] formulates the problem as ordinal regression using a series of binary classification tasks which are jointly optimized by a multiple output CNN architecture. In [112], instead of a multiple output model, a series of basic CNNs are employed (a separate CNN for each ordinal age group), and their binary outputs are aggregated. Such an approach allows capturing different patterns for different age groups.

The use of deep architectures has significantly improved the reliability of automatic age estimation, especially under pose and illumination variations. Facial expression variation, however, is still a challenge since expressions form deformations on the facial surface that can be confused with aging-related wrinkles. Yet, only a few recent works in the literature explore solutions for this issue [113–116]. Guo *et al.* [113] model correlation between aging features of the neutral face and a specific expression (i.e. smile) of individuals. Learned correlation is used to map the features of expressive faces to those of neutral ones. In this way, the confusing influence of expressions in aging patterns are removed. However, this method requires an accurate estimation of facial expressions (before the age estimation), and a separate training for each expression of interest using neutral and expressive face images of each subject in the training dataset. [114] learns a common subspace for a set of facial expressions that reduce the influence of expressions while preserving the aging patterns. [115] defines four age groups, and models each facial expression in each age group as a different class for cross-expression age estimation. In a similar manner, [116] proposes a multi-task framework that jointly learns the age and expression using the latent structured support vector machines.

Interestingly, until a recent work of Hadid [117], none of the methods in the literature have used facial videos for age estimation. In [117], the volume local binary patterns (VLBP) features are employed to describe spatio-temporal information in videos of talking faces for age grouping (child, youth, adult, middle-age, and elderly). Yet, this video-based method ([117]) could not perform as accurate as the image-based methods. On the other method, more recently, Dibeklioğlu *et al.* have presented the first successful example of video-based age estimation [86], where displacement characteristics of facial landmarks are represented by a set of handcrafted descriptors extracted from smile videos, and combined with spatio-temporal appearance features. In a follow-up study, Dibeklioğlu *et al.* [87] have enhanced their engineered features so as to capture temporal changes in 3D surface deformations, leading to a more reliable age estimation. It is important to note that these two studies exploit the aging characteristics hidden in temporal dynamics of facial expressions, rather than reducing the influence of expressions in aging features. Thus, following the informativeness of temporal dynamics of facial expressions and based on the success of deep models, the current study proposes a deep temporal architecture for automatic age estimation.

Because the literature on automatic age estimation is extensive, for further information, we refer the reader to [118], [81], and to the more recent [119].

4.2.2. ATTENTION MODELS

Much of progress in neural networks was enabled by so called neural attention, which allows the network to focus on certain elements of a sequence [48, 53, 88] or certain regions of an image [50] when performing a prediction. The appeal of such models comes from their end-to-end nature, allowing the network to learn how to attend to or align data before making a prediction.

It has been particularly popular in encoder-decoder frameworks, where it was first introduced to better translate between languages [48]. The attention network learned to focus on particular words or phrases when translating sentences, showing large performance gains on especially long sequences. It has also been extensively used for visual image and video captioning, allowing the decoder module to focus on parts of the image it was describing [50]. Similarly, the neural attention models have been used in visual question answering tasks, helping the alignment between words in the question and regions in the image [120]. Spatial Transformer Networks which focus on a particular area of image can also be seen as a special case of attention [121]. Somewhat relatedly, work in facial expression analysis has explored using particular regions for facial action unit detection [122, 123], however, they did not explore dynamically attending to regions depending on the current facial appearance. Our work is inspired by these attention models, but explores different ways of constructing neural attention and applying it to age estimation.

4.3. METHOD

Given a video displaying the facial expression of a subject, the aim is estimate the age of that person. Next to that, the model is expected to capture the salient facial regions in the spatial domain and the salient phase during facial expression in the temporal domain. Our model is composed of four functional modules: 1) a convolutional appearance module for appearance modeling, 2) a spatial attention module for spatial (facial) salience detection, 3) a recurrent dynamic module for facial dynamics, and 4) a temporal attention module for discriminating temporal salient frames. The proposed model is illustrated in Figure 4.1. We will elaborate on the four modules in a bottom-up fashion and explain step by step how they are integrated into an end-to-end trainable system.

4.3.1. CONVOLUTIONAL APPEARANCE MODULE

Convolutional neural networks (CNNs) have achieved great success for automatic latent representation learning in the image domain. We propose to employ CNNs to model the static appearance for each image of the given video. Compared to conventional handcrafted image features which are generally designed independently of the aimed task, the features learned automatically by CNNs tend to describe the aimed task more accurately due to the parameters learning by back-propagation from the loss function of the aimed task.

Our model contains three convolutional layers with coarse-to-fine filters and subsequent two fully-connected layers. The output of the last fully-connected layer is fed as input to recurrent modules at the corresponding frame. Response-normalization layers [77] and Max-pooling layers follow the first and second convolutional layers. The

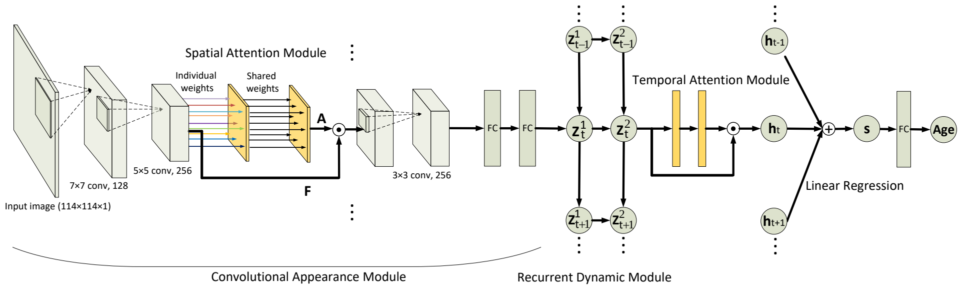


Figure 4.1: The graphical representation of our end-to-end age estimation architecture. Note that we only give one instance of the convolutional appearance module, the spatial attention module and temporal attention module for one frame at time step t. Other frames share exactly the same structure for all three modules.

ReLU [63] function is used as the nonlinear activation function for each convolutional layer as well as the fully-connected layer. We found that creating deeper networks did not lead to a performance improvement, possibly due to comparatively small training dataset used. It should be noted that the spatial attention module presented subsequently in Section 4.3.2 is embedded among convolutional layers to perform facial salience detection. The details are depicted in Figure 4.1.

For each frame, the input image with size $114 \times 114 \times 1$ is filtered by the first convolutional layer with 128 kernels with size 7×7 and stride size 2. The second convolutional layer filters the output of the first convolutional layer with 256 kernels of size 5×5. The spatial attention module (Section 4.3.2) then takes as input the output of the second convolutional layer and extracts the salient regions in the face. The generated attended feature map containing salience is filtered by the third convolutional layer that has 256 kernels of size 3×3. The first fully-connected layer has 4096 neurons while the second fully-connected layer has the same number of neurons as the subsequent recurrent network (as will be explained in Section 4.3.3).

It should be noted that the same convolutional module is shared across all frames in the time domain. Thus, the forward pass of the convolutional module can be computed in parallel for all frames. In the backward pass, the parameters in the convolutional module are optimized by back-propagating output gradients of the upper recurrent module through all frames.

4.3.2. SPATIAL ATTENTION MODULE

The goal of the spatial attention module is to dynamically estimate the salience and relevance of different image portions for the downstream task (in our case age estimation). It is implemented as a feature map filter embedded after one of the convolutional layers in the convolutional appearance modules to preserve the information based on the calculated saliency score.

Formally, suppose the output volume $\mathbf{F}$ of a convolutional layer L has dimensions $M \times N \times C$, with C feature maps of size $M \times N$. The spatial attention module embedded after the convolutional layer L is denoted by a matrix $\mathbf{A}$ with the same size $M \times N$ as the feature map. The element $\mathbf{A}_{ij}$ of $\mathbf{A}$ indicates the attention weight (interpreted as saliency score) for the feature vector $\mathbf{F}_{ij}$ composed of C channels located at (i,j) (i.e., $|\mathbf{F}_{ij}| \equiv C$) in the feature map. Each feature vector corresponds to a certain part of the input image (i.e., receptive field). Therefore, the receptive field of the attention becomes larger when the attention module is inserted in latter convolutional layers. Section 4.5.1 presents an experimental comparison of the different positions of spatial attention module in the convolutional appearance module. In practice, we insert the spatial attention module after the second convolutional layer.

We propose a spatially-indexed attention mechanism to model $\mathbf{A}$. Concretely, the attention weight $\mathbf{A}$ is modeled by two fully-connected layers: the first layer is parameterized by individual weights for each entry of feature map while the second layer shares the transformation weights across the whole feature map. Thus the attention weight $\mathbf{A}_{ij}$ is modeled as:

$$\mathbf{A}_{ij} = \sigma(\mathbf{u}^\top \tanh(\mathbf{W}_{ij}\mathbf{F}_{ij} + \mathbf{b}_{ij}) + c) \tag{4.1}$$

Herein, $\mathbf{W}_{ij} \in \mathbb{R}^{d\times C}$ is the transformation matrix for the first fully-connected layer and $\mathbf{u}^\top \in \mathbb{R}^d$ is the weight vector for the second fully-connected layer to fuse the information from different channels and c is a bias term. A sigmoid function σ is employed as the activation function at the top layer of the attention module to constraint the attention weight to lie in the interval $[0,1]$. The obtained attention matrix $\mathbf{A}$ controls the information flowing into the subsequent layer in the convolutional appearance module by an element-wise multiplication to each channel (feature map) of F:

$$\mathbf{I} = \mathbf{F} \odot \mathbf{A} \tag{4.2}$$

Here $\mathbf{I}$ is the output of the spatial attention module, which is fed into the subsequent layer of the convolutional appearance module.

It is worth mentioning that we use individual weights ($\mathbf{W}_{ij}$) for the first layer and shared weights $\mathbf{u}$ for the second fusion layer in the spatial attention module (this will be called a spatially-indexed mechanism). The first layer is expected to capture the local detailed (fine-grained) variation while the second fusion layer is designed to capture the global variation and smooth the attention distribution. It is different from the design of the soft attention model for image caption generation [50], in which the transformation weights are shared in both two layers of the attention model. In that scenario, the attention model is used to capture the related objects in the input image to each word of generated caption and the objects are always easily separable from the background scene in the image. By contrast, we aim to capture the salient parts in a face image, which requires to model more detailed variation. Employing shared weights in both layers tends to blur the spatial variation. Besides, the typical attention model is translation-invariant. Namely, if the picture is rearranged, the attention would be very similar, whereas our attention is spatially-indexed. Section 4.5.1 provides an comparison between different attention mechanisms by visualizing the learned attention weight distribution.

4.3.3. RECURRENT DYNAMIC MODULE

The temporal facial dynamics are expected to contribute to age estimation, which has been demonstrated by Dibeklioğlu et al. [87]. In contrast to the handcrafted dynamics features they use [87], we propose to employ recurrent networks to capture the underlying temporal information automatically. The potential advantages of using recurrent networks are that (1) they learn relevant dynamics feature to the aimed task (age estimation) smoothly and progressively over time; (2) all modules in our model can be trained jointly in an end-to-end manner to be compatible with each other.

Suppose the output appearance feature of last fully-connected layer of convolutional appearance module is $\mathbf{p}_t$ at frame t, then the hidden representation $\mathbf{z}_t$ is calculated by:

$$\mathbf{z}_t = g(\mathbf{W}\mathbf{p}_t + \mathbf{V}\mathbf{z}_{t-1} + b) \tag{4.3}$$

Herein $\mathbf{W}$ and $\mathbf{V}$ are the transformation matrices for appearance feature in current frame and the hidden representation in the previous frame. We use a ReLU function as the activation function g since it eliminates potential vanishing-gradient problems. In practice, we employ two-layer recurrent networks in our recurrent dynamic module, which is expected to potentially learn more latent temporal dynamics than single-layer recurrent networks.

4.3.4. TEMPORAL ATTENTION MODULE

The attention scheme can be leveraged not only for the selection of the salient facial regions in the spatial domain, but also for the selection of the salient sequential segments (frames) in the temporal domain. Hence we propose to use a temporal attention module on top of the recurrent dynamic module to capture the temporal salience information. The temporal attention module produces an attention weight as the salience score for each frame, thereby filtering the output information flow from the recurrent dynamic module.

Formally, suppose the output hidden-unit representation of the recurrent dynamic module is $\mathbf{z}_t$ at the frame t, then the temporal attention score e_t is modeled by a two-layer perceptron:

$$e_t = \sigma(\mathbf{v}^\top \tanh(\mathbf{M}\mathbf{z}_t + \mathbf{b}) + c) \tag{4.4}$$

Here $\mathbf{M} \in \mathbb{R}^{n' \times n}$ is the weight matrix and $\mathbf{b}$ is the bias term for the first perceptron layer, $\mathbf{v} \in \mathbb{R}^{n'}$ is the fusion vector of the second layer. Here n' is a hyper-parameter that is the dimension of transformed mid-representation. Again, we use a sigmoid function to constrain the values between 0 and 1. We employ this perceptron to measure the relevance of each frame to the objective task, i.e., age estimation. Next, the attention score is normalized over the whole video to get the final temporal attention weight o_t:

$$o_t = \frac{e_t}{\sum_{t'=1}^{T} e_{t'}} \tag{4.5}$$

The obtained temporal attention weights are used to control how much information for each frame is taken into account to perform the age estimation. Concretely, we calculate the weighted sum of the hidden-unit representation for all frames of the recurrent dynamic module to be the information summary $\mathbf{s}$ for the whole video:

$$\mathbf{s} = \sum_{t=1}^{T} o_t \mathbf{z}_t \tag{4.6}$$

Ultimately, the predicted age of the corresponding subject involved in the video is estimated by a linear regressor:

$$\tilde{y} = \mathbf{k} \cdot \mathbf{s} + b \tag{4.7}$$

where **k** contains the regression weights.

4.3.5. END-TO-END PARAMETER LEARNING

Given a training dataset $\mathscr{D} = \{\mathbf{x}^{(n)}_{1,\dots,T_{(n)}}, y^{(n)}\}_{n=1,\dots,N}$ containing N pairs of facial videos and their associated subject's age, we learn the involved parameters of all four modules (convolutional appearance module, spatial attention module, recurrent dynamic module, and temporal attention module) and the final linear regressor jointly by minimizing the mean absolute error loss of the training data:

$$\mathscr{L} = \frac{1}{N} \sum_{n=1}^{N} |\tilde{y}^{(n)} - y^{(n)}| \tag{4.8}$$

Since all modules and the above loss function are analytically differentiable, our whole model can be readily trained in an end-to-end manner. The loss is back-propagated through four modules successively using back-propagation through time algorithm [64] in the recurrent dynamic module and normal back-propagation way in other parts.

4.4. EXPERIMENTAL SETUP

4.4.1. UVA-NEMO SMILE DATABASE

We evaluate the performance of our proposed age estimation architecture on the UvA-NEMO Smile Database, which was collected to analyze the temporal dynamics of spontaneous/pose smiles for different ages [124]. The database is composed of 1240 smile videos (597 spontaneous and 643 posed) recorded from 400 subjects (185 female and 215 male). The involved subjects span an age interval ranging from 8 to 76 years. Figure 4.2 presents the age and gender distribution.

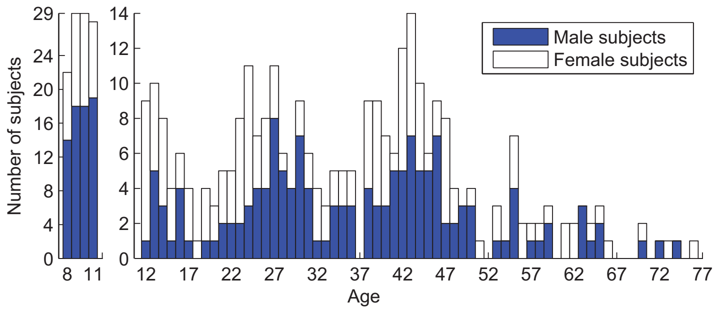


Figure 4.2: Age and gender distributions of the subjects in the UvA-NEMO Smile database.

To collect posed smiles, each subject was asked to pose a smile as realistically as possible. Spontaneous smile was elicited by short and funny video segments. For each subject, approximately five minutes of recordings were made and the genuine smiles were then segmented. A balanced number of spontaneous and posed smiles are selected and annotated by the consensus of two trained annotators for each subject. Each segment of video starts/ends with neutral or near-neutral expressions.

4.4.2. TRACKING AND ALIGNMENT OF FACES

To normalize face images in terms of rotation and scale, 68 landmarks on facial boundary (17 points), eyes & eyebrows (22 points), nose (9 points), and mouth (20 points) regions are tracked using a state-of-the-art tracker [125]. The tracker employs an extended version of Constrained Local Neural Fields (CLNF) [126], where individual point distribution and patch expert models are learned for eyes, lips and eyebrows. Detected points by individual models are then fit to a joint point distribution model. To handle pose variations, CLNF employs a 3D latent representation of facial landmarks.

The movement of the tracked landmarks is smoothed by the 4253H-twice method [127] to reduce the tracking noise. Then, each face image (in videos) is warped onto a frontal average face shape using a piecewise linear warping. Notice that the landmark points are in the same location for each of the warped/normalized faces. Such a shape normalization is applied to obtain (pixel-to-pixel) comparable face images regardless of expression or identity variations. The obtained images are cropped around the facial boundary and eyebrows, and scaled so as to have a resolution of 114×114 pixels. Images are then converted to gray scale.

4.4.3. SETTINGS

Following the experimental setup of Dibeklioğlu et al. [87], we apply a 10-fold cross-validation testing scheme with the same data split to conduct experiments. There is no subject overlap between folds. Each time one fold is used as test data and the other 9 folds are used to train and validate the model. The parameters are optimized independently of test data. For the recurrent dynamic module, the number of hidden units is tuned by selecting the best configuration from the set $\{128, 256, 512\}$ using a validation set. To prevent over-fitting, we adopt Dropout [128] in both the convolutional networks and the recurrent networks and we augment the loss function with L2-regularization terms. Two dropout values, one for the recurrent dynamic module and one for the convolutional appearance module, are validated from the option set $\{0, 0.1, 0.2, 0.4\}$. The L2-regularization parameter λ is validated from the option set $\{0, 1e^{-4}, 3e^{-4}, 5e^{-4}, 1e^{-3}, 3e^{-3}, 5e^{-3}\}$. We perform gradient descent optimization using RMSprop [129]. The gradients are clipped between -5 and 5 [65] to avoid potential gradient explosion.

4.5. EXPERIMENTS

We first investigate the different mechanisms to implement spatial attention and validate the advantages of our proposed spatially-indexed mechanism over other options. Then we present the qualitative and quantitative evaluation on our model respectively, especially to validate the functionality of each module. Subsequently, we compare our model with state-of-the-art methods. Finally, we make a statistical analysis on predicted error distributions to investigate the difference between the method based on the handcrafted features and our method with automatically learned features.

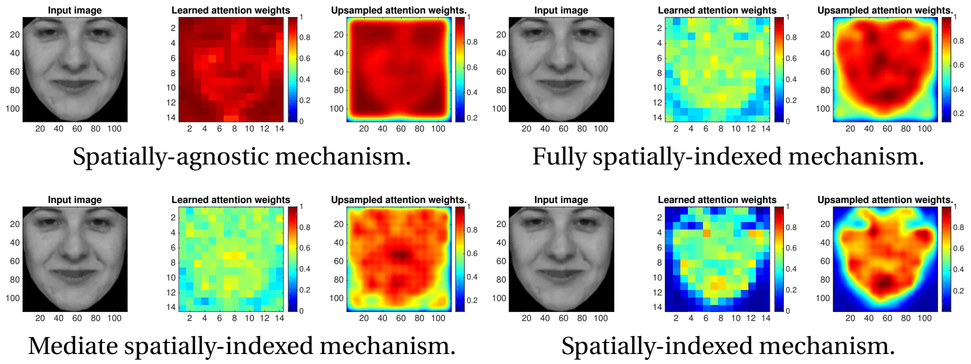


Figure 4.3: The visualization of the learned attention weights on four different spatial attention mechanisms. For each group of plots corresponding to a mechanism, we first present the original input image, subsequently we visualize the learned weights from the spatial attention module and finally we up-sample the attention distribution back to the size of input image by a Gaussian filter. Note that the spatial attention module is inserted after the 2nd convolutional layer in this set of experiments.

4.5.1. INVESTIGATION OF SPATIAL ATTENTION MODULES

We first conduct experiments to investigate the effectiveness of the proposed spatially-indexed attention mechanism compared to other options for the spatial attentions module. Then we illustrate the effect of position where the spatial attention module is inserted in the convolutional appearance module.

COMPARISON OF DIFFERENT SPATIAL ATTENTION MECHANISMS.

We propose a spatially-indexed attention mechanism indicated in Equation (4.1) to model the spatial attention weight $\mathbf{A}$. In order to validate the design motivation behind it, we investigate the difference of four different mechanisms:

- **Spatially-agnostic** mechanism: both $\mathbf{W}$ in the first layer and $\mathbf{u}$ in the second layer are shared across all entries of the feature map, which is the typical attention model [50].
- **Fully spatially-indexed** mechanism: both the transformation weights $\mathbf{W}$ and $\mathbf{u}$ are individually designed for each entry of the feature map.
- **Mediate spatially-indexed** mechanism: the first layer shares the transformation weights $\mathbf{W}$ while the second layer model each entry by individual weight $\mathbf{u}$.
- **Spatially-indexed** mechanism (adopted by our model): the weight $\mathbf{W}$ of the first layer is individually designed and the weight $\mathbf{u}$ in the second layer is shared.

Figure 4.3 presents the qualitative comparison between these four mechanisms. For each group of images, we first visualize the learned spatial attention weights for each option directly (the middle plot of each group), then we up-sample it back to the initial size of the input image by a Gaussian filter (the last plot of each group). This allow us to visualize the receptive field of attention.

It shows that the attention distribution of the spatially-agnostic mechanism appears blurred and less contrasting than other mechanisms. It is only able to capture the salient regions around mouth and near eyebrow. It even gives high scores to the background. It is not surprising since the receptive fields of each entry overlap, hence it is hard for the shared weights to capture the fine-grained differences. The fully spatially-indexed mechanism can roughly capture the contour of the facial regions in the images but with no highlights inside the facial area. This is because individual weights can hardly model the spatial continuity in the face. In contrast, the spatially-indexed mechanism achieves the best result among all options. Furthermore, adopting shared weights in the first layer and individual weights in the second layer (mediate spatially-indexed mechanism) is much worse than the other order. It is probably because the individual weights can hardly take effect after the smoothing by the shared weights. Therefore, our model employs the spatially-indexed mechanism, which can not only clearly distinguish the face from the background, but also capture the salient regions like the area under the eye, area around mouth and two nasolabial folds in cheeks. More examples are presented in Section 4.5.4.

THE EFFECT OF THE POSITION OF THE SPATIAL ATTENTION MODULE

Theoretically, the spatial attention module can be placed after any convolutional layer in the appearance module. However, the latter convolutional layers output feature maps with larger receptive fields for each entry than the previous layers, which leads to more overlapping receptive fields of adjacent entries. As a result, each spatial attention weight also corresponds to a larger receptive field in the input image. Figure 4.4 shows the learned spatial attention weights after inserting the spatial attention module after different convolutional layers. In the case that the spatial attention is placed after the first layer, the distribution of learned attention weights is very noisy. This is because the small receptive fields of each attention weight results in excessively fine-grained modeling, which causes over-fitting. In contrast, placing the spatial attention module after the third convolutional layer generates more coarse and less contrasting attention weight distributions, which weakens the effect of the spatial attention module. We achieve a good balance by inserting the spatial attention module after the second convolutional layer, as shown in Figure 4.4.

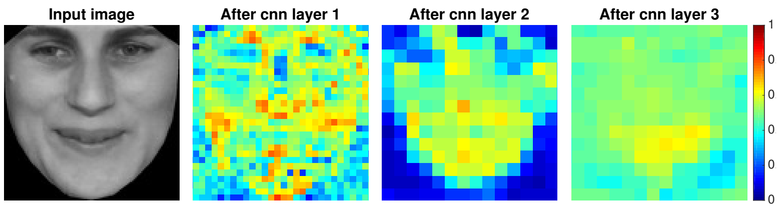


Figure 4.4: The visualization of the learned spatial attention weights when placing the spatial attention module after different convolutional layers. Note that we adopt the spatially-indexed mechanism described in Section 4.5.1.

4.5.2. QUANTITATIVE EVALUATION OF FUNCTIONALITY OF EACH MODULE

Next we perform the quantitative evaluation on the four functional modules of our module, by evaluating the functionality of each module and the contribution it makes to performance of the whole architecture. To this end, we conduct ablation experiments which begin with the single convolutional appearance module in the system and then

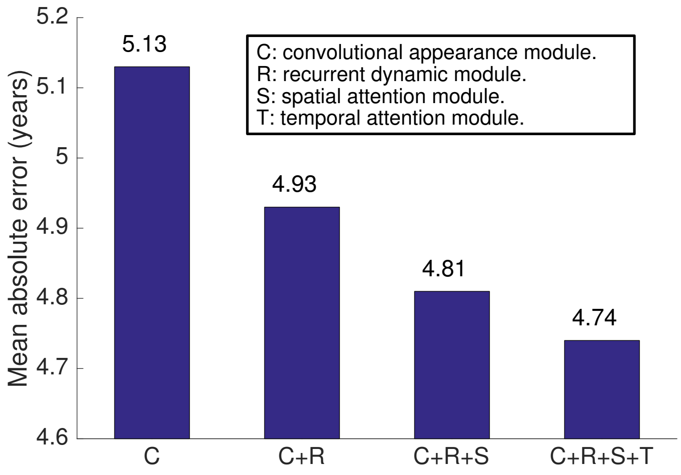


Figure 4.5: Mean absolute error (years) for different functional modules on the UvA-NEMO smile database.

incrementally augments the system by one module at a time. When we only employ the convolutional appearance module (without modeling dynamics) in our system, we perform age estimation on each single image in the video and then average the predicted results as the final age estimation. Figure 4.5 presents the performance of all ablation experiments.

The individual convolutional appearance module in the system achieves an age estimation performance of 5.13 years' mean absolute error (MAE), which is an encouraging result considering the fact that it only performs appearance learning (without any dynamics information involved for age estimation). More detailed comparisons to hand-crafted features are made in Table 4.1 presented in subsequent Section 4.5.3. Equipping the system with the recurrent dynamic module results in 4.93 years' MAE, which indicates that the dynamics learning by recurrent dynamic module makes a substantial improvement. Subsequently, the spatial attention module is added into the system to capture the spatial salience in each facial image, and the MAE is decreased to 4.81 years. We will present a qualitative visualization of learned spatial attention salience in Section 4.5.4 and Figure 4.8. Finally, including the temporal attention module, leading to our full end-to-end system, results in the best performance with 4.74 years' MAE.

It should be mentioned that the power of the temporal attention module is actually not fully exploited, since this data has been segmented to retain the smile phase mostly. Most of irrelevant segments before and after the smile have been removed. The temporal attention module will be shown qualitatively to be capable of capturing the key smile segment precisely in Section 4.5.4 and Figure 4.9. Hence more improvement by the temporal attention module is expectable given temporally noisier data.

4.5.3. COMPARISON TO OTHER METHODS

Next, the model is compared with existing methods for age estimation, that can be applied to sequential images (videos). According to the mechanism of the feature design,

Table 4.1: Mean absolute error (years) for different methods on the UvA-NEMO smile database.

Method		MAE (years) for Different Age Ranges								
		0-9	10-19	20-29	30-39	40-49	50-59	60-69	70-79	All
Spatio-temporal	VLBP [117]	10.69	12.95	15.99	18.54	18.43	16.58	23.80	26.59	15.70 (±12.40)
	LBP-TOP [130]	9.71	11.01	14.19	15.88	16.75	15.29	19.70	23.71	13.83 (±10.97)
Dynamics	Deformation [87]	4.85	8.72	12.22	13.06	13.53	11.55	14.13	17.82	10.81 (±8.85)
	Displacement [86]	5.42	9.67	11.98	14.53	12.77	15.42	20.57	20.35	11.54 (±11.49)
Appearance	IEF, Fusion [87]	3.54	4.38	5.43	6.74	6.01	8.96	13.52	14.05	5.71 (±4.65)
	LBP, Fusion [87]	4.18	4.99	6.31	7.37	6.19	8.67	10.34	12.93	6.12 (±5.11)
	CNNs	2.30	3.09	4.69	5.26	5.82	10.58	17.20	23.29	5.13 (±5.68)
Appearance+Dynamics	IEF+Dynamics [87]	3.96	4.45	4.50	5.29	4.74	6.85	12.43	11.94	5.00 (±4.25)
	LBP+Dynamics [87]	3.49	4.68	5.13	5.85	5.24	7.05	12.17	12.00	5.29 (±4.36)
	Our model	1.79	2.45	4.26	4.97	5.36	11.86	16.43	23.12	**4.74** (±5.70)
	Number of Samples	158	333	215	171	250	66	30	17	1240

these baseline methods can be classified into four categories as listed in Table 4.1:

- **Spatio-temporal**: Hadid et al. [117] propose to employ the spatio-temporal information for the age interval classification. Particularly, they extract volume LBP (VLBP) features and feed them to a tree of four SVM classifiers. Another method using spatio-temporal information is proposed to extract the LBP histograms from Three Orthogonal Planes (LBP-TOP): XY (two spatial dimensions in a single image), XT (X dimension in the image and temporal space T) and YT [130]. Thus the information in temporal domain is also utilized together with information in spatial image domain. These two spatio-temporal methods are implemented for age estimation as baselines by Dibeklioğlu et al. [87], from where we report the results.
- **Appearance + Dynamics**: Dibeklioğlu et al. [87] is the first study which leverages both the facial dynamics and appearance information for age estimation. They propose several handcrafted dynamics features specifically for facial expressions and combine them with appearance features to perform age estimation through a hierarchical architecture. They combine their dynamics features with four different kinds of appearance descriptors in their system. Among them we select two combinations with the best performance as our baselines: dynamics + IEF (Intensity-based encoded aging features [99]) and dynamics + LBP (local binary patterns) [91].
- **Dynamics**: We incorporate the baseline models using sole dynamics information. Following Dibeklioğlu et al. [87], we compare the deformation-based features and the displacement dynamics features [86].
- **Appearance**: We also compare our method to appearance-based approaches that solely employs IEF and LBP features [87], where age estimations from the first and the last frame of a smile onset (a neutral and an expressive face, respectively) are averaged for the final prediction. Furthermore, we evaluate a modified version of our method that uses only convolutional appearance module.

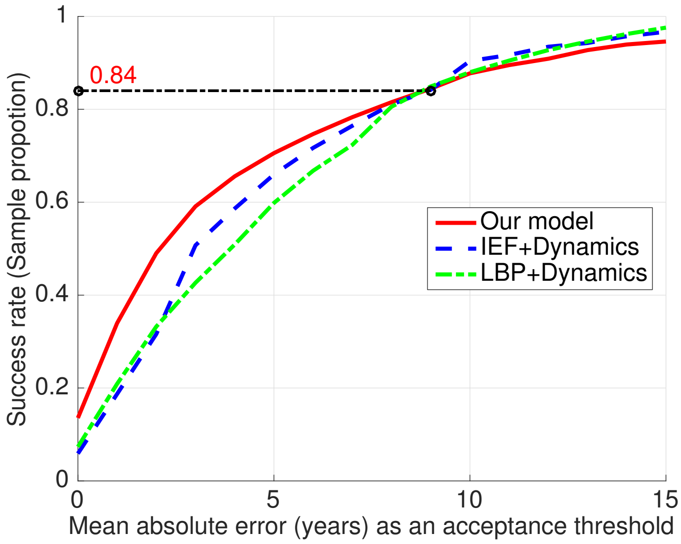


Figure 4.6: Cumulative distribution of the mean absolute error for different models using both appearance and dynamics on the UvA-NEMO Smile Database.

Performance Comparison

Table 4.1 shows the mean absolute errors (MAE) obtained by four categories of age estimation methods mentioned before. Our model achieves the best performance considering all the age ranges with the minimum MAE of 4.74 years. While spatio-temporal methods perform worst, the methods utilizing both appearance and dynamics are more accurate than the methods based on sole appearance or dynamics. It illustrates the importance of both appearance and dynamics to the task of age estimation.

In particular, the performance of our model is better than the other two methods using both appearance and dynamics: *IEF+Dynamics* and *LBP+Dynamics* [87]. Figure 4.6 shows the success rate as a function of the MAE for these three methods. For age deviations up to nine years, our model outperforms the other two methods. For larger age deviations, the model is slightly worse. Our model suffers from some severe estimation errors on a few samples. These samples appear to be from the high-age groups, where our model is severely hampered by the low sample size in these groups. Figure 4.2 and Table 4.1 both show that the number of samples in these age ranges is much less than that in the younger age ranges. Compared to handcrafted features used in *IEF+Dynamics* and *LBP+Dynamics* [87], the convolutional appearance module and recurrent dynamic module in our model require more training data.

The Effect of Training Size per Age

To investigate the effect of training size per age on the performance of three *Appearance+Dynamics* models, we conduct experiments where the data are reduced by removing ages for which a low number of samples are available. The results are presented in Figure 4.7. The experiments begin with the case that all the samples involved (threshold

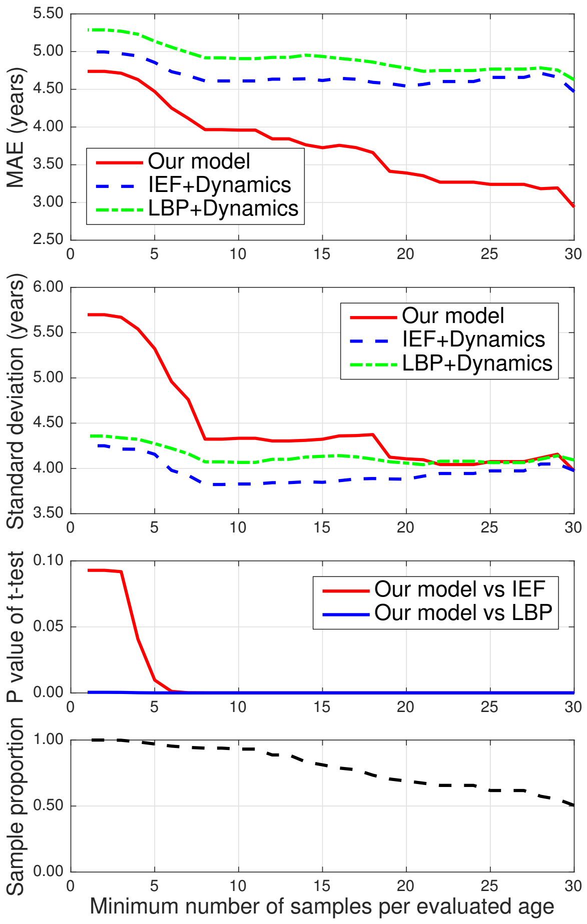


Figure 4.7: Performances of three methods using both the appearance and dynamics on the UvA-NEMO Smile Database as a function of minimum number of samples per evaluated age.

= 1). As the threshold (minimum number of samples) is increased, the performance gap between our model and the other methods becomes larger: the MAE of our model decreases much faster than the other two methods and the variance also drops deeply at the beginning of the curve. A t-test shows that our model significantly outperforms other methods when the threshold on the number of sample is larger than 5 ($p < 0.01$). They are actually quite encouraging results, since these results in turn indicate that larger data tend to explore more potential of our model and make it more promising than the other two methods on the task of age estimation.

4.5.4. QUALITATIVE EVALUATION OF ATTENTION MODULES

In this section, we qualitatively evaluate the spatial attention module and temporal attention module by visualizing the learned attention weights.

Spatial Attention Module

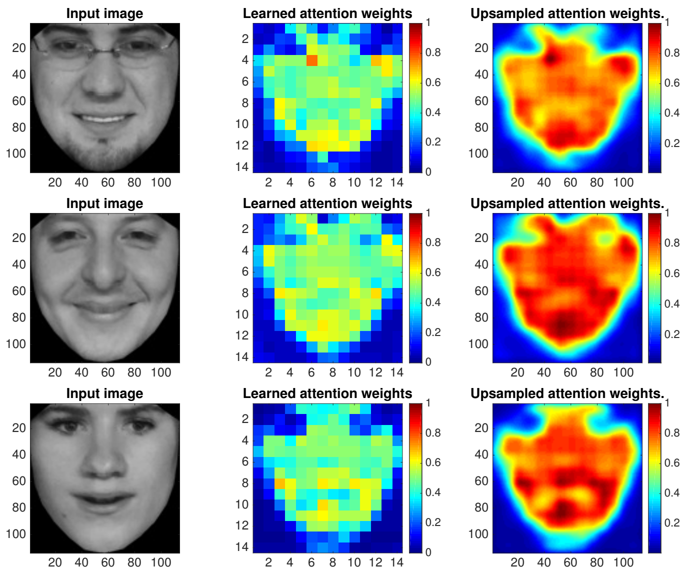


Figure 4.8: The heat map visualization of the learned attention weights by our spatial attention module. For each subject, the middle plot corresponds to attention weights and the last plot is the up-sampled attention weight distribution back to the size of initial input image by a Gaussian filter.

In Figure 4.8, three sample images are presented to visualize the results of spatial attention module. For this module the optimal configuration is used: it uses the spatially-indexed mechanism and includes the spatial attention module after the second convo-

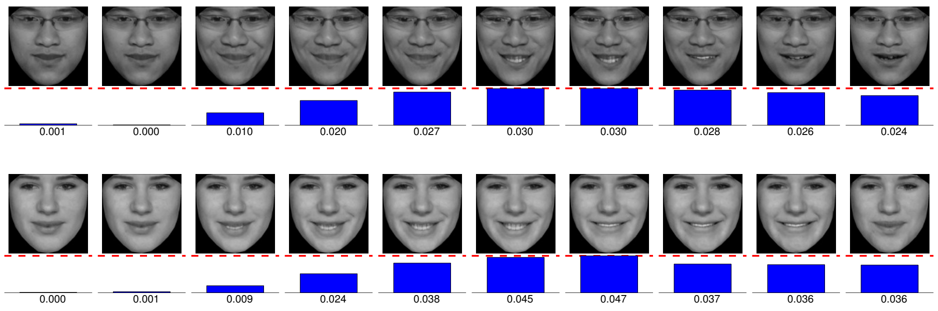


Figure 4.9: The visualization of learned temporal attention weights for examples from test set of UvA-NEMO smile database. Higher bar indicates larger attention weight. The attention weight is indicated for representative frames. Our temporal attention modules is able to cut off neutral faces in the beginning phase of the smiling and assign higher value to frames with higher degree of smiling. Note that attention weights for all frames in a video are normalized (Equation 4.4) to make them sum up to 1. The red dash line indicates the maximum value of attention weights.

lutional layer. The images on the left are the original inputs, the images in the middle are heat maps of the attention values, and the images on the right are upscaled heat maps to the original resolution. These heat images show that our spatial attention module is able to not only discriminate the core facial region from the background accurately, but also capture the salient parts inside the facial region. Specifically, the area under eyes, nasal bridge, two nasolabial folds, and area around mouth (especially mentolabial sulcus) are detected as the salient parts. It is reasonable since these areas tend to generate wrinkles easily when smiling, which are discriminative features for the task of age estimation.

THE TEMPORAL ATTENTION MODULE

To demonstrate the temporal salience detection, the learned temporal attention weights for several representative frames from two video samples are shown in Figure 4.9. Each row shows the smile progression from a neutral state to smiling and back to neutral. For the neutral faces at the beginning, our module predicts very small temporal attention weights. As the degree of smiling increases, the attention weight goes up accordingly, until to the peak value. It should be noted that the attention value grows rapidly with the appearance of the two nasolabial folds, which is consistent with the facial salience captured by the spatial attention module (shown in Figure 4.8). Then the attention value decreases with the recession of smiling. However, the value still retains a relatively high value at the last frame. It is partially because the hidden representation of the last frames contains the information of all previous frames as well as key frames about smiling, hence they are still helpful for age estimation. Besides, the smile videos in the given database do not end with a perfectly netural face. Otherwise, the attention weight would continuously decrease for the latter neutral faces.

4.6. CONCLUSION

In this work, we present an attended end-to-end model for age estimation from facial expression videos. The model employs convolutional networks to learn the effective appearance features and feed them into recurrent networks to learn the temporal dynamics. Furthermore, both a spatial attention mechanism and a temporal attention mechanism are added to the model. The spatial attention can be integrated seamlessly into the convolutional layers to capture the salient facial regions in each single image, while the temporal attention is incorporated in recurrent networks to capture the salient temporal frames. The whole model can be trained readily in an end-to-end manner. Provided that a sufficient number of samples are available for training, we show the strong performance of our model on a large smile database. Specifically, our model makes a substantial improvement over the state-of-the-art methods.

In future work, we aim to leverage the pre-trained convolutional neural networks on large image data for the appearance learning instead of training our convolutional appearance module from scratch. This would not only accelerate the training speed but also allows employing quite deeper architectures and abundant existing image data to improve the performance of the appearance learning.

5

Modeling Time Series Similarity with Siamese Recurrent Networks

Traditional techniques for measuring similarities between time series are based on hand-crafted similarity measures, whereas more recent learning-based approaches cannot exploit external supervision. We combine ideas from time-series modeling and metric learning, and study siamese recurrent networks *(SRNs) that minimize a classification loss to learn a good similarity measure between time series. Specifically, our approach learns a vectorial representation for each time series in such a way that similar time series are modeled by similar representations, and dissimilar time series by dissimilar representations. Because it is a similarity prediction models, SRNs are particularly well-suited to challenging scenarios such as signature recognition, in which each person is a separate class and very few examples per class are available. We demonstrate the potential merits of SRNs in within-domain and out-of-domain classification experiments and in one-shot learning experiments on tasks such as signature, voice, and sign language recognition.*

This chapter is in preparation for publication.

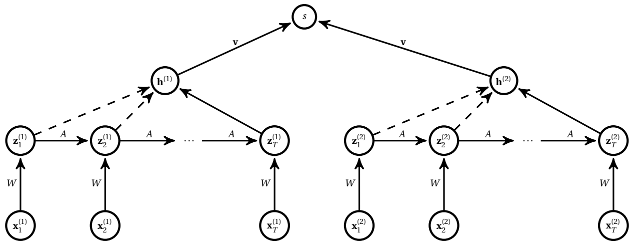


Figure 5.1: Graphical representation of the Siamese Recurrent Network (SRN). For the SRN-L model, the feature representations $\mathbf{h}$ are obtained by taking the hidden unit activations at the last timestep, $\mathbf{z}_T$ (solid line). For the SRN-A model, the feature representations $\mathbf{h}$ are obtained by averaging the hidden unit activations $\mathbf{z}$ over all timesteps (solid and dashed lines). The SRN outputs a scalar similarity measure s.

5.1. Introduction

Successful classification, verification, or retrieval of time series requires the definition of a good similarity measure between time series. Classical approaches to time-series analysis handcraft such similarity measures [11, 131], which limits their ability to incorporate information on the relative scale of features in the similarity measure. Other approaches use unsupervised learning in order to define the similarity measure [2, 12], which has the disadvantage that it cannot exploit class label information in determining which features are most relevant for the underlying similarity structure.

In this paper, we study a novel model for time-series analysis that *learns* a similarity measure over pairs of time series in a *supervised* manner. The proposed model combines ideas from metric learning with that of learning embeddings for time series using recurrent networks. The model takes as input two time series, which are both processed by the same recurrent network to produce a representation for each of time series. The similarity between the time series is defined as a weighted inner product between the resulting representations. All parameters of the model are learned jointly by minimizing a classification loss on pairs of similar and dissimilar time series. We refer to the resulting model as *siamese recurrent network* (SRN). The structure of the SRN is illustrated in Figure 5.1. We evaluate the performance of two variants of the SRN in within-domain classification and out-of-domain classification experiments representing a range of different machine-learning tasks.

The model we study in this chapter is of particular interest in challenging learning settings in which the number of classes is large and the number of training examples per class is limited. An example of such a setting is an online signature verification task. Here each person who provided one or more signatures is considered to be a separate class, and the number of training examples per person is extremely limited. Such a task may benefit from sharing parameters between classes by learning a global similarity measure over the set of all pairs of time series, which is what the SRN does. We perform one-shot learning experiments to illustrate the potential merits of the global similarity measure over time series learned by our models.

5.2. RELATED WORK

Traditional approaches to measuring time-series similarity such as dynamic time warping (DTW; Vintsyuk [11], Sakoe and Chiba [131]) use handcrafted similarity measures that are not adapted to the observed data distribution. This shortcoming was addressed by the introduction of similarity measures that first fit a generative model to the data, such as Fisher, TOP, marginalized, and product-probability kernels [12, 19, 132, 133]. In particular, Fisher kernels have seen widespread adoption in computer vision [13]. While these methods benefit from modeling the data distribution before the computation of pairwise similarities, they are limited in that they cannot exploit available supervised class or similarity information, which may hamper their performance in classification problems. By contrast, the time-series similarity approach we study in this work is based on *supervised learning*. It combines ideas from modeling time series using recurrent networks with those from metric learning. We discuss related work on both topics separately below.

Recurrent networks learn a representation for each timestep that is influenced by both the observation at that time step and by the representation in the previous timestep [59, 64]. The recurrent nature of the models equips them with a memory that is capable of preserving information over time. This has made them popular for tasks such as language [60, 134], handwriting [62], and image generation [61], and music prediction [65]. SRNs employ a pair of standard recurrent networks, the parameters of which are shared between the two networks. It differs from prior work in the loss that it minimizes: instead of minimizing a "generative" loss such as negative log-likelihood, it minimizes a loss that encourages representations to be close together for similar time series and far apart for dissimilar time series.

Metric learning techniques learn a similarity measure on data that lives in a vectorial space. While several studies have explored learning non-linear "metrics" by backpropagating pairwise losses through feedforward networks [135–141], most prior work on metric learning focuses on learning Mahalanobis metrics; prominent examples of such studies include Goldberger *et al.* [142], Weinberger and Saul [143], Davis *et al.* [144]; and Xing *et al.* [145]. Our work is most similar to latent coincidence analysis (LCA; Der and Saul [146]) in terms of the loss it is minimizing, but it differs substantially from LCA in that it backpropagates the loss through the recurrent network that is modeling the time series.

5.3. SIAMESE RECURRENT NETWORKS

A time-series similarity model produces a single similarity value for each input pair of time series (with potentially different lengths). Similarly to a siamese network, our time-series similarity model employs two neural networks that share their network parameters in order to extract comparable hidden-unit representations from the inputs. The resulting hidden-unit representations are compared to compute the similarity between the two time series. The parameters of the neural networks and the comparison function are learned jointly in a supervised manner to predict whether two time series are similar or not. We use recurrent networks as the basis for our siamese architecture, leading to the siamese recurrent network (SRN) depicted in Figure 5.1. The advantage of using recurrent networks is that they allow our model (1) to extract relevant features for the similarity computation and (2) to remember these relevant features over time when needed. The resulting features have the same size irrespective of the time series length.

Suppose we are given two time series $X^{(1)} = \left\{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_{T_1}^{(1)}\right\}$ and $X^{(2)} = \left\{\mathbf{x}_1^{(2)}, \ldots, \mathbf{x}_{T_2}^{(2)}\right\}$ whose lengths are respectively T_1 and T_2. The hidden-unit representations $\mathbf{z}_t^{(1)}$ and $\mathbf{z}_t^{(2)}$ in the SRN model are defined as:

$$\mathbf{z}_t^{(i)} = \mathrm{g}\left(\mathbf{W}\mathbf{x}_t^{(i)} + \mathbf{A}\mathbf{z}_{t-1}^{(i)} + \mathbf{b}\right). \tag{5.1}$$

We use a rectified linear unit (ReLU) function $\mathrm{g}(x) = \max(0, x)$ as this activation function eliminates potential vanishing-gradient problems.

The hidden-unit representations obtained from the two sub-networks for the corresponding input time series, $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$, are combined to compute the SRN's prediction for the similarity of the two time series. We consider two approaches for comparing hidden-unit representations.

In the first approach, the element-wise product between the hidden representations on the last time steps T_1 and T_2 is computed and the output is a weighted sum of the resulting products. This approach encourages the recurrent networks to remember relevant features over time, thereby making these features available for the final similarity computation.

In the second approach, all the hidden-unit representations for each of the two time series are averaged over time to construct a single feature representation for both time series, and the resulting feature representations are combined in the same way as before to compute the time-series similarity. This approach removes the burden on the recurrent networks to memorize all important features over time, but may potentially pollute the time-series features by averaging over time.

Mathematically, the two approaches compute the following latent representations $\mathbf{h}$ for each time series:

- The **SRN-L** (last timestep) model:

  $$\mathbf{h}^{(i)} = \mathbf{h}\left(X^{(i)}\right) = \mathbf{z}_T^{(i)}. \tag{5.2}$$

  The recurrent connections in recurrent networks allow it to memorize the previous inputs in the hidden states in a recursive way. Consequently, the hidden units in the last time step should be able to store the information accumulated in the time domain for the whole time series. Therefore, we conjecture it is capable of modeling the entire time series.
- The **SRN-A** (average) model:

  $$\mathbf{h}^{(i)} = \mathbf{h}\left(X^{(i)}\right) = \frac{1}{T}\sum_{t=1}^{T} \mathbf{z}_t^{(i)}. \tag{5.3}$$

  By averaging the hidden units $\mathbf{z}$ over time, this model treats the information of each time step equally and avoids the potential memory-vanishing problem whilst still considering the temporal information in the previous time steps when computing hidden-unit representations.

Denoting the latent representations obtained from the two recurrent networks as $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$, the SRN model defines the similarity of the two time series as:

$$s\left(X^{(1)}, X^{(2)}\right) = \frac{1}{1 + e^{-\mathbf{v}^\top \left[\operatorname{diag}\left(\mathbf{h}^{(1)} \mathbf{h}^{(2)\top}\right)\right] + c}}. \tag{5.4}$$

Herein, the similarity between two time series is defined as a weighted inner product between the latent representations $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$. Such similarity measures between hidden-units activations have previously been used as part of attention mechanisms in speech recognition [147], machine translation [48], and handwriting generation [62].

5.3.1. PARAMETER LEARNING

Suppose we are given a training set $\mathscr{T}$ containing two sets of in total N pairs of time series, a set with pairs of similar time series $\mathscr{S}$ and a set with pairs of dissimilar time series $\mathscr{D}$. We learn all parameters $\Theta = \{\mathbf{A}, \mathbf{W}, \mathbf{v}, c, \mathbf{b}\}$ of the SRN jointly by minimizing the binary cross-entropy of predicting to which set each pair of time series belongs with respect to the parameters. This is equivalent to maximizing the conditional log-likelihood of the training data:

$$\mathscr{L}(\Theta; \mathscr{T}) = -\Big[\sum_{(n_1,n_2)\in\mathscr{S}} \log s\left(X^{(n_1)}, X^{(n_2)}\right) + \sum_{(n_1,n_2)\in\mathscr{D}} \left(1 - \log s\left(X^{(n_1)}, X^{(n_2)}\right)\right) \Big],$$

where n_1 and n_2 indicate the indices of the first and second time series in a training pair. The loss function is backpropagated through both recurrent networks (the weights of which are shared) using a variant of the backpropagation through time algorithm [64] with gradient clipping between -5 and 5 [65].

The sets $\mathscr{S}$ and $\mathscr{D}$ of similar and dissimilar time series can be constructed in various ways, for instance, by asking human annotators for similarity judgements. When class labels y_n are available for time series $X^{(n)}$, the sets can be defined as $\mathscr{S} = \{(n_1, n_2) : y_{n_1} = y_{n_2}\}$ and $\mathscr{D} = \{(n_1, n_2) : y_{n_1} \neq y_{n_2}\}$. In contrast to time-series classification models [4, 18, 148, 149], this allows SRNs to be used on objects from unknown classes as well. For instance, the SRN may be trained on the signatures of a collection of people, and like any classification model, it can then be used within-domain to verify new signatures of the same people. However, the SRN can also be used out-of-domain to verify the signatures from people that were not present in the training set. The SRN only needs one genuine, verified signature to compute the similarity to a new, unknown signature (one-shot learning). The underlying assumption is that the inter-person variation of the signatures is modeled well by the SRN because it was trained on signatures from many other people.

5.4. EXPERIMENTS

We performed experiments with SRNs on three different datasets in three different learning settings: (1) within-domain similarity prediction, (2) out-of-domain similarity prediction, and (3) one-shot learning. Before presenting the setup and results of our experiments, we first introduce the three datasets below.

Table 5.1: Characteristics of the five datasets considered in our experiments: dimensionality of features, number of classes, number of samples, and the minimum, mean, and maximum length of the time series.

				Time series length		
Dataset	**Dimens.**	**Classes**	**Samples**	**Min.**	**Mean**	**Max.**
Arabic (digit)	13×2	10	8800	3	39	92
Arabic (voice)	13×2	88	8800	3	39	92
MCYT (without forgery)	5×3	100	2500	34	349	1161
MCYT (with forgery)	5×3	100	5000	34	438	2687
Sign	77×2	19	40	760	112	198

5.4.1. DATASETS

We performed experiments on three different datasets.

The *Arabic Spoken Digit dataset* [150] comprises 8,800 utterances of digits produced by 88 different speakers. Each speaker uttered each digit ten times. The data is represented as a time series of 13-dimensional MFCCs that were sampled at 11,025Hz and 16 bits using a Hamming window. We use two different versions of the spoken digit dataset: (1) a *digit* version in which the uttered digit is the class label and (2) a *voice* version in which the speaker of a digit is the class label.

The *MCYT signature dataset* [1] contains online signature data collected from 100 subjects. For each subject, the data comprises 25 authentic signatures and 25 skilled forgeries. The signatures are represented as time series of five features: the x-coordinate, y-coordinate, pressure, azimuth, and elevation of the pen. We consider two different versions of the dataset, namely, a version *without forged* data and a version *with forged* data.

The *American sign language dataset* [151] contains eight manual signs that represent different words and eleven non-manual signs such as head or shoulder motions. The data thus comprises nineteen classes. Each sign was produced five times by eight different subjects, leading to a total of 760 samples. The time series are represented using a hand-crafted feature representation that contains a total of 77 hand motion, hand position, hand shape, and head motion features [151].

Following common practice in time series analysis, we preprocessed all three datasets by applying a sliding window (with stride 1) to the time series, concatenating the features in the frames under the window into a single frame. This enriches the feature representation, making it easier for the models to capture feature gradients. For the Arabic, MCYT, and Sign datasets, we used a window size of 2, 3, and 2, respectively. In Table 5.1, the main characteristics of all five datasets are summarized.

5.4.2. EXPERIMENTAL SETUP

In our experiments, the model parameters of the SRNs were initialized by sampling them from a uniform distribution within an interval $[-0.1, 0.1]$. Training of the model is performed using a RMSprop [152] stochastic gradient descent procedure using mini-batches of 50 pairs of time series. To prevent the gradients from exploding, we clip all gradients [65] to lie in the interval $[-5, 5]$. We decay the learning rate during training by multiplying it by 0.4 every time the AUC on the validation set stops increasing. We applied dropout on the hidden-unit activations of our model: the dropout rate was tuned to maximize the

AUC on a small held-out validation set. Code reproducing the results of our experiments is available on `http://www.anonymized.com`.

In all experiments except for those on the MCYT (with forgery) dataset, we defined the sets of similar and dissimilar time series as suggested in Section 5.3, that is, we define similar time series to be those with the same class label and dissimilar time series to be those with different class labels: $\mathscr{S} = \{(n_1, n_2) : y_{n_1} = y_{n_2}\}$ and $\mathscr{D} = \{(n_1, n_2) : y_{n_1} \neq y_{n_2}\}$. Herein, y_n represents the class label of the time series as described in section 5.4.1. On the MCYT (with forgery) dataset, we define the positive pairs in the same way but we define the set of negative pairs $\mathscr{D}$ slightly differently: the negative pairs are pairs of a genuine signature and a forged version of the same signature. These negative pairs are more difficult to distinguish, as a result of which training on them will likely lead to better models.

We compare the performance of our SRNs with that of three variants of our model, and with three baseline models. The three variants of our model we consider are: (1) a feedforward variant of SRN-A, called SN-A, that removes all recurrent connections from the model, *i.e.*, in which $\mathbf{A} = \mathbf{0}$ but which still averages the hidden representation over time; (2) a feedforward variant of SRN-L, called SN-L, that removes all recurrent connections from the model and uses the hidden representation of the last time step; and (3) a naive logistic model that removes all hidden units from the model and that predicts similarities by averaging all features over time and computing a weighted sum of the element-wise product of the resulting feature representations. These three variants of SRNs allow us to investigate the effect of the recurrent connections and non-linearities on the prediction performance of our models.

The three time-series similarity models we use as baseline models are: (1) dynamic time warping [11]; (2) Fisher kernels [12]; and (3) Fisher vectors [13]. Details of these three baseline models are given below.

Dynamic time warping (DTW; Vintsyuk [11]) measures time series similarities by aligning both time series and summing the pairwise distances between all corresponding frames, minimized over the set of all possible alignments between the two time series. An alignment is a set of (potentially many-to-many) correspondences between frames, with the restriction that correspondences cannot be crossing each other in time. DTW similarities can be computed efficiently using a dynamic-programming algorithm. Despite its simplicity, DTW has been quite successful, in particular, on problems in which the time series are already relatively well aligned and the time series show some clear salient features in time. We leave comparisons with approaches that combine dynamic time warping and metric learning [153] to future work.

Fisher kernels measure the similarity between two time series by the inner product of the log-likelihood gradients that are induced by the time series with respect to the parameters of a generative model [12]. Our generative model of choice for time series is the hidden Markov model (HMM). Mathematically, we denote the gradient of the log-likelihood $L(X^{(n)})$ of a time series $X^{(n)}$ with respect to the parameters of the HMM as $\mathbf{g}_n = \left[\forall \theta \in \Theta : \frac{\partial L(X^{(n)})}{\partial \theta}\right]$. We define the Fisher kernel similarity κ between two time series as an inner product between their corresponding gradients:

$$\kappa\left(X^{(i)}, X^{(j)}\right) = \mathbf{g}_i^\top \mathbf{U}^{-1} \mathbf{g}_j. \tag{5.5}$$

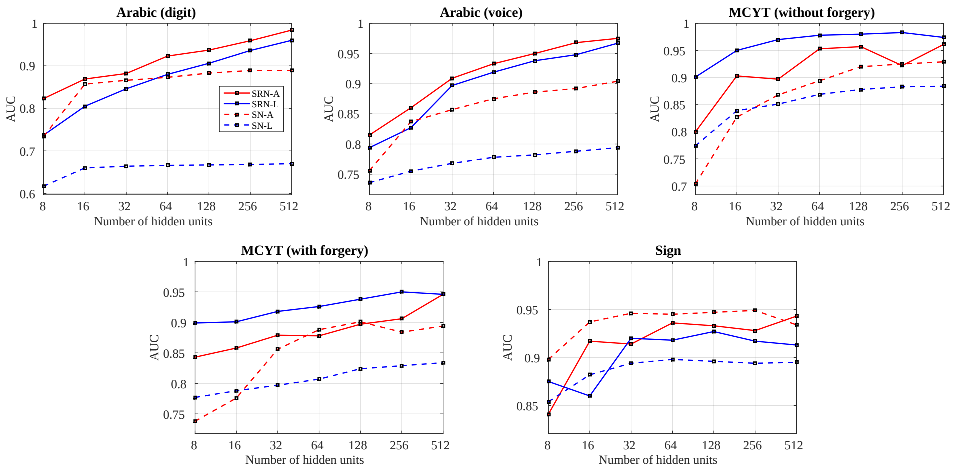


Figure 5.2: Area under the receiving-operator curve curve (AUC) of our two variants of Siamese Recurrent Networks (SRN-A and SRN-L) on five datasets as a function of the number of hidden units (higher is better). For reference, the performance of SRNs without recurrent connections (SNs) is also shown. All results were obtained by averaging over five repetitions. The standard deviation of the results is typically smaller than 0.01.

Herein, the matrix $\mathbf{U}$ is the Fisher information metric, which is replaced with identity matrix $\mathbf{I}$ in our experiments. The number of hidden states of our HMMs is tuned by maximizing the AUC on a small, held-out validation set.

Fisher vectors compute the same gradients $\mathbf{g}_n$ as before, but instead of computing their inner products, we concatenate the gradients $\mathbf{g}_i$ and $\mathbf{g}_j$ to obtain a feature representation of the time-series pair $\left(X^{(i)}, X^{(j)}\right)$. Such Fisher vector representation are commonly used in computer vision [13]. Because the concatenated Fisher vectors cannot directly measure time-series similarity, we perform 1-nearest classification on the collection of similar and dissimilar pairs to predict whether a pair of time series is similar. (In other words, the time series similarity is the negative Euclidean distance between the example and its nearest pair of similar time series in the concatenated Fisher vector space.)

5.4.3. Results

Below, we separately present the results for the three learning settings we considered: (1) within-domain similarity prediction, (2) out-of-domain similarity prediction, and (3) one-shot learning. We also present t-SNE visualizations of the learned time-series representations.

Within-Domain Similarity Prediction

We first evaluate the within-domain similarity prediction performance of the SRN: we randomly split the time series into a training and a test set, and we measure the ability of the models to accurately predict whether pairs of time series in the test set are similar

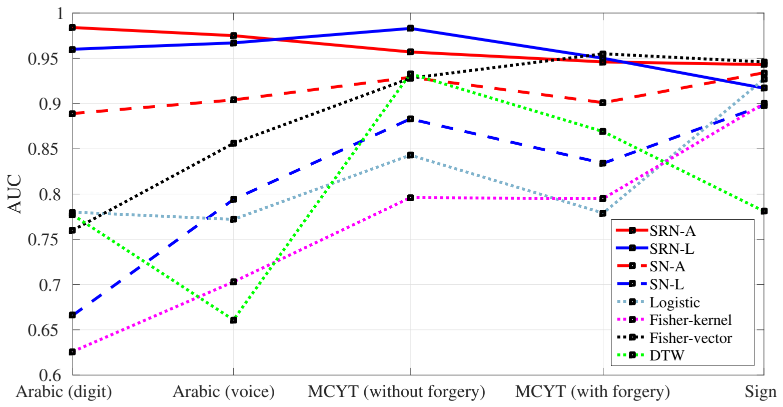


Figure 5.3: Area under the receiving-operator curve curve (AUC) on five different datasets using eight different time-series similarity learning models in a within-domain similarity prediction setting (higher is better). See text for details.

or not in terms of the area under the receiving-operator curve (AUC). We opt for the AUC as a performance measure because it naturally deals with the potential imbalance in the sizes of $\mathscr{S}$ and $\mathscr{D}$. We refer to this experiment as *within-domain* because all classes in the test data were also observed during training.

Figure 5.2 presents the within-domain similarity prediction performance of SRNs as a function of the number of hidden units in the model on five different datasets. We present results for both the variant that averages all hidden-unit activations over time (SRN-A) and the variant that uses only the hidden unit activations at the last timestep (SRN-L). The reported results were averaged over five repetitions, randomly initializing the parameter of the models in each repetition. The figure also reports the performance of models without recurrent connections, called a Siamese network (SN, where SN-A is a Siamese network with averaged hidden activations and SN-L is a network that uses the last time step activations). From the results presented in Figure 5.2, we make three main observations.

First, the results show that the performance of SRNs tends to increase with the number of hidden units, in particular, on challenging datasets such as the Arabic speech datasets. This shows that SRNs effectively use the additional capacity that is provided by additional hidden units to learn more informative features for the time-similarity measurements. In our experiments, we did not observe much overfitting, although overfitting is likely to occur when the number of hidden units is increased much further.

Second, we observe that there is no clear winner between averaging hidden unit activations over time (SRN-A) and using the activations at the last timestep (SRN-L). This suggests that the recurrent networks in the SRN-L models are at least partly successful in remembering relevant features over time.

Third, we observe that the recurrent connections in the SRN models are, indeed, helpful: the SRN models outperform their counterparts without recurrent connections

Table 5.2: Area under the receiving-operator curve curve (AUC) of eight time-series similarity models on five datasets in an out-of-domain similarity prediction setting (higher is better). The standard deviation of the five repetitions we performed is typically smaller than 0.01. The best performance per dataset is boldfaced. See text for details.

Dataset	**Training classes**	**Test classes**	**Model**							
			SRN-A	**SRN-L**	**SN-A**	**SN-L**	**Logist.**	**DTW**	**Fisher K.**	**Fisher V.**
Arabic (digit)	1-7	8-10	0.681	0.714	**0.768**	0.539	0.761	0.725	0.600	0.561
Arabic (voice)	1-60	61-88	**0.849**	0.788	0.802	0.684	0.730	0.640	0.698	0.630
MCYT (without forgery)	1-70	71-100	0.914	0.920	0.816	0.760	0.824	**0.952**	0.752	0.844
MCYT (with forgery)	1-70	71-100	0.888	0.876	0.828	0.668	0.782	**0.894**	0.805	0.813
Sign	1-14	15-19	**0.862**	0.670	0.748	0.565	0.836	0.729	0.770	0.566

(SNs) in nearly all experiments[1]. This result underlines the hypothesis that recurrent connections can preserve features relevant for time-series similarity computations over time. Somewhat surprisingly, the performance of the SN-L models is not as bad as one may expect. It should be noted that the windowing of features makes the feature representation of the last timestep richer, which is sufficient to obtain acceptable performances on some of the datasets.

Comparison with baseline models. Next, we compare the performance between of SRNs with the naive logistic model and three other baseline time-series similarity learning models: (1) dynamic time warping, (2) Fisher kernels, and (3) Fisher vectors (see section 5.4.2 for details). We used the same experimental setup as in the previous experiment, but we tuned the main hyperparameters of the models (the number of hidden units in SRNs and SNs; the number of HMM hidden states for Fisher kernels and Fisher vectors) on a small held-out validation set. Figure 5.3 presents the results of these experiments.

The results of these experiments show that, indeed, the SRN can be a very competitive time-series similarity model, even when trained on relatively small datasets. In particular, SRNs substantially outperform the baselines models on the Arabic (digit), Arabic (voice), and MCYT (without forgery) datasets. On most datasets, the Fisher vectors are the best baseline model (they perform substantially better than standard Fisher kernels), which is line with results in the literature [13]. The naive logistic model performs substantially worse than the SRN models, which suggests that hidden units are essential in solving difficult similarity assessment problems.

Dynamic time warping (DTW) performs reasonably well on relatively simple datasets such as the Sign dataset, but its performance deteriorates on more challenging datasets in which the similarity labels are not aligned with the main sources of variation in the data, such as the Arabic (voice) dataset: the main sources of variation in this dataset are likely due to the differences in the digits being uttered, whereas the similarity labels we are interested in concern the speaker of the digit and not the digit itself. DTW (as well as Fisher vectors and kernels) cannot exploit this information, which explains its inferior

[1]It should be noted that because we preprocess the time-series data by windowing features, the SN is actually a convolutional network that is very similar to the time-delay neural networks of Bromley *et al.* [135].

Table 5.3: Classification accuracy of one-shot learning models of an 1-nearest neighbor classifier using three different similarity measures on four different datasets (higher is better). The best performance per dataset is boldfaced. See text for details.

Dataset	SRN-A	SRN-L	DTW
Arabic (digit)	0.618	0.613	**0.801**
Arabic (voice)	**0.273**	0.228	0.151
MCYT (without forgery)	0.418	0.548	**0.913**
Sign	**0.599**	0.381	0.531

performance on the Arabic (voice) dataset.

OUT-OF-DOMAIN SIMILARITY PREDICTION

In the next set of experiments, we measure the performance of SRNs on *out-of-domain* similarity prediction: we use the same experimental setup as before, however, we split the training and test data in such a way that the set of class labels appearing in the training set and the set of class labels appearing in the test set are disjoint. This is a more challenging learning setting, as it relies on the time-series similarity model exploiting structure that is shared between classes in order to produce good results. We obtain the test data by selecting 3 out of 10 classes on the Arabic (digit) dataset, 28 out of 88 classes on the Arabic (voice) dataset, 30 out of 100 classes on the MCYT datasets, and 5 out of 19 classes on the Sign dataset. As before, we measure the performance of our models in terms of AUC, and we tune the hyperparameters of the models on a validation set. The results of these experiments are presented in Table 5.2.

From the results presented in the table, we observe that the strong performance of SRNs on difficult datasets such as the Arabic (voice) datasets generalizes to the out-of-domain prediction setting. This suggests that, indeed, the SRN models are able to learn some structure in the data that is shared between classes. On the (much smaller) MCYT datasets, however, dynamic time warping outperforms SRNs. Most likely, this result is caused by the SRNs (which have high capacity) overfitting on the classes that are observed during training.

ONE-SHOT LEARNING

To further explore the potential of SRNs in out-of-domain settings, we performed an experiment in which we measured the performance of SRNs in one-shot learning. We adopt the same dataset splits as in 5.4.3 to obtain train and test data. On the training data, we train the SRNs to learn a similarity measure for time series. This similarity measure is used to train and evaluate a nearest-neighbor classifier on the test set. We use only a single time series per class from the test set to train the nearest-neighbor classifier, and use the remaining time series in the test set for evaluation. We measure the classification accuracy using leave-one-per-class-out validation.

The results are presented in Table 5.3. For datasets that have clear salient features, like the MCYT, and to a lesser degree the Sign dataset, DTW performs well. For more complex data, the SRN performs well provided that sufficient training data is available. For the Arabic (digit) dataset, the seven classes used in training are insufficient for the

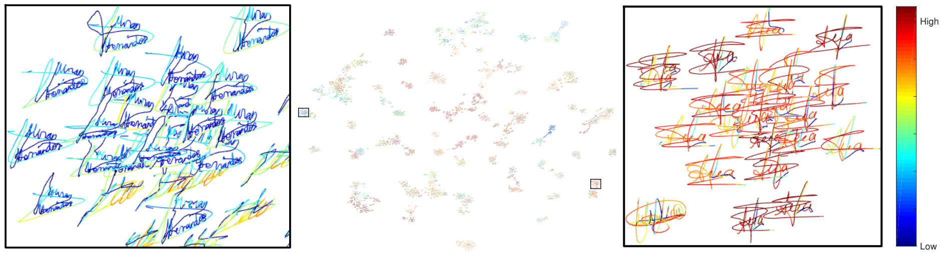


Figure 5.4: t-SNE map of the 2,500 signatures in the MCYT test set (100 subjects) data based on embeddings computed by an SRN-L. The signatures were drawn by integrating the pen movements over time, and colors indicate the pen pressure (red indicates high pressure and blue indicates low pressure). A full-resolution version of this map is presented in the supplemental material.

SRN, and the SRN overfits on those classes. On the Arabic (voice) dataset 60 classes are available, which allows the SRN to fully exploit its potential.

Visualizing the Representations

The one-shot learning experiment presented above exploits an interesting property of the SRN model, namely, that it learns a single embedding for a time series. An advantage of this is that the resulting time-series embeddings can be used in a wide variety of other learning algorithms that operate on vectorial data, such as alternative classification techniques, clustering models, *etc.* To obtain more insights into what the SRN models have learned, we apply t-SNE [154] on embeddings obtained by a SRN-L on the MCYT (without forgery) test set. Figure 5.4 shows a map of the 2500 signatures in the test set; the signatures were drawn by integrating the pen movements over time. The color indicates the pen pressure. We refer the reader to the supplemental material for a full-resolution version of this map. The t-SNE visualization shows that, indeed, the SRN-L is capable of grouping similar signatures together very well.

In Figure 5.5, we show a t-SNE map of the Arabic (voice) test set constructed on SRN-L embeddings. For comparison, we also show a t-SNE map of the same data, based on pairwise distances computed with DTW. The two maps clearly show the potential advantage of SRN: it has used the supervised similarity information to group all the utterances corresponding to a single class together, something that DTW is unable to do due to its unsupervised nature.

5.5. Conclusions

We have investigated models for learning similarities between time series based on supervised information. Our study shows that a combination of ideas from metric learning and deep time-series models has the potential to improve the performance of models for time-series classification, retrieval, and visualization. The proposed siamese recurrent networks (SRNs) are particularly effective compared to alternative techniques in settings in which the similarity function that needs to be learned is complicated, or when the number of labeled time series for some of the classes of interest is limited. When a rea-

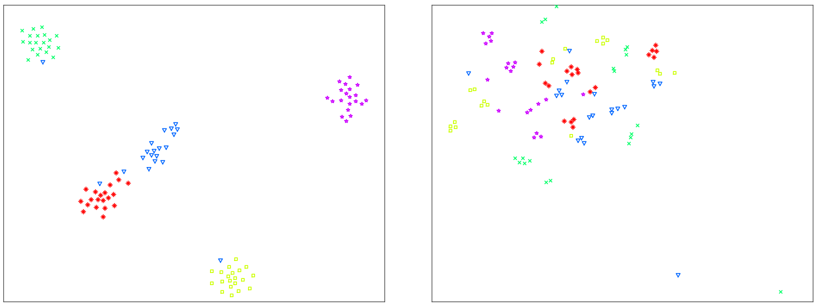

(a) SRN-L. (b) DTW.

Figure 5.5: t-SNE maps of the Arabic (voice) test data from five randomly selected classes, constructed based on (a) siamese recurrent network (last timestep) embeddings of the time series and (b) pairwise similarities computed using dynamic time warping.

sonably large collection of examples of similar and dissimilar time series is available to train the models, the siamese recurrent networks can produce representations that are suitable for challenging problems such as one-shot learning or extreme classification of time series. This result is in line with earlier results for siamese convolutional networks by, for instance, Kamper *et al.* [155].

This study is an initial investigation into learning similarities between time series, and we foresee several directions for future work. In particular, we intend to explore variants of our model architecture: (1) that employ a bilinear model to measure the similarity of the RNN representations; (2) that employ long-term short-term units [8] or gated recurrent units [7] instead of the simple rectified linear units we are currently using; (3) that employ multiple layers of recurrent units; and (4) that have a tree structure or generic (planar) graph structure instead of the current sequential structure. The latter extension would make our models applicable to problems such as molecule classification [156]. We also plan to explore improvements to our learning algorithm. In particular, our current implementation selects negative pairs of time series in a somewhat arbitrary manner: in all our experiments, we select negative examples uniformly at random for the set of all candidate negative pairs. We plan to investigate approaches that perform a kind of “hard negative mining” during learning, akin to some modern metric learning [143] and multi-modal learning [157] approaches. We also plan to study applications of SRNs in, for instance, learning word-discriminative acoustic features [158].

6
Interacting Attention-gated Recurrent Networks for Recommendation

Capturing the temporal dynamics of user preferences over items is important for recommendation. Existing methods mainly assume that all time steps in user-item interaction history are equally relevant to recommendation, which however does not apply in real-world scenarios where user-item interactions can often happen accidentally. More importantly, they learn user and item dynamics separately, thus failing to capture their joint effects on user-item interactions. To better model user and item dynamics, we present the Interacting Attention-gated Recurrent Network (IARN) which adopts the attention model to measure the relevance of each time step. In particular, we propose a novel attention scheme to learn the attention scores of user and item history in an interacting way, thus to account for the dependencies between user and item dynamics in shaping user-item interactions. By doing so, IARN can selectively memorize different time steps of a user's history when predicting her preferences over different items. Our model can therefore provide meaningful interpretations for recommendation results, which could be further enhanced by auxiliary features. Extensive validation on real-world datasets shows that IARN consistently outperforms state-of-the-art methods.

This chapter is published as:
Wenjie Pei*, Jie Yang*, Zhu Sun, Jie Zhang, Alessandro Bozzon and David M.J. Tax. Interacting Attention-gated Recurrent Networks for Recommendation. The 26th ACM International Conference on Information and Knowledge Management (CIKM), 2017.

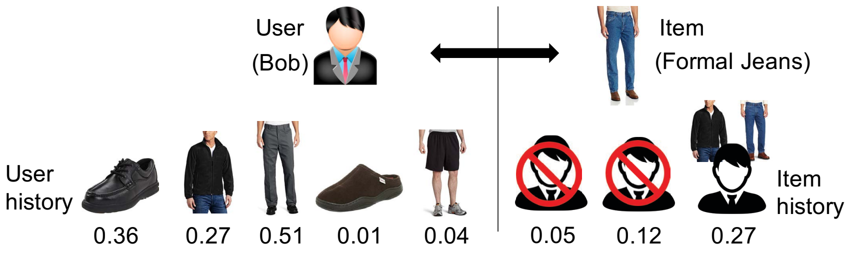


Figure 6.1: Example of user-item interactions determined by dependent user dynamics and item dynamics. The numbers below user and item history are the attention scores inferred by our proposed model, which are used to select relevant time steps in user and item history to accurately predict the user's preference over the item.

6.1. Introduction

Recommendation is a fundamental task to enable personalized information filtering, thus to mitigate the information overload problem [159]. The goal is to learn user preferences from historical user-item interactions, based on which recommend relevant items. In reality, user preferences often evolve over time, affected by dynamic user inclinations, item perception and popularity. Temporal context therefore has been recognized as an important type of information for modeling the dynamics of user preferences. It has extensive applications, ranging from movie recommendation [160], music recommendation [161], to location recommendation [162].

Most existing methods [161–165] model the temporal dynamics by extending the latent factor model (LFM) [166] with handcrafted features, so as to describe certain temporal patterns of user-item interactions. For example, they either bin user-item interactions into time windows, assuming similar user behavioral patterns in the same window [161, 162], or adopt a time decay function to under-weight the interactions occurring deeper into the past [163, 164]. The handcrafted features, though proven to be effective, cannot capture complex temporal patterns in reality [167]. More importantly, these methods cannot automatically select important interaction records in user-item interaction history when modeling user preferences. This greatly limits their application in real-world scenarios where user-item interactions can often happen accidentally.

Recently, recurrent neural network (RNN) [168] based methods have emerged as a promising approach to model the temporal dynamics of user preferences [167, 169, 170]. RNN captures both the latent structures in historical user-item interactions – through hidden units – and their dynamics along the temporal domain. Unlike LFM based methods, these methods are nonparametric, thus can learn inherent dynamics that are more complex and suitable for making recommendations. A specific type of *gated* RNN, i.e. Long Short-Term Memory (LSTM) [8], is employed by the state-of-the-art recommendation method [167] to model both user and item dynamics. The gating mechanism is adopted to balance the information flow from the current and previous time steps, thus can more effectively preserve historical information over time for recommendation.

Nevertheless, LSTM models the gate w.r.t. each hidden unit instead of the whole time step, making it difficult to interpret the importance of each time step for the final recommendation. More importantly, gates for modeling user dynamics and item dynamics so far are learned separately. In real-world scenarios, however, user and item dynamics are dependent on each other, and can jointly affect user-item interactions. Consider the target user Bob in Figure 6.1, who is interested in both formal clothing (e.g., leather shoes and trousers) and casual clothing (e.g., casual shoes and shorts), as described by his purchasing history. We observe that Bob buys a pair of formal jeans, which were historically bought by users with various interests. The interaction between Bob and the formal jeans is therefore determined by his interest in formal clothing and the inherent property of the formal jeans, namely, the formal style. Such an interest and property could only be learned from historical user-item interactions when no additional auxiliary features are given. Therefore, to accurately capture Bob's preference over the formal jeans, the recommendation model should be able to identify the important time steps of Bob's purchasing history when he bought formal clothing. Similarly, in the history of formal jeans it should be able to identify time steps when they were bought by users who are also interested in formal style clothing, thus to capture the item property relevant to Bob's interest.

In this chapter, we introduce the Interacting Attention-gated Recurrent Network (IARN) which adopts the attention model to measure the relevance of each time step of user history and item history for recommendation. In particular, we propose a novel attention scheme which allows IARN to learn the relevance – measured by attention scores – of time steps in user and item history in an interacting way, so as to capture the dependencies between user and item dynamics in shaping user-item interactions. As a result, IARN can selectively memorize different time steps of a user's history when predicting her preferences over different items, thereby providing meaningful interpretations for the prediction. For instance, attention scores learned by IARN for the example in Figure 6.1 are shown under the user and item history in the figure (note this example is based on our results on a real-world dataset).

IARN could be further enhanced by incorporating auxiliary features of users or items. In this chapter we provide methods to integrate IARN with auxiliary features organized in a flat or a hierarchical structure. More specifically, our main contributions include:

- We extend recurrent networks for modeling user and item dynamics with a novel gating mechanism, which adopts the attention model to measure the relevance of individual time steps of user and item history for recommendation.
- We design a novel attention scheme which allows the user- and item-side recurrent networks to interact with each other, thus to capture the dependencies between user and item dynamics to improve recommendation accuracy and interpretability.
- We propose the IARN method implementing the interacting attention-gate as described above, and show how it can be further enhanced by auxiliary features organized in different structures.
- We conduct extensive experiments to evaluate the proposed IARN method on six real-world datasets, demonstrating that IARN consistently outperforms state-of-the-art methods.

6.2. RELATED WORK

This section provides an overview of state-of-the-art recommendation methods related to our work. We review them from two orthogonal perspectives: (1) the underlying recommendation models; (2) the incorporation of side information for recommendation.

6.2.1. UNDERLYING RECOMMENDATION MODELS

The recently pervasive recommendation methods can be broadly categorized into two types, namely, the latent factor model based methods and the neural network based ones.

Latent Factor Model. Due to the high efficiency, state-of-the-art recommendation methods have been dominated by Latent Factor Model (LFM). It decomposes the high-dimensional user-item rating matrix into low-dimensional user and item latent matrices. A panoply of algorithms have been proposed to date based on LFM, including matrix factorization (MF) [171], Bayesian personalized ranking (BPR) [172], collective matrix factorization (CMF) [173], factorization machine (FM) [174], SVD++ [175], to name a few. Despite of their success, LFM based methods suffer from the following essential limitations. First of all, they merely leverage global statistical information of user-item interaction data, while cannot capture fine-grained regularities in the latent factors [68]. Second, LFM based recommendation methods generally learn latent representations of users and items in a linear fashion, which may not be always suitable in real-world scenarios. Besides, most LFM based methods ignore the temporal dynamics of user preferences, assuming that the future user-item interactions are known in advance, which is contradictory with the real-world application. There are a few LFM based methods specifically designed for fusing temporal information, which will be reviewed in section 2.2.

Neural Networks. Stemming from the success in related domains (e.g., computer vision, speech recognition, and natural language processing), Neural Network (NN) based methods have recently attracted a considerable amount of interests from the recommendation community. In contrast to LFM based recommendation methods, NN based methods have shown to be highly effective in capturing local item relationships by modeling item co-occurrence in individual users' interaction records. Typical methods are User2Vec [176] and Item2Vec [177], which are inspired by word embedding techniques [178, 179]. Furthermore, NN based models can learn nonlinear latent representations through the activation functions (e.g., sigmoid, ReLU [63]). For instance, Suvash et al. propose the AutoRec [180] recommendation method based on autoencoders [181]. He et al. propose neural collaborative filtering [182] to learn non-linear interactions between users and items. Recently, the Recurrent Neural Network (RNN) based methods [167, 169, 170, 183] have gained significant enhancement in recommendation thanks to the ability of preserving historical information over time for recommendation. These methods learn time-varying representations of users/items (i.e., hidden-states) in each time step, by taking into account both the present and historical data. The learned states can be used for generating recommendations for the future, therefore being more realistic and attractive for real-world applications. To sum up, NN based methods possess essential advantages and have shown to be more effective to enhance recommendation performance.

6.2.2. INCORPORATING SIDE INFORMATION

To better model user preferences thus to further improve recommendation performance, many researchers endeavor to incorporate side information, i.e., information complementing user-item interactions, into recommendation models. Here we focus on the literature with consideration of two types of side information related to our work, i.e., temporal context and auxiliary features.

Temporal Context. It has been well recognized that user preferences change over time. This can be due to drifting user inclinations for item, or the constantly changing item perception and popularity when new selection emerges [163, 170]. Hence, recommendation methods that capture temporal dynamics of user preferences could provide improved recommendation performance. In the branch of LFM based methods, some take temporal information into consideration based on time windows, assuming user-item interactions in the same window have similar patterns. For instance, Koenigstein et al. [161] and Yuan et al. [162] propose such methods for music and Point-of-Interest recommendation. A disadvantage is that they regard all interactions within the considered time window equally, completely ignoring the relationships of interactions among different windows. In addition, binning user-item interactions aggravates the data sparsity problem. Some other LFM based methods attempt to address these issues by adopting a time decay function to under-weight the instances as they occur deeper into the past. These include TimeSVD++ proposed by Koren [163] and HeteRS proposed by Pham et al. [164]. However, these methods could not capture other types of temporal patterns, e.g., certain user-item interactions could be driven by the long-term interest of a user which could not be modeled in a decay manner. In fact, all LFM based methods handle temporal context by creating handcrafted features, thus cannot capture complex temporal patterns in reality.

Contrarily, RNN based methods are nonparametric, thus can learn inherent dynamics of user preferences that are more complex. For instance, Hidasi et al. [169] propose a RNN based approach for session-based recommendation. Hosseini et al. [183] introduce a recurrent Poisson factorization framework for recommendation. Among different RNN models, Long Short-Term Memory (LSTM) [8] has gained much popularity in recommendation due to their capability in dealing with the gradient vanishing problem [184]. Jing et al. [170] present a LSTM based method to estimate when a user will return to a site and what her future listening behavior will be. Wu et al. [167] propose a LSTM based method, i.e., recurrent recommender network (RRN), to model user and item dynamics. This is the most closely related work to ours. However, one major shortcoming of these gated RNN based methods is that the learned gate lacks interpretability, limiting further improvements of recommendation accuracy. More importantly, these methods model user and item dynamics separately, thus failing to capture their dependencies and their joint effects on user-item interactions.

Motivated by the attention scheme in human foveal vision, attention mechanism has been employed by NN based methods to cope with the data noisy problem by identifying relevant parts of the input for the prediction task. It has been applied in a broad spectrum of disciplines, from natural language processing [185] to computer vision [50, 88]. However, how to effectively exploit the attention mechanism in recommender systems is still an open research question. To the best of our knowledge, we are the first to propose recurrent network based recommendation method that integrates attention mechanism to automatically learn the relevance of individual time steps for recommendation, so as to enhance both recommendation interpretability and accuracy. More importantly, we design a novel attention scheme that allows user- and item-side recurrent networks to interact with each other, thus to capture the dependencies between user and item dynamics and their joint effects on user-item interactions.

Auxiliary Features. Better representations of users and items can also be obtained by incorporating auxiliary features into recommendation, due to the rich semantic information encoded by them. Most existing feature-based recommendation approaches are built upon LFM. These methods are either designed to incorporate features in a flat structure or a hierarchy. For instance, the popular CMF [173] and FM [174] are designed for integrating flat features into recommendation. Recently it has been found that feature hierarchies, i.e., hierarchically organized features, can be more effective in boosting the accuracy as well as the interpretability of recommendation. He et al. [186] devise a visually-aware recommendation model by manually defining the feature hierarchy influence on items. Yang et al. [187] design a recommendation method that automatically learns feature hierarchy influence on user/item by a parameterized regularization traversing from root to leaf features. More recently, Sun et al. [188] introduce a unified recommendation framework that seamlessly incorporates both vertical and horizontal dimensions of feature hierarchies for effective recommendation. In this chapter, we show how to incorporate features organized in both flat and hierarchical structures into our model. Note that although in other domains like nature language processing, a few work [185] attempts to integrate hierarchies into RNN model, there is few such kind of approach in recommendation. Hence, we are the first to explore the effect of features organized in different structures together with recurrent networks to learn optimal representations of users and items for improved recommendation interpretability and accuracy.

6.3. INTERACTING ATTENTION-GATED RECURRENT NETWORKS

Given the historical user-item interaction data as the input, we aim to learn high-quality hidden representations for both users and items, which are then used for subsequent recommendation. The extracted representations are expected to: 1) capture the temporal dynamics contained in both user and item history with physical interpretability of each time step; 2) learn the dependencies between user and item dynamics in shaping user-item interactions; 3) extract semantically rich information from training data through the incorporation of auxiliary features. To achieve these goals, we propose the Interacting Attention-gated Recurrent Network (IARN), which is composed of three modules: Attention-gated Recurrent Module, Interacting Attention Module and Feature Encoder. They are designed correspondingly to the three aforementioned goals.

The overall architecture of IARN is illustrated in Figure 6.2. It employs two recurrent networks to learn compact and effective hidden representations for the paired user and item, respectively. Each recurrent network is composed of an Attention-gated Recurrent Module, an Interacting Attention Module, and a Feature Encoder. Instead of behaving independently, the two recurrent networks interact with each other to model the dependencies between user and item dynamics.

Input and Output Layers. We first describe the input and output layer of IARN, then in the following subsections we will elaborate on the three modules in a top-down fashion

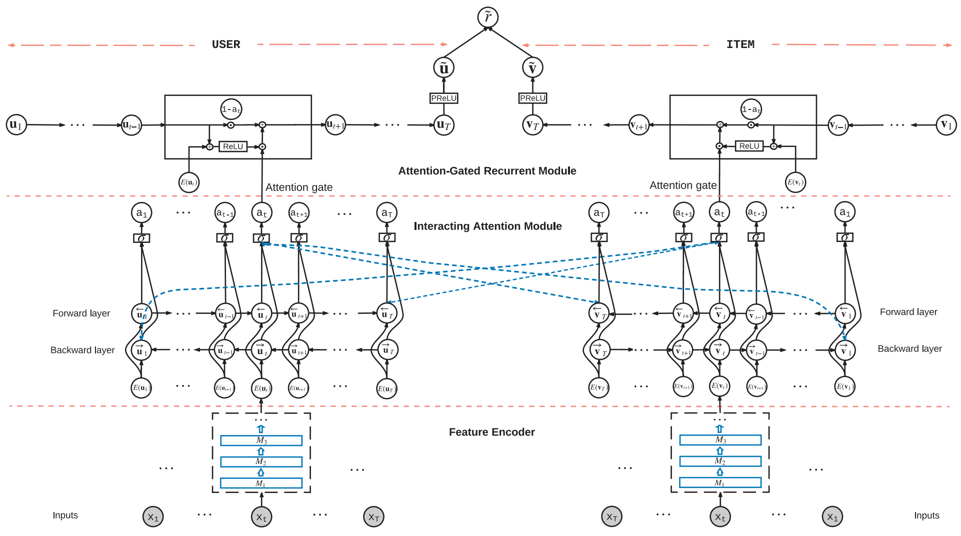


Figure 6.2: Architecture of IARN. Two recurrent networks are employed to learn hidden representations of users and items. Each recurrent network is composed of an Attention-gated Recurrent Module to capture the user/item dynamics, an Interacting Attention Module that models the attention, and a Feature Encoder to extract semantically rich information from the input data. For the sake of clarity, here we only show the connections from a single attention gate in the user (item) network to the forward and backward layers of the item (user) network, however the attention gates of all time steps in the user (item) network are connected to forward and backward layers of the item (user) network. Overall, the two recurrent networks interact with each other to learn the dependencies between user and item dynamics.

to explain step by step how the input and output layers are connected to achieve our goal.

Let $\mathcal{U}$ and $\mathcal{V}$ be the user and item set, respectively. As input, each user $i \in \mathcal{U}$ is described by a sequence $\mathbf{x}_i$, which contains the representations (e.g., the embeddings) of all the items rated by her, ordered by the rating time. Similarly, each item $j \in \mathcal{V}$ is described by a sequence $\mathbf{x}_j$ that contains the representations of all users who have rated the item, ordered by the rating time. Through the three modules, IARN learns from $\mathbf{x}_i$ and $\mathbf{x}_j$ the *hidden representation* of user i, denoted by $\tilde{\mathbf{u}}_i$, and the *hidden representation* of item j, denoted by $\tilde{\mathbf{v}}_j$. $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{v}}_j$ are then used to predicted user i's preference rating $\tilde{r}_{ij}$ over item j via an inner product operation:

$$\tilde{r}_{ij} = \langle \tilde{\mathbf{u}}_i, \tilde{\mathbf{v}}_j \rangle \tag{6.1}$$

6.3.1. ATTENTION-GATED RECURRENT MODULE

In order to learn high-quality hidden representations $\tilde{\mathbf{u}}_{it}$ and $\tilde{\mathbf{v}}_{jt}$, we propose the Attention-gated Recurrent Module to preserve the information of previous time steps with relevance modeled by attention scores, which are obtained by the Interactive Attention Module that will be present in section 6.3.2. Specifically, we construct two attention-gated recurrent modules for the paired user and item, respectively. It should be noted that these two modules do not share parameters, since users and items are not expected to share similar hidden representations. This makes our method different from Siamese

Network [135, 189], a well-known method for object comparison.

Both user- and item-side Attention-gated Recurrent Modules contain two layers, namely, a recurrent layer and a fully-connected layer. The recurrent layer models the temporal dynamics of users and items as *hidden-states*, while the fully-connected layer transform the hidden-states of users and items in the last time step to the *hidden representations* for prediction. We first describe the full-connected layer, then introduce in detail the recurrent layer.

User- and Item-Side Fully-Connected Layers. Denote the last hidden-states of user- and item-side recurrent layers as $\mathbf{u}_{iT_i}$ and $\mathbf{v}_{jT_j}$, respectively. The hidden representations $\tilde{\mathbf{u}}_i$ and $\tilde{\mathbf{v}}_j$ are transformed from these hidden-states by non-linear transformations:

$$\begin{cases} \tilde{\mathbf{u}}_i = g(\widetilde{\mathbf{W}}_u \cdot \mathbf{u}_{iT_i} + \tilde{b}_u) \\ \tilde{\mathbf{v}}_j = g(\widetilde{\mathbf{W}}_v \cdot \mathbf{v}_{jT_j} + \tilde{b}_v) \end{cases} \tag{6.2}$$

Herein, $\widetilde{\mathbf{W}}_u$ and $\widetilde{\mathbf{W}}_v$ are linear transformation parameters of the user- and item-side layers, respectively; $\tilde{b}_u$ and $\tilde{b}_v$ are the bias terms; g is the activation function, for which we use the Parametric Rectified Linear Unit (PReLU) [190]. PReLU allows the output of the unit to be either positive or negative, thus is more suitable for representing users/items – intuitively, a user could either like or dislike certain types of items (e.g., action movies), and an item could either be of a specific type or not.

6

User-Side Attention-gated Recurrent Layer. Given the user i whose corresponding input sequence is $\mathbf{x}_i = \{\mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots\}$. We denoted the attention score at time step t by a_{it}, which is a scalar value between $[0,1]$ inferred by the Interacting Attention Module. The hidden-state of user i at time t is then modeled as

$$\mathbf{u}_{it} = (1 - a_{it}) \cdot \mathbf{u}_{i(t-1)} + a_{it} \cdot \mathbf{u}'_{it} \tag{6.3}$$

where $\mathbf{u}_{i(t-1)}$ is the hidden-state in the previous time step and $\mathbf{u}'_{it}$ is the candidate state value obtained by fully incorporating the input at the current time step:

$$\mathbf{u}'_{it} = g(\mathbf{W}_u \cdot \mathbf{u}_{i(t-1)} + \mathbf{H}_u \cdot E_u(\mathbf{x}_{it}) + b_u) \tag{6.4}$$

where $\mathbf{W}_u$ and $\mathbf{H}_u$ are respectively the linear transformation parameters for the previous and current time steps; b_u is the bias term; and $E_u(\cdot)$ is the Feature Encoder that transforms the original user sequence by considering auxiliary features, which will be detailed in section 6.3.3. We use ReLU for the activation function g.

Equation 6.3 balances the contributions of the input of the current candidate hidden-state and the previous hidden-state with an attention gate described by the attention score a_{it}. Attention gates with high scores will focus more on the current input than previous hidden-states, while recurrent gates with low attention scores will ignore the current input and inherit more information from previous time steps. The attention score therefore quantifies the importance of individual time steps in the final prediction.

Item-Side Attention-gated Recurrent Layer. Similarly, for the item-side recurrent layer, we model the the hidden-state as follows

$$\begin{cases} \mathbf{v}_{jt} = (1 - a_{jt}) \cdot \mathbf{v}_{j(t-1)} + a_{jt} \cdot \mathbf{v}'_{jt} \\ \mathbf{v}'_{jt} = g(\mathbf{W}_v \cdot \mathbf{v}_{j(t-1)} + \mathbf{H}_v \cdot E_v(\mathbf{x}_{jt}) + b_v) \end{cases} \tag{6.5}$$

where $\mathbf{x}_{jt}$ is the input of item j at time t; $\mathbf{W}_v$, $\mathbf{H}_v$, and b_v are the network parameters; a_{jt} is the attention score that serves as attention gate; and $E_v(\cdot)$ is the Feature Encoder for transforming item sequences, introduced in section 6.3.3.

6.3.2. INTERACTING ATTENTION MODULE

We propose the Interacting Attention Module for both users and items to measure the saliency and relevance of the input in each time step to rating prediction. The key point in this module is that the inferred attention score should not only consider the current time step in the sequence on its own side, but also take into account the information of the other side so as to model the interacting dependency between the paired user and item.

User-Side Interacting Attention Module. To maximize the utility of the input sequence, we model the saliency score based on both the input observation at the current time step and the information from neighboring observations in both directions. This is achieved by using a bi-directional RNN [191], which includes a forward layer and a backward layer, as depicted in Figure 6.2. The attention score a_{it} at time step t in Equation 6.3 on the user side is modeled as:

$$a_{it} = \sigma(\mathbf{M}_u{}^{\top} \cdot \tanh(\mathbf{L}_u \cdot (\overrightarrow{\mathbf{u}}_{it}; \overleftarrow{\mathbf{u}}_{it}; \overrightarrow{\mathbf{v}}_{jT_j}; \overleftarrow{\mathbf{v}}_{j1}) + b'_u))) \tag{6.6}$$

Wherein a two-layer network is used to calculate the attention score: $\mathbf{L}_u$ is a matrix as the parameter of the fusion layer that measures the compatibility of information summary in user side and item side; b'_u is the bias term of the fusion layer; and $\mathbf{M}_u$ is the weight vector of the second layer; σ is sigmoid function applied as the activation function to control the attention score to lie between $[0,1]$; $(;)$ denotes the concatenation among vectors; $\overrightarrow{\mathbf{u}}_{it}$ and $\overleftarrow{\mathbf{u}}_{it}$ perform as the summary of context information around time step t in the user sequence x_i. Specifically,

$$\begin{aligned} \overrightarrow{\mathbf{u}}_{it} &= g(\overrightarrow{\mathbf{W}}_u \cdot E_u(\mathbf{x}_{it}) + \overrightarrow{\mathbf{H}}_u \cdot \overrightarrow{\mathbf{u}}_{i(t-1)} + \overrightarrow{b}_u) \\ \overleftarrow{\mathbf{u}}_{it} &= g(\overleftarrow{\mathbf{W}}_u \cdot E_u(\mathbf{x}_{it}) + \overleftarrow{\mathbf{H}}_u \cdot \overleftarrow{\mathbf{u}}_{i(t+1)} + \overleftarrow{b}_u) \end{aligned} \tag{6.7}$$

Therefore, $\overrightarrow{\mathbf{u}}_{it}$ summarizes the sequence from the beginning to time t, while $\overleftarrow{\mathbf{u}}_{it}$ summarizes the sequence from the end to time t.

Similarly, $\overrightarrow{\mathbf{v}}_{jT_j}$ $\overleftarrow{\mathbf{v}}_{j1}$ in Equation 6.6 are the summary of the paired item sequence $\mathbf{x}_j$, whose calculation will be introduced later in detail by Equation 6.9. They are concatenated together with the summary of the user-side sequence, and used as input of the fusion layer. In this way, the resulting attention score a_{it} is used to characterize the relevance of the current time step t of user sequence $\mathbf{x}_i$ conditioned on the paired item sequence $\mathbf{x}_j$.

Item-Side Interactive Attention Module. Similarly, for item-side, we have

$$a_{jt} = \sigma(\mathbf{M}_v{}^{\top} \cdot \tanh(\mathbf{L}_v \cdot (\overrightarrow{\mathbf{v}}_{jt}; \overleftarrow{\mathbf{v}}_{jt}; \overrightarrow{\mathbf{u}}_{iT_i}; \overleftarrow{\mathbf{u}}_{i1}) + b'_v))) \tag{6.8}$$

where $\mathbf{L}_v, b'_v$ are the parameters of the fusion layer, and $\mathbf{M}_v$ is the weight vector of the second layer; $\overrightarrow{\mathbf{v}}_{jt}$ and $\overleftarrow{\mathbf{v}}_{jt}$ perform as the summary of the context information around time step t in the item sequence $\mathbf{x}_j$:

$$\begin{aligned} \overrightarrow{v}_{jt} &= g(\overrightarrow{\mathbf{W}}_v \cdot E_v(\mathbf{x}_{jt}) + \overrightarrow{\mathbf{H}}_v \cdot \overrightarrow{\mathbf{v}}_{j(t-1)} + \overrightarrow{b}_v) \\ \overleftarrow{\mathbf{v}}_{jt} &= g(\overleftarrow{\mathbf{W}}_v \cdot E_v(\mathbf{x}_{jt}) + \overleftarrow{\mathbf{H}}_v \cdot \overleftarrow{v}_{j(t+1)} + \overleftarrow{b}_v) \end{aligned} \tag{6.9}$$

The summary of user sequence, i.e., $\overrightarrow{u}_{iT_i}$ $\overleftarrow{u}_{i1}$, are taken as input for modeling the attention score a_{jt}, so as to condition the learning of a_{jt} on the paired user sequence $\mathbf{x}_i$.

By modeling the attention of each time step in both the user- and item-side networks, our method can capture the interacting dependency and the joint effects of user and item dynamics on user preferences. It thus enable us to gain "second order" insights such as how user preferences are determined by the dynamics of user inclinations and the change of item perception/popularity together.

6.3.3. Feature Encoder

We now introduce Feature Encoder, which is used to extract semantically rich information from the input data for learning high-quality hidden-states. Here we focus on Feature Encoder for processing item-side input, as features of items are in generally richer than users (e.g., the datasets we will take for validation in section 6.4). It is however non-trivial to adapt our method for processing the user-side input when auxiliary features of users are given.

We consider two structures of feature organizations, namely, flat structure and hierarchy. Formally, let $\mathscr{F}$ denote the set of features organized in a flat structure or a hierarchy. Each item $j \in \mathcal{V}$ is affiliated with a subset of features $\mathscr{F}(j) = \{f_j^1, f_j^2, \ldots, f_j^L\}$. The effect of feature f_j^k is modeled as a linear transformation function, denoted by $\mathbf{M}_j^k$, that projects the input $\mathbf{x}_{jt}$ for all $1 \le t \le T_j$ to a new space determined by the feature (i.e., the column space of $\mathbf{M}_j^k$)

$$\mathbf{M}_j^k \cdot \mathbf{x}_{jt} \tag{6.10}$$

The problem is how to combine the effects of different features of $\mathscr{F}(j)$ to project the input for best learning the hidden-states. Considering feature organizations, we design our Feature Encoder as follows.

Flat Feature Encoder. In the case when features are organized in a flat structure, we simply add the effects of different features together. Formally, for the input $\mathbf{x}_{jt}$, the combined effects of all affiliated features $\mathscr{F}(j)$ are given by

$$E_v(\mathbf{x}_{jt}) = \textstyle\sum_{k=1}^{L} \mathbf{M}_j^k \cdot \mathbf{x}_{jt} \tag{6.11}$$

Hierarchical Feature Encoder. In the case when $\mathscr{F}(j)$ is a feature hierarchy, let f_j^1 be the feature in the leaf layer and f_j^L be the root feature. Intuitively, features in top-layers (close to root in the hierarchy) provide more general description of the item, while those in bottom-layers (close to the leaf layer in the hierarchy) provide more refined description. Inspired by the recursive nature of a hierarchy, we consider the recursive parent-children relationships between features in connected layers from the root to leaf layer. In every two connected-layers, the input will be first projected by the parent feature, then by the child feature. By doing so, they will be first mapped to a more general feature space, and then mapped to a more semantically refined feature space. The effects of all affiliated features in different layers will be combined recursively, such that the input can be sequentially mapped to more refined spaces.

Formally, for the input $\mathbf{x}_{jt}$, the combined effects of all affiliated features $\mathscr{F}(j)$ are given by

$$E_v(\mathbf{x}_{jt}) = (\mathbf{M}_j^1 \cdot (\mathbf{M}_j^2 \ldots \cdot (\mathbf{M}_j^L \cdot \mathbf{x}_{jt}) \ldots)) = \prod_{k=1}^{L} \mathbf{M}_j^k \cdot \mathbf{x}_{jt} \tag{6.12}$$

6.3.4. END-TO-END PARAMETER LEARNING

Given the training data $\mathscr{D}_{train}$ containing N instances in the form of $(i, j, r_{ij}, timestamp)$, IARN learns the involved parameters by minimizing the mean squared error loss function:

$$\mathscr{J} = \frac{1}{N} \sum_{r_{ij} \in \mathscr{D}_{train}} (\tilde{r}_{ij} - r_{ij})^2 \tag{6.13}$$

Since all the modules and the above loss function are analytically differentiable, IARN can be readily trained in an end-to-end manner. In the learning process, parameters are updated by the back-propagation through time (BPTT) algorithm [64] in the recurrent layers of the Attention-gated Recurrent Module and the Interacting Attention Module, and by normal back-propagation in other parts. We use RMSprop [152] to adaptively update the learning rate, which has proven to be highly effective for training neural networks. To prevent over-fitting, we use dropout [192] to randomly drop hidden units of the network in each iteration during the training process.

6.3.5. COMPARISON WITH RECURRENT NETWORK BASED METHODS

Comparison with RNN- and LSTM-backbone. One could argue that our framework can also employ two RNN or LSTM as the backbone for user- and item-side recurrent networks. However, the major downside of RNN- and LSTM-backbone is two-fold. First, RNN- and LSTM-backbone cannot provide interpretable recommendation results either due to the lack of gates (RNN-backbone), or the gates modeled as multi-dimensional vectors (LSTM-backbone). In contrast, gates in IARN are represented by attention scores in scalar values, therefore IARN can provide meaningful interpretations on the relevance of each time step for recommendation. Second, RNN- or LSTM-backbone models user dynamics and item dynamics separately, thus can only learn fixed attention scores for each user and item. The attention scores for a specific user (item) actually indicate the general importance (e.g., the frequency) of each item (user) in purchased history of this user (item), which may not be effective in predicting specific user-item interactions. Unlike them, the novel attention scheme designed for IARN can learn different attention scores for an individual user (item) when interacting different items (users), thus can model the dependencies between user and item dynamics. In addition to the above, when compared with LSTM-backbone, IARN has less parameters, so is less prone to be over-fitting. Moreover, IARN uses the bi-directional recurrent network to model attention gates, which helps to maximize the utility of the input data.

Comparison with TAGM. IARN is inspired by the Temporal Attention Gated Model (TAGM) [88] recently proposed for sequence classification. IARN inherits the bi-directional at-

Table 6.1: The statistics of datasets, where #U_av_T (#I_av_T) is the average length of sequences w.r.t. users (items).

Datasets	#User	#Item	#Rating	#Feature	#U_av_T	#I_av_T
Netflix	17,043	9,598	721,017	–	36.45	64.73
MovieLens	9,737	5,121	316,891	19	31.07	59.08
Electronic	11,117	15,985	136,998	590	11.35	7.89
Home	15,745	19,383	201,660	883	11.62	9.44
Clothing	19,939	20,785	135,128	690	6.07	5.82
Sport	11,723	13,811	127,178	1,130	9.82	8.33

tention gates from TAGM, however, our attention scheme is specifically designed with the purpose of recommendation in mind. The nature of recommendation requires proper modeling user-item interactions, for which we design the Interacting Attention Module for modeling the interacting attention for both users and items. This allows IARN to capture the dependencies between user and item dynamics, making IARN particularly suitable for modeling user-item interactions.

6.4. EXPERIMENTS AND RESULTS

In this section, we conduct experiments to evaluate the performance of IARN on six real-world datasets. We aim to answer the following research questions: (1) How do the interacting attention scheme and feature encoder of IARN contribute to recommendation performance and interpretability? (2) How effective is IARN compared to state-of-the-art recommendation methods in both rating prediction and personalized ranking? They will be addressed by section 6.4.2 and section 6.4.3, respectively.

6.4.1. EXPERIMENTAL SETUP

Datasets. To evaluate the effectiveness of IARN, we utilize six real-world datasets, namely Netflix prize dataset, MovieLens, and four Amazon Web store datasets introduced by McAuley et al. [193], i.e., Electronic, Home, Clothing, Sport. Each data point in these datasets is a tuple – (user id, item id, rating, time stamp). Specifically, the Netflix dataset is a large movie rating dataset scaled from 1 to 5 with a step size of 1, which is collected between November 1999 to December 2005. MovieLens is also a personalized movie rating dataset collected from September 1995 to March 2015 with ratings ranging from 0.5 to 5.0 with a step size of 0.5. Besides, it also contains for each movie the genre information as features in a flat structure. The Amazon Web store datasets are collected from Amazon[1], which is a large on-line shopping website, including electronics, clothing, etc. The time span is from May 1996 to July 2014. In addition, there is an item category hierarchy associated with each of the four datasets. We sample the datasets such that only users and items with more than 3 ratings are preserved. Table 6.1 summarizes the statistics of all the considered datasets.

Comparison Methods. We compare with the following state-of-the-art algorithms, 1) **MF** [171]: matrix factorization as the basic latent factor model (LFM) aiming at rating

[1]https://www.amazon.com/

prediction; 2) **BPR** [172]: Bayesian personalized ranking as the basic LFM designed for item ranking; 3) **TimeSVD++** [163]: LFM with the incorporation of temporal context; 4) **HieVH** [188]: LFM integrating feature hierarchies; 5) **Item2Vec** [177]: the basic neural network (NN) model; 6) **NCF** [182]: neural collaborative filtering replacing the inner product with non-linear network layers for item ranking; 7) **MP2Vec** [194]: NN model considering auxiliary features. Note that methods designed for incorporating feature hierarchies can also handle features in a flat structure, by considering all features in the same level; similarly, methods designed for incorporating features in a flat structure can also handle feature hierarchies by flattening them into flat structures, with the loss of certain structural information.

To investigate the effect of attention-gates and our novel attention scheme, we also compare the following IARN variants using different recurrent networks as the backbone, a) **RNN-backbone**: the basic variant using RNN as the backbone of user- and item-side recurrent neural networks; b) **LSTM-backbone**: the variant using LSTM as the backbone; c) **TAGM-backbone**: the variant using TAGM as the backbone; d) **IARN-Plain**: the variant of our proposed attention-gated recurrent networks integrated with the interacting attention scheme; e) **IARN**: the upgraded version of IARN-Plain by fusing auxiliary features. Note that LSTM-backbone is similar to [167] which also employs LSTM as the backbone; while TAGM-backbone is a completely new method which is adapted from TAGM for recommendation. Given their same goal in modeling temporal dynamics, we compare them together.

Evaluation Metrics. We adopt Root Mean Square Error (RMSE) and Area Under the ROC Curve (AUC) to measure the performance of rating prediction and personalized ranking, respectively. The smaller RMSE and the larger AUC, the better the performance. We split all the datasets into training and test data according to the following time stamps: June 1st, 2005 for Netflix dataset; January 1st, 2010 for MovieLens dataset; and January 1st, 2014 for the four Amazon datasets. The data before these time stamps are treated as training data, while the rest are considered as the test data.

Parameter Settings. We empirically find out the optimal parameter settings for each comparison method. For all the methods, we set the dimension of the latent factor $d = 25$ on Netflix and MovieLens datasets, and $d = 50$ on the four Amazon datasets. We apply a grid search in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ for the learning rate and regularization coefficient. For TimeSVD++, $decay\_rate = 0.4; bin = 30$. For HieVH, $\alpha = 0.01$. For MP2Vec, $\alpha = 0.1$. For all recurrent networks mentioned in this work (RNN-backbone, LSTM-backbone, TAGM-backbone, IARN) as well as NCF, the number of hidden units is set to 64 which is selected as the best configuration from the option set {32, 64, 128} based on a held-out validation set. To avoid potential over-fitting, the dropout value is validated from the option set {0.00, 0.25, 0.50}. Model training is performed using a RMSprop stochastic gradient descent optimization algorithm with mini-batches of 50 pairs of user-item interactions. All the gradients are clipped between -10 and 10 to prevent exploding [65].

6.4.2. EFFECTS OF ATTENTION AND FEATURE ENCODER

Attention. In order to investigate the impact of the proposed attention scheme, we compare the performance (measured by RMSE) of IARN-Plain with different recurrent net-

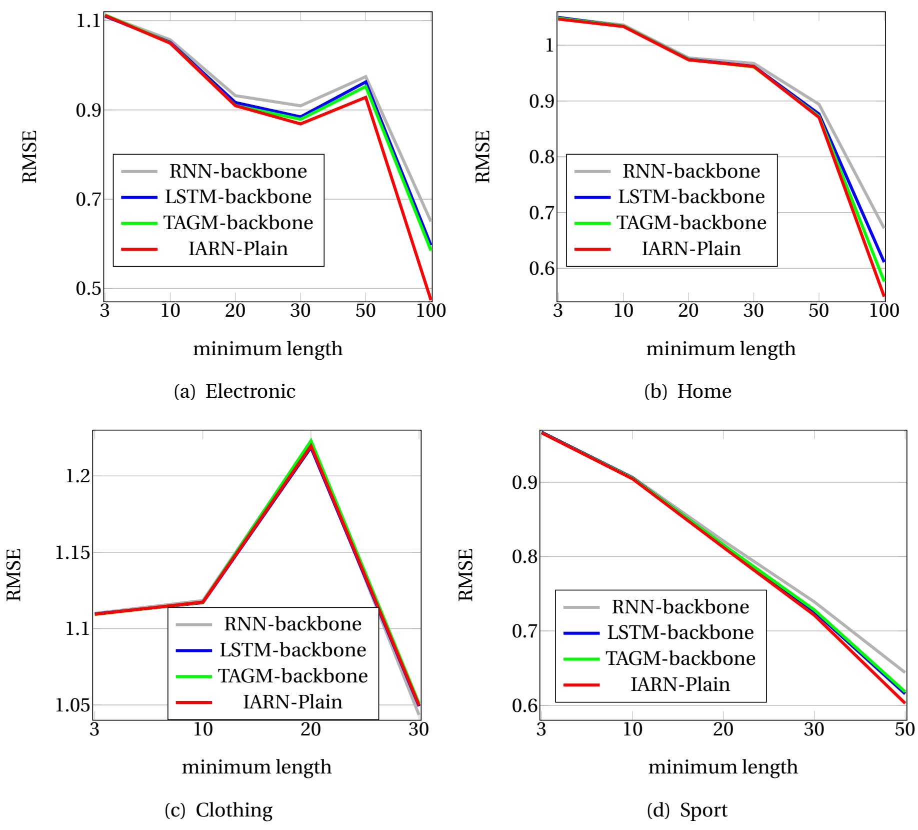


Figure 6.3: Performance (measured by RMSE) of IARN variants with different recurrent networks as the backbone on different configurations of the real-world datasets with varying minimum sequence lengths.

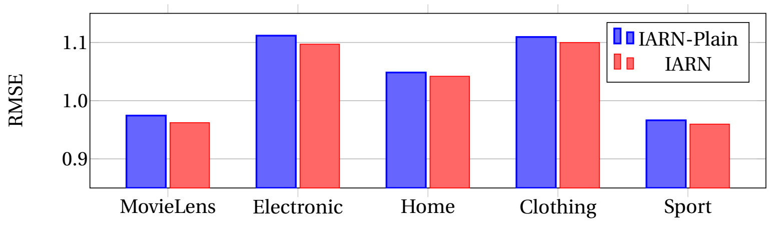


Figure 6.4: Performance of rating prediction of IARN variants with and without feature encoder on the five datasets.

works as the backbone, including RNN-backbone, LSTM-backbone, and TAGM-backbone. To understand their capability in modeling temporal dynamics of users and item history in different lengths, we test their performance on different configurations of the datasets by constraining the minimum length of user and item input sequences. A grid search in $\{3, 10, 20, 30, 50, 100\}$ is applied for the minimum length of sequences on all the datasets, excluding Clothing and Sport since there are few users possessing long length sequences in these two datasets. Due to space limitation, we only show the results on four Amazon datasets as depicted by Figure 6.3, however similar observations as below can be obtained on all the datasets.

As the minimum length of input sequences increases, the performance of all methods generally improves, indicating that sufficient temporal context could ensure recurrent network based methods to better model the dynamics of user preferences. The performance of gated recurrent networks, i.e., LSTM-backbone, TAGM-backbone, and IARN-plain, is generally better than the non-gated recurrent network, i.e., RNN-backbone. Such a difference is minor when the minimum sequence length is less than a threshold (e.g., 30), and becomes significant with the further growth of the sequence length. This shows the benefit of gating mechanism in effectively preserving historical information deep into the past for recommendation. The observation further explains the close performance of different methods on Clothing dataset, whose sequences are mostly less than 30 and the average sequence length (i.e., around 6, Table 6.1) is significantly smaller than all the other datasets.

The overall performance of TAGM-backbone and IARN-Plain, is better than that of LSTM-backbone. LSTM-backbone adopts multi-dimensional gates w.r.t. each hidden unit, which can be more easily over-fitted than the (bi-directional) attention-gates employed by TAGM-backbone and IARN-Plain. With respect to attention-gated recurrent networks, IARN-Plain outperforms TAGM-backbone across all different configurations of minimum sequence length. This is mainly due to the fact that TAGM-backbone learns user and item dynamics separately, i.e., only fixed attention scores for user history and item history are learned (LSTM-backbone suffers from the same issue). Whereas equipped with our novel attention scheme, IARN-Plain can adaptively learn different attention scores for user (item) history when the user (item) interacts with different items (users). Such a comparison clearly shows the advantage of our proposed attention scheme for modeling the dependencies between user and item dynamics.

Overall, IARN-Plain achieves the best performance across all different configurations of all the datasets, especially when the sequence length gets larger. On average, the relative improvements w.r.t. the second best method are 2.54% with minimum length = 50 and 11.65% with minimum length = 100. This implies the remarkable advantage of IARN-Plain in dealing with long sequences.

Feature Encoder. We further examine the effectiveness of auxiliary features which are organized in either a flat or hierarchical structure on all the datasets, excluding Netflix which does not contain any auxiliary features. The results are given by Figure 6.4. By integrating auxiliary features IARN outperforms IARN-Plain across all the datasets, with 1.19% lift (p-value < 0.01) in RMSE on average. This clearly indicates the benefit of considering feature encoder in our proposed IARN approach.

Interpretation by IARN. The attention scores learned by IARN for individual time steps in user and item history can help quantify the relevance of each time steps in user and item history for recommendation. We now qualitatively analyze such attention scores to investigate their effects on providing meaningful interpretations for recommendation. Figure 6.5 shows the attention scores learned by IARN on examples of four datasets.

In each of the four examples, we can observe varying attention scores assigned to

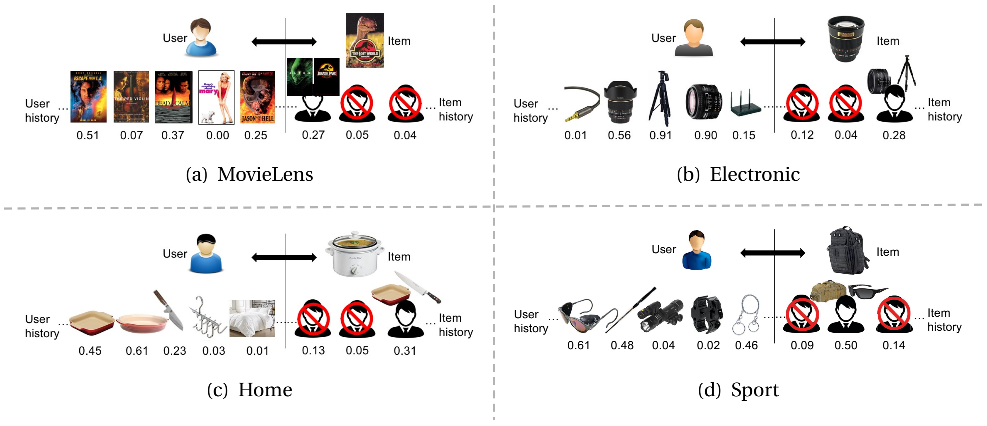


Figure 6.5: Examples of attention scores learned by IARN. The target user and item in each sub-figure have an observed interaction. The attention scores of individual time steps are shown under user and item history.

different time steps in both user and item history. Such attention scores can effectively capture the target user's preference related to the inherent property of the target item, as inferred from the data. For example in MovieLens dataset, IARN learns high attention scores for the time steps in user history when the user was watching movies of genre "Adventure" and "Action". These time steps are highly indicative of his potential preference over the target item, i.e., "The Lost World: Jurassic Park". In contrast, low attention scores are assigned to those time steps when he was watching movies of other genres, e.g., "Drama". IARN thus can selectively memorize most relevant time steps of the user's history in predicting his preference over the target item. Similarly, IARN can also select the most relevant time steps in item history to characterize the inherent genre of the item, i.e., those time steps when it was being watched by users who share the same interest as the target user, i.e., "Adventure" and "Action" movies (e.g., "Aliens"). Similar observations can be noted in the other three examples. For instance in the Sport dataset, IARN can infer the most relevant time steps in the user history when the user bought hiking related item; and in the item history when the item was bought by users who like hiking. Such dependency between the relevance of time steps in user history and in item history is highly useful for discovering the link between the target user and item, and thus provides strong interpretations for the recommendation results.

6.4.3. Comparative Results

Rating Prediction. The left side of Table 6.2 presents the rating prediction performance on the six real-world datasets. BPR, Item2Vec, NCF, and MP2Vec are excluded since RMSE cannot be applied to these methods. BPR and NCF optimize ranking based objective function. Item2Vec and MP2Vec learn the embeddings of items and then adopt the similarity score between item embeddings to predict recommendations, instead of minimizing the difference between the real ratings and the estimated ratings. Several interesting observations can be obtained.

Table 6.2: Performance of rating prediction (measured by RMSE) and personalized ranking (measured by AUC) of all comparison methods on the six real-world datasets. The best performance is boldfaced; the runner up is labeled with "*". The results of HieVH and MP2Vec on Netflix is not available (marked by "–") due to the lack of feature information in the Netflix dataset.

Datasets	Rating prediction: RMSE				Personalized ranking: AUC							
	MF	**TimeSVD++**	**HieVH**	**IARN**	**MF**	**BPR**	**TimeSVD++**	**HieVH**	**Item2Vec**	**NCF**	**MP2Vec**	**IARN**
Netflix	1.1828	1.1243*	–	**1.0582**	0.6147	0.7414	0.6289	–	0.7642	0.7654*	–	**0.7901**
MovieLens	1.1319	1.0623	1.0326*	**0.9622**	0.6305	0.6971	0.6504	**0.7214**	0.7130	0.7128	0.7135*	0.7135*
Electronic	1.3213	1.3152	1.1694*	**1.0970**	0.5757	0.6699	0.5820	0.7257*	0.6794	0.7052	0.7072	**0.7359**
Home	1.2010	1.1974	1.1231*	**1.0419**	0.5305	0.6341	0.5520	0.7132*	0.6902	0.6973	0.7007	**0.7210**
Clothing	1.3587	1.2851	1.2283*	**1.0998**	0.5092	0.6246	0.5205	**0.7011**	0.6717	0.6720	0.6919	0.7004*
Sport	1.2021	1.1690	1.1279*	**0.9597**	0.5489	0.6492	0.5515	0.6962*	0.6787	0.6759	0.6792	**0.6975**

It is unsurprising that MF – as the basic LFM – considering no auxiliary information, performs the worst among all the considered methods. By integrating temporal context into LFM, TimeSVD++ outperforms MF. This confirms that modeling temporal dynamics of user preferences can significantly improve the recommendation performance. HieVH is also a LFM based approach, which takes into account the influence of both vertical and horizontal dimensions of feature hierarchies on recommendation. It outperforms MF, and even slightly exceeds TimeSVD++, confirming the effectiveness of auxiliary features for better recommendation.

Our proposed approach – IARN, consistently outperforms the other methods in the comparison pool, with an average performance gain (w.r.t. the second best method) of 8.58% on RMSE. Pair-wised t-test demonstrates that the improvements of IARN on all the datasets are significant ($p-$value< 0.01). Such big enhancement clearly shows the effectiveness of the integration of interacting attention scores as well as auxiliary features.

Ranking Performance. We further evaluate the ranking quality of items recommended by the methods in the comparison pool. Results are shown on the right side of Table 6.2. A number of meaningful findings can be noted from the table.

In terms of the LFM based methods, TimeSVD++ and HieVH outperform MF by taking temporal context and feature hierarchies into account, respectively. This observation further verifies the usefulness of the two types of side information for better recommendations. For NN based method, the fact that the performance of MP2Vec is better than that of Item2Vec and NCF also helps to reach the same conclusion, as MP2Vec considers auxiliary features while Item2Vec and NCF do not. The superior performance of NCF over Item2Vec shows the effectiveness of hidden layers in neural networks for modeling non-linear user-item interactions. In both LFM based methods and NN based methods, those specifically designed for personalized ranking, i.e., BPR and NCF, perform better than methods for rating prediction, i.e., MF and Item2Vec, which strongly confirms the conclusion that methods designed for personalized ranking are more efficient than rating prediction methods for the item ranking problem [172].

Our proposed approach IARN generally achieves the best performance on item ranking when compared with the other considered methods. This demonstrates the effectiveness of IARN in modeling user and item dynamics for improving recommendation performance. However, the performance improvements of IARN on ranking prediction is far behind those on rating prediction. The underlying explanation is that the objective function of IARN aims to minimize the squared error between the observed ratings and the estimated ratings, which is just in accordance with the definition of RMSE. IARN is therefore more effective on rating prediction. We leave it as future work the improvement of IARN on item ranking.

6.5. CONCLUSIONS

User preferences often evolve over time, thus modeling their temporal dynamics is essential for recommendation. This chapter proposes the Interacting Attention-gated Recurrent Network (IARN) to accommodate temporal context for better recommendation. IARN can not only accurately measure the relevance of individual time steps of user and item history for recommendation, but also capture the dependencies between user and item dynamics in shaping user-item interactions. We further show that IARN can be easily integrated with auxiliary features for enhanced recommendation performance. Extensive validation on six real-world datasets demonstrates the superiority of IARN against other state-of-the-art methods. For future work, we intend to further improve the effectiveness of IARN on the item ranking problem.

7

DISCUSSION

In this chapter, we first conclude this thesis and then foresee several research topics for future work.

7.1. CONCLUSIONS

This thesis focuses on the supervised learning on sequence data. In particular, we have made advances by proposing models to answer the five research questions posed in Chapter 1.

Research Question 1: *Can we propose a sequence classification model which is able to model complex decision boundaries with a limited number of latent variables?*

We proposed a sequence classification model called *Hidden-Unit Logistic Model (HULM)* in Chapter 2. The model is similar in structure to the popular hidden CRF model, but it employs binary stochastic hidden units instead of multinomial hidden units between the data and label. As a result, the HULM can model exponentially more latent states than a hidden CRF with the same number of parameters. In addition, our HULM is also more computationally efficient than the HCRF model given the same number of hidden variables. The results of experiments on several real-world data sets showed that this advantage results in improved performance on challenging time-series classification tasks.

One potential drawback of our model is that the modeling of temporal dependencies is somewhat limited by the linear-chain connection between adjacent time steps.

Research Question 2: *Can we propose a sequence classification model which is able to deal with unsegmented sequences and meanwhile measure the relevance of each time step of the input sequence to the classification task?*

To address this question, we presented the *Temporal Attention-Gated Model (TAGM)* in Chapter 3, which is able to deal with unsegmented sequences. The model is inspired by attention models and gated recurrent networks and is able to detect salient parts of the sequence while ignoring irrelevant and noisy ones. The resulting hidden representation suffers less from the effect of noise and and thus leads to better performance. Furthermore, the learned attention scores provide a physically meaningful interpretation of relevance of each time step observation for the final decision. We showed the generalization of our approach on three, very different, datasets and sequence classification tasks.

The attention scores are learned per frame, which is able to accurately identify the relevance of each frame. However, the learned attention scores do not proceed continuously and smoothly, which prevent them to be used for sequence segmentation or detection.

Research Question 3: *Can we propose a one-stage supervised sequence model which learns features together with the supervised learning (classification or regression)?*

We presented an attended end-to-end model for age estimation from facial expression videos in Chapter 4, which is a one-stage supervised learning model that learns features for each time step together with the supervised learning. The model employs convolutional networks to learn the effective appearance features and feed them into recurrent networks to learn the temporal dynamics. Furthermore, both a spatial attention mechanism and a temporal attention mechanism are added to the model. The spatial attention can be integrated seamlessly into the convolutional layers to capture the salient facial regions in each single image, while the temporal attention is incorporated in recurrent networks to capture the salient temporal frames. The whole model can be trained readily in an end-to-end manner. Specifically, our model makes a substantial improvement over the state-of-the-art methods.

Research Question 4: *Can we propose a sequence model that learns a good similarity measure in a supervised way?*

As an attempt of modeling symmetric association, we presented a supervised learning model for similarity scoring between two sequences. Our proposed model, named *Siamese Recurrent Networks (SRNs)*, combines the idea from metric learning and deep recurrent networks. It is particularly effective compared to alternative techniques in settings in which the similarity function that needs to be learned is complicated, or when the number of labeled time series for some of the classes of interest is limited. When a reasonably large collection of examples of similar and dissimilar time series is available to train the models, the siamese recurrent networks can produce representations that are suitable for challenging problems such as one-shot learning or extreme classification of time series.

Research Question 5: *Given two input sequences representing a pair of historic user and item data, how to model the preference of the user over the item, which takes into account not only the information contained in each individual sequence, but also the interdependencies between them?*

The proposed model in Chapter 6, named *Interacting Attention-gated Recurrent Network (IARN)*, is designed to learn the user preference over the item in recommendation systems. *IARN* can not only accurately measure the relevance of individual time steps of user and item history for recommendation, but also capture the dependencies between user and item dynamics in shaping user-item interactions. We further showed that IARN can be easily integrated with auxiliary features for enhanced recommendation performance. Extensive validation on six real-world datasets demonstrated the superiority of IARN against other state-of-the-art methods.

7.2. FUTURE WORK

While the research presented in this thesis has gained new insights, we foresee several future research directions in which the work in this thesis can be extended.

7.2.1. SEQUENCE CLASSIFICATION

Sequence Classification by Extending the *HULM* model. Our proposed *Hidden-Unit Logistic Model (HULM)* for sequence classification employs simple first-order Markov chains on the hidden units to model the temporal dependencies between different time steps. One promising future work is to study the variants of the *HULM* model by replacing the first-order chains with more powerful, higher-order chains via a similar factorization as used in neural autoregressive distribution estimators [46]. The resulting models will likely have longer temporal memory than our current model, which will likely lead to stronger performance on complex time-series classification tasks.

A second interesting direction might be to remove the linearly connected Markov chains completely and model the temporal relationships as fully-connected chains between adjacent time steps, which is more powerful to model the temporal dependencies. However, the resulting model has no analytical formulation. A potential option to tackle the difficulty is to employ variational inference methods [195] to find an approximated solution.

Sequence Classification by CNN-based models. Convolutional Neural Networks (CNNs) led to great progress in image domain such as object detection [196, 197] and image classification [198]. Recently, CNNs have been shown to be powerful for sequence data too in the case of the neural machine translation [199]. Specifically, the temporal information contained in the sequence can be modeled well by convolutional operations. Inspired by this fact, a promising future direction for sequence classification is to leverage the Convolutional Neural Networks to model both, the static information in each single time step, and the temporal information of the sequence. It might be especially beneficial for video data due to the success of CNNs in image domain.

Sequence Classification by densely connected RNNs. Recurrent Neural Networks (RNNs) have been widely applied to language [134], age estimation from videos [200], image generation [61] and recommendation system [201]. One important limitation of a standard RNNs is that it is difficult to store long-term information due to the vanishing gradient problem caused by the Markov connections between adjacent time steps. Similar limitation about the vanishing gradient also exists in the deep CNNs. An effective way to address this problem in the case of CNNs is to introduce the shorter or high-order connections between nonadjacent layers, such as Resnet [198] or dense connected CNNs [202]. Hence, it might be interesting to introduce the shorter or densely connection into RNNs to maintain a long-term memory since RNNs can be viewed as deep feed-forward networks in which all layers share the same weights.

7.2.2. MULTI-TASK LEARNING ON SEQUENCE DATA

Multi-task learning scenarios are common among sequence applications. For instance, recognition of short-term actions and long-term activities, or optical character recognition and word classification, can be performed simultaneously. This can potentially improve the learning performance since what is learned for one task can be beneficial for other tasks. Hence, multi-task learning on sequence data is a promising research direction that is worth to explore in the future.

7.2.3. SIMILARITY MEASUREMENT

The *Siamese Recurrent Networks (SRNs)* proposed in Chapter 5 employ a weighted inner product between the hidden representations of two sequences which is learned by recurrent networks to represent the similarity between the two sequences. Recently, Deep Canonical Correlation Analysis [203] (DCCA) was proposed to learn complex nonlinear transformations of a pair of objects such that the resulting representations are maximally correlated. The method employs two deep networks to learn a hidden representation for each of object. The parameters are trained to maximize the correlation between the two hidden representations using gradient-based optimization. It can be considered as a nonlinear extension of the linear Canonical Correlation Analysis. Interestingly, DCCA can be applied to a pair of sequence data (eg., using CNNs or RNNs) to learn the hidden representations for two sequences by maximizing the correlation between them.

Another direction is to apply triplet loss [204] to the task of similarity measurement. The main idea of triplet loss is to train the model so as to ensure a predefined margin between the distance of a similar pair of sequences and a dissimilar pair of sequences. It is interesting to compare this scheme to the binary cross-entropy loss used in our model.

7.2.4. UNSUPERVISED LEARNING ON SEQUENCE DATA

This thesis focuses on the supervised learning on sequence data. However, it is worth to explore the unsupervised learning on sequence data. First, this might be used to learn a fixed-length hidden representations for sequences. Nowadays, a popular approach is to apply an autoencoder to sequence data. Having, a fixed length representation for variable length sequences, standard learners can be used.

Another possibility is to employ Generative Adversarial Nets (GAN) [205] , which enable to generate new sequence data to enlarge the training set. To adapt GANs to sequence data, one might use RNNs as backbone model rather than multilayer perceptrons or deep CNNs. Another option is to assimilate the idea of Variational Autoencoder [206], which is a generative model that encourages the learned latent vectors to follow a gaussian distribution. How to apply VAE to sequence data is a nontrivial task that is worth to investigate.

As a final remark, we have shown that building predictive models on sequence data is challenging. We are, however, confronted more and more with sequence data, so we are in a need for new approaches that deal with these challenges. With this thesis, we have made a step in that direction by contributing to different aspects of supervised learning on sequence data. Nevertheless, more research is still necessary.

REFERENCES

[1] J. Ortega-Garcia, J. Fierrez-Aguilar, D. Simon, J. Gonzalez, M. Faundez-Zanuy, V. Espinosa, A. Satue, I. Hernaez, J.-J. Igarza, C. Vivaracho, D. Escudero, and Q.-I. Moro, *MCYT baseline corpus: A bimodal biometric database,* IEE Proceedings on Vision, Image and Signal Processing **150** (2003).

[2] L. Rabiner, *A tutorial on hidden markov models and selected applications in speech recognition,* Proceedings of the IEEE **77**, 257 (1989).

[3] J. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labelling sequence data,* in *ICML* (2001) pp. 282–289.

[4] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, *Hidden conditional random fields,* IEEE Trans. Pattern Anal. Mach. Intell. **29**, 1848 (2007).

[5] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell, *Hidden conditional random fields for gesture recognition,* in *CVPR*, Vol. 2 (2006) pp. 1521–1527.

[6] Y. Wang and G. Mori, *Learning a discriminative hidden part model for human action recognition,* NIPS **21** (2008).

[7] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches,* in *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation* (2014) pp. 103–111.

[8] S. Hochreiter and J. Schmidhuber, *Long short-term memory,* Neural Comput. **9**, 1735 (1997).

[9] A. Graves, A.-r. Mohamed, and G. Hinton, *Speech recognition with deep recurrent neural networks,* in *ICASSP* (2013).

[10] M. Müller, *Information Retrieval for Music and Motion* (Springer-Verlag New York, Inc., 2007).

[11] T. Vintsyuk, *Speech discrimination by dynamic programming,* Kibernetika **4**, 81 (1968).

[12] T. Jaakkola and D. Haussler, *Exploiting generative models in discriminative classifiers,* in *In Advances in Neural Information Processing Systems 11* (1998) pp. 487–493.

[13] F. Perronnin, J. Sánchez, and T. Mensink, *Improving the Fisher kernel for large-scale image classification,* in *Proceedings of the European Conference on Computer Vision* (2010) pp. 143–156.

[14] H. Larochelle and Y. Bengio, *Classification using discriminative restricted boltzmann machines,* in *ICML* (2008) pp. 536–543.

[15] L. van der Maaten, M. Welling, and L. Saul, *Hidden-unit conditional random fields,* Int. Conf. on Artificial Intelligence & Statistics , 479 (2011).

[16] L. A. Jeni, A. Lőrincz, Z. Szabó, J. F. Cohn, and T. Kanade, *Spatio-temporal event classification using time-series kernel based structured sparsity,* in *ECCV* (2014) pp. 135–150.

[17] T. Jaakkola, M. Diekhans, and D. Haussler, *A discriminative framework for detecting remot protein homologies,* Journal of Computational Biology **7**, 95 (2000).

[18] L. van der Maaten, *Learning discriminative Fisher kernels,* in *Proceedings of the International Conference on Machine Learning* (2011) pp. 217–224.

[19] T. Jebara, R. Kondor, and A. Howard, *Probability product kernels,* Journal of Machine Learning Research **5**, 819 (2004).

[20] L.-P. Morency, A. Quattoni, and T. Darrell, *Latent-dynamic discriminative models for continuous gesture recognition,* in *CVPR* (2007).

[21] J. Peng, L. Bo, and J. Xu, *Conditional neural fields,* in *Advances in Neural Information Processing Systems 22,* edited by Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (2009) pp. 1419–1427.

[22] T.-M.-T. Do and T. Artieres, *Neural conditional random fields,* in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics,* Vol. 9 (JMLR: W&CP, 2010).

[23] K. Bousmalis, S. Zafeiriou, L.-P. Morency, and M. Pantic, *Infinite hidden conditional random fields for human behavior analysis,* IEEE Transactions on Neural Networks and Learning Systems **24**, 170 (2013).

[24] K. Bousmalis, S. Zafeiriou, L.-P. Morency, M. Pantic, and Z. Ghahramani, *Variational hidden conditional random fields with coupled dirichlet process mixtures,* in *ECML PKDD* (2013) pp. 531–547.

[25] G. E. Hinton, *Training products of experts by minimizing contrastive divergence,* Neural Comput. **14**, 1771 (2002).

[26] M. Welling, M. Rosen-Zvi, and G. Hinton, *Exponential family harmoniums with an application to information retrieval,* in *Advances in Neural Information Processing Systems,* Vol. 17 (2004).

[27] S. Ray and D. Ren, *On the upper bound of the number of modes of a multivariate normal mixture,* Journal of Multivariate Analysis **108**, 41 (2012).

[28] G. Montufar and J. Morton, *When does a mixture of products contain a product of mixtures?* SIAM Journal on Discrete Mathematics **29**, 321 (2015).

[29] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, *The ucr time series classification archive,* (2015), `www.cs.ucr.edu/~eamonn/time_series_data/`.

[30] B. Williams, M. Toussaint, and A. Storkey, *Modelling motion primitives and their timing in biologically executed movements,* in *NIPS* (2008) pp. 1609–1616.

[31] N. Hammami and M. Bedda, *Improved tree model for Arabic speech recognition,* in *Int. Conf. on Computer Science and Information Technology* (2010) pp. 521–526.

[32] P. Lucey, J. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, *The extended Cohn-Kanade dataset (CK+): A complete dataset for action unit and emotion-specified expression,* in *CVPR Workshops* (2010) pp. 94–101.

[33] W. Li, Z. Zhang, and Z. Liu, *Action recognition based on a bag of 3d points,* in *CVPR* (2010).

[34] J. Wang, Z. Liu, Y. Wu, and J. Yuan, *Mining actionlet ensemble for action recognition with depth cameras,* in *CVPR* (2012).

[35] L. van der Maaten and E. Hendriks, *Action unit classification using active appearance models and conditional random fields,* Cognitive Processing **13**, 507 (2012).

[36] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, *Real-time human pose recognition in parts from single depth images,* in *CVPR* (2011).

[37] K. Bousmalis, *Hidden conditional random fields implementation,* Http://www.doc.ic.ac.uk/ kb709/software.shtml.

[38] P.-F. Marteau and S. Gibet, *On recursive edit distance kernels with application to time series classification,* IEEE Transactions on Neural Networks and Learning Systems **26**, 1121 (2015).

[39] P. Marteau, S. Gibet, and C. Reverdy, *Down-sampling coupled to elastic kernel machines for efficient recognition of isolated gestures,* in *International Conference on Pattern Recognition (ICPR)* (2014) pp. 363–368.

[40] S. Koelstra, M. Pantic, and I. Patras, *A dynamic texture-based approach to recognition of facial actions and their temporal models,* IEEE Trans. on PAMI **32**, 1940 (2010).

[41] M. F. Valstar and M. Pantic, *Fully automatic recognition of the temporal phases of facial actions,* IEEE Trans. on SMC, Part B: Cybernetics **42**, 28 (2012).

[42] Y. Li, J. Chen, Y. Zhao, and Q. Ji, *Data-free prior model for facial action unit recognition,* IEEE Trans. on Affective Computing **4**, 127 (2013).

[43] Y. Li, S. Wang, Y. Zhao, and Q. Ji, *Simultaneous facial feature tracking and facial expression recognition,* IEEE Trans. on Image Processing **22**, 2559 (2013).

[44] X. Ding, V. Chu, F. De la Torre, J. F. Cohn, and Q. Wang, *Facial action unit detection by cascade of tasks,* in *ICCV* (2013).

[45] X. Zhang, M. H. Mahoor, S. M. Mavadati, and J. F. Cohn, *A l_p-norm MTMKL framework for simultaneous detection of multiple facial action units,* in *IEEE Winter Conf. on Applications of Computer Vision* (2014) pp. 1104–1111.

[46] H. Larochelle and I. Murray, *The neural autoregressive distribution estimator,* Journal of Machine Learning Research **15**, 29 (2011).

[47] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, *Large-scale video classification with convolutional neural networks,* in *CVPR* (2014).

[48] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate,* in *International Conference on Learning Representations* (2015).

[49] W. Pei, H. Dibeklioğlu, D. M. J. Tax, and L. van der Maaten, *Multivariate time-series classification using the hidden-unit logistic model,* IEEE Transactions on Neural Networks and Learning Systems (2017).

[50] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, *Show, attend and tell: Neural image caption generation with visual attention,* in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15),* edited by D. Blei and F. Bach (JMLR Workshop and Conference Proceedings, 2015) pp. 2048–2057.

[51] X. Chen and C. L. Zitnick, *Mind's eye: A recurrent visual representation for image caption generation.* in *CVPR* (IEEE Computer Society, 2015) pp. 2422–2431.

[52] H. Fang, S. Gupta, F. N. Iandola, R. K. Srivastava, L. Deng, P. Dollár, J. Gao, X. He, M. Mitchell, J. C. Platt, C. L. Zitnick, and G. Zweig, *From captions to visual concepts and back.* in *CVPR* (IEEE Computer Society, 2015) pp. 1473–1482.

[53] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, *Describing videos by exploiting temporal structure,* in *Computer Vision (ICCV), 2015 IEEE International Conference on* (IEEE, 2015).

[54] M.-T. Luong, H. Pham, and C. D. Manning, *Effective approaches to attention-based neural machine translation,* in *EMNLP* (2015).

[55] B. Sankaran, H. Mi, Y. Al-Onaizan, and A. Ittycheriah, *Temporal attention model for neural machine translation,* CoRR **abs/1608.02927** (2016).

[56] A. Haque, A. Alahi, and L. Fei-Fei, *Recurrent attention models for depth-based person identification,* in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[57] A. Graves, S. Fernández, and F. Gomez, *Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,* in *In Proceedings of the International Conference on Machine Learning, ICML 2006* (2006) pp. 369–376.

[58] S. Sharma, R. Kiros, and R. Salakhutdinov, *Action recognition using visual attention,* CoRR **abs/1511.04119** (2015).

[59] J. SCHMIDHUBER, *A local learning algorithm for dynamic feedforward and recurrent networks,* Connection Science **1**, 403 (1989).

[60] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur, *Extensions of recurrent neural network language model.* in *ICASSP* (IEEE, 2011) pp. 5528–5531.

[61] L. Theis and M. Bethge, *Generative image modeling using spatial lstms,* in *Proceedings of the 28th International Conference on Neural Information Processing Systems,* NIPS'15 (2015) pp. 1927–1935.

[62] A. Graves, *Generating sequences with recurrent neural networks,* in *arXiv:1308.0850* (2013).

[63] V. Nair and G. E. Hinton, *Rectified linear units improve restricted boltzmann machines,* in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (2010).

[64] P. J. Werbos, *Generalization of backpropagation with application to a recurrent gas market model,* Neural Networks **1** (1988).

[65] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, *Advances in optimizing recurrent networks,* in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing* (2013).

[66] B. Pang and L. Lee, *A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts,* in *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics,* ACL '04 (2004).

[67] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank,* in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Association for Computational Linguistics, Stroudsburg, PA, 2013) pp. 1631–1642.

[68] J. Pennington, R. Socher, and C. D. Manning, *Glove: Global vectors for word representation,* in *Empirical Methods in Natural Language Processing (EMNLP)* (2014) pp. 1532–1543.

[69] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, *Deep unordered composition rivals syntactic methods for text classification,* in *Association for Computational Linguistics* (2015).

[70] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, *Semi-supervised recursive autoencoders for predicting sentiment distributions,* in *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* EMNLP '11 (Association for Computational Linguistics, Stroudsburg, PA, USA, 2011) pp. 151–161.

[71] O. İrsoy and C. Cardie, *Deep recursive neural networks for compositionality in language,* in *Advances in Neural Information Processing Systems 27* (Curran Associates, Inc., 2014).

[72] K. S. Tai, R. Socher, and C. D. Manning, *Improved semantic representations from tree-structured long short-term memory networks,* in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers* (2015) pp. 1556–1566.

[73] Y. Kim, *Convolutional neural networks for sentence classification,* in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, 2014) pp. 1746–1751.

[74] Q. Le and T. Mikolov, *Distributed representations of sentences and documents,* in *Proceedings of the 31st International Conference on Machine Learning (ICML-14),* edited by T. Jebara and E. P. Xing (JMLR Workshop and Conference Proceedings, 2014) pp. 1188–1196.

[75] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui, *Consumer video understanding: A benchmark database and an evaluation of human and machine performance,* in *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR)* (2011).

[76] Y.-G. Jiang, Q. Dai, T. Mei, Y. Rui, and S.-F. Chang, *Super fast event recognition in internet videos,* IEEE Transactions on Multimedia **17**, 1 (2015).

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks,* in *Advances in Neural Information Processing Systems 25,* edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (2012) pp. 1097–1105.

[78] G. Guo, Y. Fu, C. R. Dyer, and T. S. Huang, *Image-based human age estimation by manifold learning and locally adjusted robust regression,* IEEE Transactions on Image Processing **17**, 1178 (2008).

[79] A. Lanitis, C. Draganova, and C. Christodoulou, *Comparing different classifiers for automatic age estimation,* IEEE Transactions on Systems, Man, and Cybernetics-Part B **34**, 621 (2004).

[80] A. Albert, K. Ricanek, and E. Patterson, *A review of the literature on the aging adult skull and face: Implications for forensic science research and applications,* Forensic science international **172**, 1 (2007).

[81] Y. Fu, G. Guo, and T. S. Huang, *Age synthesis and estimation via faces: A survey,* IEEE Transactions on Pattern Analysis and Machine Intelligence **32**, 1955 (2010).

[82] A. Gallagher and T. Chen, *Understanding images of groups of people,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2009).

[83] A. Gallagher and T. Chen, *Estimating age, gender and identity using first name priors,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[84] H. Han, C. Otto, and A. K. Jain, *Age estimation from face images: Human vs. machine performance,* in *IAPR International Conference on Biometrics* (2013).

[85] E. Agustsson, R. Timofte, S. Escalera, X. Baró, I. Guyon, and R. Rothe, *Apparent and real age estimation in still images with deep residual regressors on APPA-REAL database,* in *IEEE International Conference on Automatic Face & Gesture Recognition* (2017) pp. 87–94.

[86] H. Dibeklioğlu, T. Gevers, A. A. Salah, and R. Valenti, *A smile can reveal your age: Enabling facial dynamics in age estimation,* in *ACM International Conference on Multimedia* (2012).

[87] H. Dibeklioğlu, F. Alnajar, A. Salah, and T. Gevers, *Combining facial dynamics with appearance for age estimation,* IEEE Transactions on Image Processing **24**, 1928 (2015).

[88] W. Pei, T. Baltrusaitis, D. M. Tax, and L.-P. Morency, *Temporal attention-gated model for robust sequence classification,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[89] T. R. Alley, *Social and applied aspects of perceiving faces* (Psychology Press, 2013).

[90] A. Lanitis, C. J. Taylor, and T. F. Cootes, *Toward automatic simulation of aging effects on face images,* IEEE Transactions on Pattern Analysis and Machine Intelligence **24**, 442 (2002).

[91] T. Ojala, M. Pietikäinen, and T. Maenpaa, *Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, Pattern Analysis and Machine Intelligence, IEEE Transactions on,* **24**, 971 (2002).

[92] J. Ylioinas, A. Hadid, X. Hong, and M. Pietikäinen, *Age estimation using local binary pattern kernel density estimate,* in *International Conference on Image Analysis and Processing* (2013) pp. 141–150.

[93] G. Guo, G. Mu, Y. Fu, and T. S. Huang, *Human age estimation using bio-inspired features,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2009) pp. 112–119.

[94] P. Thukral, K. Mitra, and R. Chellappa, *A hierarchical approach for human age estimation,* in *IEEE International Conference on Acoustics, Speech and Signal Processing* (2012) pp. 1529 – 1532.

[95] X. Geng, Z.-H. Zhou, Y. Zhang, G. Li, and H. Dai, *Learning from facial aging patterns for automatic age estimation,* in *ACM International Conference on Multimedia* (2006) pp. 307–316.

[96] X. Geng, K. Smith-Miles, and Z. Zhou, *Facial age estimation by nonlinear aging pattern subspace,* in *ACM International Conference on Multimedia* (2008) pp. 721–724.

[97] C. Zhan, W. Li, and P. Ogunbona, *Age estimation based on extended non-negative matrix factorization,* in *IEEE International Workshop on Multimedia Signal Processing* (2011).

[98] Y. Chen and C. Hsu, *Subspace learning for facial age estimation via pairwise age ranking,* IEEE Transactions on Information Forensics and Security **8**, 2164 (2013).

[99] F. Alnajar, C. Shan, T. Gevers, and J.-M. Geusebroek, *Learning-based encoding with soft assignment for age estimation under unconstrained imaging conditions,* Image and Vision Computing **30**, 946 – (2012).

[100] W.-L. Chao, J.-Z. Liu, and J.-J. Ding, *Facial age estimation based on label-sensitive learning and age-oriented regression,* Pattern Recognition **46**, 628 (2013).

[101] C. Li, Q. Liu, J. Liu, and H. Lu, *Ordinal distance metric learning for image ranking,* IEEE Transactions on Neural Networks and Learning Systems **26**, 1551 (2015).

[102] X. Wang, R. Guo, and C. Kambhamettu, *Deeply-learned feature for age estimation,* in *IEEE Winter Conference on Applications of Computer Vision* (2015) pp. 534–541.

[103] G. Levi and T. Hassner, *Age and gender classification using convolutional neural networks,* in *IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2015) pp. 34–42.

[104] F. Gurpinar, H. Kaya, H. Dibeklioğlu, and A. Salah, *Kernel ELM and CNN based facial age estimation,* in *IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2016) pp. 80–86.

[105] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, *Deep face recognition.* in *British Machine Vision Conference*, Vol. 1 (2015) p. 6.

[106] R. Rothe, R. Timofte, and L. Van Gool, *Deep expectation of real and apparent age from a single image without facial landmarks,* International Journal of Computer Vision , 1 (2016).

[107] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition,* arXiv preprint arXiv:1409.1556 (2014).

[108] E. Agustsson, R. Timofte, L. Van Gool, and K. ESAT, *Anchored regression networks applied to age estimation and super resolution,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 1643–1652.

[109] J. Xing, K. Li, W. Hu, C. Yuan, and H. Ling, *Diagnosing deep learning models for high accuracy age estimation from a single image,* Pattern Recognition **66**, 106 (2017).

[110] H.-F. Yang, B.-Y. Lin, K.-Y. Chang, and C.-S. Chen, *Automatic age estimation from face images via deep ranking,* in *British Machine Vision Conference* (2015) pp. 55.1–55.11.

[111] Z. Niu, M. Zhou, L. Wang, X. Gao, and G. Hua, *Ordinal regression with multiple output CNN for age estimation,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 4920–4928.

[112] S. Chen, C. Zhang, M. Dong, J. Le, and M. Rao, *Using ranking-CNN for age estimation,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2017) pp. 5183–5192.

[113] G. Guo and X. Wang, *A study on human age estimation under facial expression changes,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2012) pp. 2547–2553.

[114] C. Zhang and G. Guo, *Age estimation with expression changes using multiple aging subspaces,* in *IEEE International Conference on Biometrics: Theory, Applications and Systems* (2013).

[115] G. Guo, R. Guo, and X. Li, *Facial expression recognition influenced by human aging,* IEEE Transactions on Affective Computing **4**, 291 (2013).

[116] Z. Lou, F. Alnajar, J. Alvarez, N. Hu, and T. Gevers, *Expression-invariant age estimation using structured learning,* IEEE Transactions on Pattern Analysis and Machine Intelligence (2017).

[117] A. Hadid, *Analyzing facial behavioral features from videos,* in *Human Behavior Understanding: Second International Workshop* (2011) pp. 52–61.

[118] N. Ramanathan, R. Chellappa, and S. Biswas, *Computational methods for modeling facial aging: A survey,* Journal of Visual Languages & Computing **20**, 131 (2009).

[119] G. Panis, A. Lanitis, N. Tsapatsoulis, and T. F. Cootes, *Overview of research on facial ageing using the FG-NET ageing database,* IET Biometrics **5**, 37 (2016).

[120] Z. Yang, X. He, J. Gao, L. Deng, and A. J. Smola, *Stacked attention networks for image question answering.* in *IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 21–29.

[121] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, *Spatial transformer networks,* NIPS (2015).

[122] W. Li, F. Abitahi, and Z. Zhu, *Action unit detection with region adaptation, multi-labeling learning and optimal temporal fusing,* IEEE Conference on Computer Vision and Pattern Recognition (2017).

[123] K. Zhao, W.-S. Chu, and H. Zhang, *Deep region and multi-label learning for facial action unit detection,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2016) pp. 3391–3399.

[124] H. Dibeklioğlu, A. A. Salah, and T. Gevers, *Are you really smiling at me? spontaneous versus posed enjoyment smiles,* in *European Conference on Computer Vision* (2012) pp. 525–538.

[125] T. Baltrušaitis, P. Robinson, and L.-P. Morency, *Openface: An open source facial behavior analysis toolkit,* in *IEEE Winter Conference on Applications of Computer Vision* (2016).

[126] T. Baltrušaitis, P. Robinson, and L.-P. Morency, *Constrained local neural fields for robust facial landmark detection in the wild,* in *IEEE International Conference on Computer Vision Workshops* (2013) pp. 354–361.

[127] P. F. Velleman, *Definition and comparison of robust nonlinear data smoothing algorithms,* Journal of the American Statistical Association **75**, 609 (1980).

[128] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors,* arXiv preprint arXiv:1207.0580 (2012).

[129] T. Tieleman and G. Hinton, *Lecture 6.5-RMSprop: Divide the gradient by a running average of its recent magnitude,* COURSERA: Neural Networks for Machine Learning **4** (2012).

[130] G. Zhao and M. Pietikainen, *Dynamic texture recognition using local binary patterns with an application to facial expressions,* IEEE Transactions on Pattern Analysis and Machine Intelligence **29**, 915 (2007).

[131] H. Sakoe and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition,* IEEE Transactions on Acoustics, Speech, and Signal Processing **26**, 43 (1978).

[132] K. Tsuda, M. Kawanabe, G. Rätsch, S. Sonnenburg, and K. Müller, *A new discriminative kernel from probabilistic models,* Neural Computation **14**, 2397 (2002).

[133] K. Tsuda, T. Kin, and K. Asai, *Marginalized kernels for biological sequences,* Bioinformatics **18**, 268 (2002).

[134] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, *Grammar as a foreign language,* in *Advances in Neural Information Processing Systems 28* (2015) pp. 2755–2763.

[135] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, *Signature verification using a siamese time delay neural network,* in *Advances in Neural Information Processing Systems*, Vol. 6 (1993).

[136] S. Chopra, R. Hadsell, and Y. LeCun, *Learning a similarity measure discriminatively with applications to face verification,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2005).

[137] R. Salakhutdinov and G. Hinton, *Learning a nonlinear embedding by preserving class neighbourhood structure,* in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, Vol. 11 (2007).

[138] G. Koch, R. Zemel, and R. Salakhutdinov, *Siamese neural networks for one-shot image recognition,* in *ICML 2015 Deep Learning Workshop* (2015).

[139] R. Min, L. van der Maaten, Z. Yuan, A. Bonner, and Z. Zhang, *Deep supervised t-distributed embedding,* in *Proceedings of the International Conference on Machine Learning* (2010) pp. 791–798.

[140] R. Hadsell, S. Chopra, and Y. LeCun, *Dimensionality reduction by learning an invariant mapping,* in *IEEE Conference on Computer Vision and Pattern Recognition* (2006).

[141] J. Hu, J. Lu, and Y.-P. Tan, *Discriminative deep metric learning for face verification in the wild,* in *IEEE International Conference on Computer Vision and Pattern Recognition* (2014).

[142] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, *Neighbourhood component analysis,* in *Neural Information Processing Systems* (2004).

[143] K. Weinberger and L. Saul, *Distance metric learning for large margin nearest neighbor classification,* The Journal of Machine Learning Research **10**, 207 (2009).

[144] J. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, *Information-theoretic metric learning,* in *Proceedings of the International Conference on Machine Learning* (2007) pp. 209–216.

[145] E. Xing, A. Ng, M. Jordan, and S. Russell, *Distance metric learning, with application to clustering with side-information,* in *Advances in Neural Information Processing Systems 16* (2002) pp. 521–528.

[146] M. Der and L. Saul, *Latent coincidence analysis: a hidden variable model for distance metric learning,* in *Advances in Neural Information Processing Systems 25* (2012) pp. 3239–3247.

[147] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, *End-to-end continuous speech recognition using attention-based recurrent NN: First results,* in *Deep Learning and Representation Learning Workshop: NIPS* (2014).

[148] S. Eddy, G. Mitchison, and R. Durbin, *Maximum discrimination hidden Markov models of sequence consensus,* Journal of Computational Biology **2**, 9 (1995).

[149] M. Kim and V. Pavlovic, *Discriminative learning of mixture of Bayesian network classifiers for sequence classification,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2006) pp. 268–275.

[150] N. Hammami and M. Sellami, *Tree distribution classifier for automatic spoken Arabic digit recognition,* in *Proceedings of the IEEE ICITST09 Conference* (2009) pp. 1–4.

[151] O. Aran, I. Ari, A. Benoit, P. Campr, A. H. Carrillo, F.-X. Fanard, L. Akarun, A. Caplier, M. Rombaut, and B. Sankur, *Sign language tutoring tool,* in *Proceedings of eNTERFACE 2006, The Summer Workshop on Multimodal Interfaces* (2006) pp. 23–33.

[152] T. Tieleman and G. Hinton, *Lecture 6.5 - rmsprop, coursera: Neural networks for machine learning,* (2012).

[153] D. Garreau, R. Lajugie, S. Arlot, and F. Bach, *Metric learning for temporal sequence alignment,* in *Advances in Neural Information Processing Systems* (2014).

[154] L. van der Maaten and G. Hinton, *Visualizing high-dimensional data using t-SNE,* Journal of Machine Learning Research **9**, 2579 (2008).

[155] H. Kamper, W. Wang, and K. Livescu, *Deep convolutional acoustic word embeddings using word-pair side information,* in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing* (2016).

[156] K. Riesen and H. Bunke, *Iam graph database repository for graph based pattern recognition and machine learning,* in *Lecture Notes in Computer Science*, Vol. 5342 (2008) pp. 287–297.

[157] J. Weston, S. Bengio, and N. Usunier, *Wsabie: Scaling up to large vocabulary image annotation,* in *Proceedings of the International Joint Conference on Artificial Intelligence* (2011).

[158] G. Synnaeve, T. Schatz, and E. Dupoux, *Phonetics embedding learning with side information,* in *IEEE Spoken Language Technology Workshop* (2014) pp. 106–111.

[159] X. Su and T. M. Khoshgoftaar, *A survey of collaborative filtering techniques,* Advances in Artificial Intelligence **2009**, 4 (2009).

[160] J. Bennett, S. Lanning, *et al.*, *The netflix prize,* in *KDD Cup Workshop*, Vol. 2007 (New York, NY, USA, 2007) p. 35.

[161] N. Koenigstein, G. Dror, and Y. Koren, *Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy,* in *RecSys* (ACM, 2011) pp. 165–172.

[162] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, *Time-aware point-of-interest recommendation,* in *SIGIR* (ACM, 2013) pp. 363–372.

[163] Y. Koren, *Collaborative filtering with temporal dynamics,* in *KDD* (ACM, 2009) pp. 447–456.

[164] T.-A. N. Pham, X. Li, G. Cong, and Z. Zhang, *A general graph-based model for recommendation in event-based social networks,* in *ICDE* (IEEE, 2015) pp. 567–578.

[165] H. Gao, J. Tang, X. Hu, and H. Liu, *Exploring temporal effects for location recommendation on location-based social networks,* in *RecSys* (ACM, 2013) pp. 93–100.

[166] Y. Shi, M. Larson, and A. Hanjalic, *Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,* CSUR **47**, 3 (2014).

[167] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, *Recurrent recommender networks,* in *WSDM* (ACM, 2017) pp. 495–503.

[168] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors,* Cognitive Modeling **5**, 1 (1988).

[169] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, *Session-based recommendations with recurrent neural networks,* arXiv preprint arXiv:1511.06939 (2015).

[170] H. Jing and A. J. Smola, *Neural survival recommender,* in *WSDM* (ACM, 2017) pp. 515–524.

[171] Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems,* Computer **42** (2009).

[172] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, *Bpr: Bayesian personalized ranking from implicit feedback,* in *UAI* (AUAI Press, 2009) pp. 452–461.

[173] A. P. Singh and G. J. Gordon, *Relational learning via collective matrix factorization,* in *KDD* (ACM, 2008) pp. 650–658.

[174] S. Rendle, *Factorization machines,* in *ICDM* (IEEE, 2010) pp. 995–1000.

[175] Y. Koren, *Factorization meets the neighborhood: a multifaceted collaborative filtering model,* in *KDD* (ACM, 2008) pp. 426–434.

[176] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, *E-commerce in your inbox: rroduct recommendations at scale,* in *KDD* (ACM, 2015) pp. 1809–1818.

[177] O. Barkan and N. Koenigstein, *Item2vec: Neural item embedding for collaborative filtering,* IEEE Workshop on MLSP (2016).

[178] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space,* arXiv preprint arXiv:1301.3781 (2013).

[179] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality,* in *NIPS* (2013) pp. 3111–3119.

[180] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, *Autorec: Autoencoders meet collaborative filtering,* in *WWW* (ACM, 2015) pp. 111–112.

[181] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks,* Science **313**, 504 (2006).

[182] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, *Neural collaborative filtering,* in *WWW* (ACM, 2017) pp. 173–182.

[183] S. A. Hosseini, K. Alizadeh, A. Khodadadi, A. Arabzadeh, M. Farajtabar, H. Zha, and H. R. Rabiee, *Recurrent poisson factorization for temporal recommendation,* arXiv preprint arXiv:1703.01442 (2017).

[184] W. Zaremba, I. Sutskever, and O. Vinyals, *Recurrent neural network regularization,* arXiv preprint arXiv:1409.2329 (2014).

[185] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, *Hierarchical attention networks for document classification,* in *NACAL-HLT* (2016) pp. 1480–1489.

[186] R. He, C. Lin, J. Wang, and J. McAuley, *Sherlock: Sparse hierarchical embeddings for visually-aware one-class collaborative filtering,* in *IJCAI* (2016) pp. 3740–3746.

[187] J. Yang, Z. Sun, A. Bozzon, and J. Zhang, *Learning hierarchical feature influence for recommendation by recursive regularization,* in *RecSys* (ACM, 2016) pp. 51–58.

[188] Z. Sun, J. Yang, J. Zhang, and A. Bozzon, *Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation,* in *AAAI* (2017) pp. 189–195.

[189] D. M. J. T. Wenjie Pei and L. van der Maaten, *Modeling time series similarity with siamese recurrent networks,* **arXiv preprint arXiv:1603.04713** (2016).

[190] K. He, X. Zhang, S. Ren, and J. Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,* in *ICCV* (2015) pp. 1026–1034.

[191] M. Schuster and K. K. Paliwal, *Bidirectional recurrent neural networks,* IEEE Transactions on Signal Processing **45**, 2673 (1997).

[192] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting.* JMLR **15**, 1929 (2014).

[193] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, *Image-based recommendations on styles and substitutes,* in *SIGIR* (ACM, 2015) pp. 43–52.

[194] F. Vasile, E. Smirnova, and A. Conneau, *Meta-prod2vec: Product embeddings using side-information for recommendation,* in *RecSys* (ACM, 2016) pp. 225–232.

[195] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, *Variational inference: A review for statisticians,* Journal of the American Statistical Association **112**, 859 (2017).

[196] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks,* in *Advances in Neural Information Processing Systems 28* (2015) pp. 91–99.

[197] K. He, G. Gkioxari, P. Dollár, and R. Girshick, *Mask r-cnn,* in *IEEE International Conference on Computer Vision* (2017).

[198] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition,* in *CVPR* (2016) pp. 770–778.

[199] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. Dauphin, *Convolutional sequence to sequence learning,* in *ICML* (2017).

[200] W. Pei, H. Dibeklioğlu, T. Baltrušaitis, and D. M. J. Tax, *Attended end-to-end architecture for age estimation from facial expression videos, arXiv preprint arXiv:1711.08690,* (2017).

[201] W. Pei, J. Yang, Z. Sun, J. Zhang, A. Bozzon, and D. M. Tax, *Interacting attention-gated recurrent networks for recommendation,* in *ACM International Conference on Information and Knowledge Management (CIKM)* (2017).

[202] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, *Densely connected convolutional networks,* in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[203] G. Andrew, R. Arora, K. Livescu, and J. Bilmes, *Deep canonical correlation analysis,* in *International Conference on Machine Learning (ICML)* (Atlanta, Georgia, 2013).

[204] E. Hoffer and N. Ailon, *Deep metric learning using triplet network,* in *ICLR workshop* (2015).

[205] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets,* in *Advances in Neural Information Processing Systems 27* (2014) pp. 2672–2680.

[206] D. P. Kingma and M. Welling, *Auto-encoding variational bayes,* in *ICLR* (2013).

SUMMARY

Much of the observational data that we see around is, is ordered in space or time. For instance, video data, audio data or text data. This ordered data, called sequence data, calls for automatic analysis using supervised learning. Traditional single-observation supervised learning is challenged by sequence data, because (1) the length of sequence examples is often variable; (2) the sequence data may contain irrelevant segments which yield negative impact on the learning performance and (3) there exist temporal dependencies between consecutive observations in a sequence that need to be exploited by supervised learning on sequence data. This thesis introduces new models for supervised learning on sequence data that specifically address these challenges.

We first propose a sequence classification model which is a graphical model using hidden variables to model the latent structure in the sequence data. It advances the state-of-the art by using the same number of hidden variables to model much more complex decision boundaries. Subsequently, we present a sequence classification model which is able to deal with unsegmented sequences. The proposed model integrates ideas from attention models and gated recurrent neural networks. It is able to discern the salient segments and filter out the irrelevant ones, but it also measures the relevance of each time step of the sequence data to the final task. Finally, we propose an end-to-end model for age estimation from facial expression videos that performs feature learning and supervised learning for the final task jointly.

Next we considered the supervised learning on paired sequences in which we want to predict whether the two sequences are similar. We combined ideas from sequence modeling and metric learning, and propose Siamese Recurrent Networks to learn a good similarity measure between two sequences. Our model is superior to current techniques that are based on handcrafted similarity measures or models using unsupervised learning. Finally, we present a model that predicts the preference of users for items in a recommendation system. In this case, two input sequences represent a pair of historic user and item data each with their own properties. The dependencies between the two sequences are modeled using an attention scheme.

SAMENVATTING

Veel van de waarnemingsgegevens die we om ons heen zien, is geordend in ruimte of tijd. Bijvoorbeeld videogegevens, audiogegevens of tekstgegevens. Deze geordende data, sequentiegegevens genaamd, roept op tot automatische analyse met behulp van gesuperviseerd leren. Voor traditioneel gesuperviseerd leren op ongeordende data zijn sequentiegegevens een uitdaging, omdat (1) de lengte van sequentievoorbeelden vaak variabel is; (2) de sequentiedata kunnen irrelevante segmenten bevatten die een negatieve invloed hebben op de leerprestaties en (3) er bestaan temporele afhankelijkheden tussen opeenvolgende waarnemingen in een sequentie die moeten worden uitgebuit door gesuperviseerd leren over sequentiegegevens. Dit proefschrift introduceert nieuwe modellen voor gesuperviseerd leren over sequentiegegevens die specifiek deze uitdagingen aanpakken.

We stellen eerst een sequentieklassificatiemodel voor, een grafisch model dat verborgen variabelen gebruikt om de latente structuur in de sequentiegegevens te modelleren. Het verbetert de state-of-the-art door hetzelfde aantal verborgen variabelen te gebruiken om veel complexere beslissingsgrenzen te modelleren. Vervolgens presenteren we een sequentieclassificatiemodel dat niet-gesegmenteerde sequenties kan verwerken. Het voorgestelde model integreert ideeën uit aandachtsmodellen en Gated Recurrente Neurale Netwerken. Het is in staat om de saillante segmenten te onderscheiden en de irrelevante segmenten eruit te filteren, maar het meet ook de relevantie van elke tijdstap van de sequentiegegevens. Ten slotte stellen we een end-to-end-model voor leeftijdschatting voor van gelaatsuitdrukkingvideo's waarin zowel de kenmerken als de uiteindelijke taak gelijktijdig worden geleerd.

Vervolgens hebben we gekeken naar het gesuperviseerde leren op gepaarde sequenties waarin we willen voorspellen of de twee sequenties vergelijkbaar zijn. We combineerden ideeën uit sequentiemodellering en metrisch leren en stelden Siamese Recurrente netwerken voor om een goede similariteitsmeting tussen twee sequenties te leren. Ons model is superieur aan de huidige technieken die zijn gebaseerd op handgemaakte gelijkheidsmaatstaven of modellen die gebruikmaken van niet-gesuperviseerd leren. Ten slotte presenteren we een model dat de voorkeur van gebruikers voor items in een aanbevelingssysteem voorspelt. In dit geval vertegenwoordigen twee invoerreeksen een paar historische gebruikers- en artikelgegevens met elk hun eigen eigenschappen. De afhankelijkheden tussen de twee sequenties worden gemodelleerd met behulp van een aandachtschema.

ACKNOWLEDGEMENTS

This thesis is kind of a wonderful reward for my four years' research journey, which is indeed challenging and full of memorable stories. Those stories involve many important roles in my life, who offer me the guidance, support, encouragement, faith and love. Without them, the stories along my PhD journey cannot be so vivid and this thesis would be impossible. At this very moment, I would like to deliver my sincere gratitude to each of them.

First and foremost, I would like to give my deepest gratitude to my supervisor David Tax. Thank you, David, for your always patient guidance and thoughtful advices along my PhD journey. Actually, each of my research projects, succeeded or failed, went through a lot of helpful and inspiring discussion between us, which trained me to think more rigorously, logically and deeply, and communicate more clearly and efficiently. More importantly, you were always trying to guide me to be an independent researcher, namely, teach me to fish instead of giving me a fish. Your word "There is always a place to fit you, you'll find it." consoles me whenever I am confused or depressed. I have learned a lot from you on both the way of doing research and the viewpoints of dealing with life.

I am greatly indebted to my co-supervisor Laurens van der Maaten. Thank you for leading me to the fascinating world of scientific research and guiding me to do research in a professional, efficient and smart way in my first PhD year, which brings me a smooth PhD startup. I am deeply influenced by your attitude of tackling the tough research difficulties: confident, calm and dedicated. You always encourage me in a convincing way and meanwhile motivate me in an inspiring way. It was a great pleasure and honor to work with you.

My promoter, Marcel Reinders, also deserves my full gratitude. Thank you very much for your overall advices and supports during my whole PhD period. Thanks for making time to revise this thesis earnestly.

My sincere gratitude goes to all my committee members for your time on reading my thesis and participating in my defense.

Special thanks go to Hamdi Dibeklioğlu who provided me a lot of help and support as a senior researcher and great collaborator, and brought me enormous joy as a close friend and officemate. Many thanks to Jan van Gemert, from whom I learned a lot during our collaboration. Your 'writing guidelines' is really instructive and you paper tips, like"Each experiment should aim to answer one research question specifically" or "Avoid using too strong words in the experimental analysis since reviewers tend to be harsher", are really helpful.

I would like to thank LP Morency, who offers me the precious and productive internship opportunity at Language Technology Institute of Carnegie Mellon University. Sincere thanks also go to Tadas Baltrušaitis, who is a great collaborator and gave me much help and guidance.

I enjoyed my four years in PRB group thanks to my pretty nice colleagues and the harmonious atmosphere they created. As senior researchers, Lu, Veronika, Jesse, Julian, Silvia, and Gorkem gave me much help and good advices, thank you so much. Laura, Wouter and Ekin, thank you a lot for your encouragement whenever I feel depressed, and thanks for those candid and sincere chats between us. Taygun, thanks for being my defense paranymph and I am looking forward very much to your book (you mentioned many times in our talks). Bob, Hayley, and Marco, thanks for your continuous supply of brilliant and insightful research thoughts and comments, which broaden and deepen my research vision. I will never forget countless pleasant moments at Thursday's drinks, especially dancing with Marco and Sally after getting slight drunk. I would also like to thank other members of PRB groups for their help and friendship: Osman, Tom, Alexander, Saskia, Ruud, Bart, John, Seyran, Reza, Amogh, John, Thomas, Joana, Amin, stavros, Robbert, Christine, Sjoerd, Arlin, Christian, Ahmed, and Alexey. Warm thanks to the mini Chinese community in the group: Yuan Zeng, Yunqiang, Yancong, Stephanie, Yanxia, who give me much enjoyable time during my PhD life. Special thanks go to Yazhou, Lu and Yuanhao for your great friendship and endless help.

I feel grateful to my Chinese friends in the Netherlands, who made my PhD life more enjoyable and less stressful. Yan Jin, Hui Jiang, Yuan Lu and Xuefei, thanks a lot for our sincere and eternal friendship. I also would like to thank the 'ping-pong and BBQ community': Tian Zhang, Fei Xu, Jiao Chen, Jin Chang, Yun Wan, Junru Wu and Xiaolin Wang for much joy you brought to me. Many thanks to Bo Qu, Xinchao, Zhu Sun, Qin Lin, Chao Zhang, Fan Feng, Nan Zhong, Qingqing Ye, Zhijie Ren, Yu Xin, Yue Zhao, Guanliang Chen, Qiang Liu, Zhidong He and all other Chinese friends whom I couldn't mention here. Particular thanks to Jie Yang, my intimate friend and excellent collaborator from my master in Eindhoven to PhD in Delft.

I am deeply grateful to my fiancee Xiangrong. Thank you, Rong, for your love, company, support, understanding and everything you did for me. I firmly believe that our love will last forever.

Last but not least, I would like to express my upmost gratitude to my family: my parents, my brother, and all my beloved relatives, who gave me unconditional love all the time in my life. In particular, I would like to thank my grandparents, who raised me with their deep love, taught me the first but most important lesson of my life, and endowed me the courage to live.

'Be kind, for everyone is fighting a hard battle.' – Wonder (film)

Wenjie Pei
April 2018
Delft, the Netherlands

CURRICULUM VITÆ

Wenjie Pei is currently a PhD candidate at Pattern Recognition and Computer Vision Lab, Delft University of Technology. He works with Dr. David M.J. Tax (TU Delft) and Dr. Laurens van der Maaten (Facebook AI Research). His research focuses on the sequence (time series) modeling. In 2016, he visited Language Technology Institute of Carnegie Mellon University (CMU), working with Prof. Louis-Philippe Morency and Dr. Tadas Baltrušaitis.

Wenjie Pei received his BSc degree from Shanghai Jiao Tong University of Computer Science in 2008. Then he obtained his first master degree majored in Visualization, under the supervision of Prof. Hujun Bao and Prof. Jin Huang, from State Key Lab of CAD&CG Lab, Zhejiang University in 2011. Afterwards, he moved to Eindhoven University of Technology to pursue his second master degree in Computer Science and Engineering. During that time, he worked on pattern mining advised by Prof. Toon Calders and Dr. Hoang Thanh Lam. In 2013, he had an internship at Philips Research Eindhoven.

LIST OF PUBLICATIONS

Thesis Research:

- **Wenjie Pei**, Hamdi Dibeklioğlu, David M. J. Tax and Laurens van der Maaten. *Multivariate Time-Series Classification Using the Hidden-Unit Logistic Model*, IEEE Transactions on Neural Networks and Learning Systems (TNNLS) **29** (4), pages 920-931, 2018.
- **Wenjie Pei**, Tadas Baltrušaitis, David M.J. Tax and Louis-Philippe Morency. *Temporal Attention-Gated Model for Robust Sequencce Classification*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- **Wenjie Pei**, Hamdi Dibeklioğlu, Tadas Baltrušaitis and David M.J. Tax. *Attended End-to-End Architecture for Age Estimation from Facial Expression Videos*, IEEE Transactions on Image Processing (TIP), 2017, submitted.
- **Wenjie Pei**, David M.J. Tax and Laurens van der Maaten. *Modeling Time Series Similarity with Siamese Recurrent Networks*, arXiv, 2016.
- **Wenjie Pei***, Jie Yang*, Zhu Sun, Jie Zhang, Alessandro Bozzon and David M.J. Tax. *Interacting Attention-gated Recurrent Networks for Recommendation*, The 26th ACM International Conference on Information and Knowledge Management (CIKM), 2017. (*Equal contribution)

Side Research:

- **Wenjie Pei** and David M.J. Tax. *Unsupervised Learning of Sequence Representations by Autoencoders*, Pattern Recognition, 2018, submitted.
- Yunqiang Li*, **Wenjie Pei***, Yufei Zha and Jan van Gemert. *Deep Fisher Discrete Hashing*, Neural Information Processing Systems (NIPS), 2018, submitted. (*Equal contribution)
- Yunqiang Li*, **Wenjie Pei***, Yufei Zha, Bing Chen, David M.J. Tax and Jan van Gemert. *Shift-variant Dissimilarity Learning for Deep Siamese Person Re-Identification*, European Conference on Computer Vision (ECCV), 2018, submitted. (*Equal contribution)
- Hoang Thanh Lam, **Wenjie Pei**, Adriana Prado, Baptiste Jeudy and Élisa Fromont. *Mining top-k largest tiles in a data stream*, Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD), 2014.
- **Wenjie Pei**, *Extracting Features to Discriminate OSA and non-OSA*, Master Thsis in Eindhoven University of Technology (TU/e), 2013.
- Jin Huang, **Wenjie Pei**, Chunfeng Wen, Guoning Chen, Wei Chen and Hujun Bao. *Output-coherent image-space lic for surface flow visualization*, IEEE Pacific Visualization Symposium (PacificVis), 2012.
- Jin Huang, MuYang Zhang, **WenJie Pei**, Wei Hua and HuJun Bao. *Controllable highly regular triangulation*, Science China Information Sciences **54** (6), pages 1172-1183, 2011.