A COSINE-DISTANCE BASED NEURAL NETWORK FOR MUSIC ARTIST RECOGNITION USING RAW I-VECTOR FEATURES

Hamid Eghbal-Zadeh
Department of Computational Perception,
Johannes Kepler University
Linz, Austria
hamid.eghbal-zadeh@jku.at

Matthias Dorfer
Department of Computational Perception,
Johannes Kepler University
Linz, Austria
matthias.dorfer@jku.at

Gerhard Widmer
Department of Computational Perception,
Johannes Kepler University
Linz, Austria
gerhard.widmer@jku.at

ABSTRACT

Recently, i-vector features have entered the field of Music Information Retrieval (MIR), exhibiting highly promising performance in important tasks such as music artist recognition or music similarity estimation. The i-vector modelling approach relies on a complex processing chain that limits by the use of engineered features such as MFCCs.

The goal of the present paper is to make an important step towards a truly *end-to-end* modelling system inspired by the i-vector pipeline, to exploit the power of Deep Neural Networks[1] (DNNs) to learn optimized feature spaces and transformations. Several authors have already tried to combine the power of DNNs with i-vector features, where DNNs were used for feature extraction, scoring or classification. In this paper, we try to use neural networks for the important step of i-vector post-processing and classification for the task of music artist recognition.

Specifically, we propose a novel neural network for i-vector features with a cosine-distance loss function, optimized with stochastic gradient decent (SGD). We first show that current networks do not perform well with unprocessed i-vector features, and that post-processing methods such as Within-Class Covariance Normalization (WCCN) and Linear Discriminant Analysis (LDA) are crucially important to improve the i-vector representation. We further demonstrate that these linear projections (WCCN and LDA) can not be learned using general objective functions usually used in neural networks.

We examine our network on a 50-class music artist recognition dataset using i-vectors extracted from frame-level timbre features. Our experiments suggest that using our network with fully unprocessed i-vectors, we can achieve the performance of the i-vector pipeline which uses i-vector post processing methods such as LDA and WCCN.

1. INTRODUCTION

In the area of MIR, music artist modelling can have different applications including in recommender systems, playlist generation and music similarity estimation. Each artist can be recognized by a combination of multiple factors such as musical instruments, genre and voice of the singer(s).

I-vector features first were proposed in the field of speaker verification [1] and after their revolutionary success, they were used in other areas such as emotion recognition [2], language recognition [3] and audio scene classification [4]. Recently, they were imported into the MIR domain, for singing language identification [5], music artist recognition [6] and music similarity [7].

I-vector features have shown to be a promising song-level representation for artists. These features project songs into a fixed-length and low-dimensional space, which is built from frame-level features such as Mel-Frequency Cepstrum Coefficients (MFCCs) using Factor Analysis (FA).

First by using a Universal Background Model (UBM) trained on a sufficient number of songs, similarities among all the songs of different artists are captured, then via FA these similarities are discarded and songs are projected into a new space called **Total Variability Space (TVS)** which contains the remaining factors that are in a stronger correlation with artist variability. Further, an estimation of these factors in each song is calculated which contains rich information about the artist. These estimated factors are called **identity vectors** or in short, **i-vectors**. The use of Neural Networks (NNs) in different areas is increasing every day and recent advances in this area, enabled researchers to tackle problems which were previously solved by a variety of different approaches in machine learning, now by only using NNs. The outcome is the appearance of different NN layers and architectures specialized for different tasks.

I-vector based systems usually follow a specific pipeline which contains a chain of different processing steps with specific goals. Multiple efforts are done by different researchers to come up with a solution that replaces each of those blocks with NNs. The reason is that once all of these blocks are replaced with a NN, they all can be connected through a deep network and optimized together using the back-propagation algorithm.

The frame-level feature extraction – a part of the i-vector extraction procedure – and scoring and classification of i-vectors are examples of the steps that have been replaced with NNs. Yet, a solution for post-processing the raw i-vectors[2] using neural networks is not provided in classification tasks. We seek for a NN-based solution to post-process and classify i-vectors without any help from the i-vector pipeline. We hope that our efforts makes us one step closer to an *end-to-end* music artist recognition system inspired by the i-vector pipeline, using neural networks.

In this paper, we extract i-vectors from frame-level timbre features and use them as input to NNs. By defining a cosine-distance loss function, we lead the network to learn a cosine metric which works the best with i-vector features. Our results suggest that us-

[1] The term **Deep Neural Networks** in this paper refers to **Multi-Layer Artificial Neural Networks** which use recent techniques from **Deep Learning** such as **batch-normalization**, **drop-out** and **stochastic gradient descent**.

[2] In this paper, i-vectors that are not encountered with linear projections such as LDA and WCCN are called *raw i-vectors*. In contrast, i-vectors that are projected with linear projections such as LDA and WCCN are called *processed i-vectors*.

ing our network, we can achieve the performance of the i-vector pipeline which is the state-of-the-art in music artist recognition.

The remainder of this paper is organized as follows. In Section 2, the related work is provided. In Section 3, the i-vector features are described. In Section 4, we explain our proposed neural network. In Section 5 the details of the experiments are explained and the results are reported. And finally, Section 6 concludes the paper.

2. RELATED WORK

For the task of Music Artist Recognition (MAR) in MIR, multiple approaches have been followed. Frame-level features [8], ensemble [9] and coding approaches [10] are some of the most used methods.

NNs are also known to perform well in MAR. In [11, 12, 13], Deep Belief Networks (DBNs) are used MAR using spectrograms and timbral features.

I-vector features have proven to be a promising song-level feature for MAR. In [6], i-vector features extracted from MFCCs and spectral features are used for MAR. Also, in [14] i-vectors have shown a significant performance for MAR in noisy environments.

Even though i-vector features are not used with DNNs for artist recognition, they are combined with deep learning techniques in many different ways. In [15, 16] speech recognition DNNs are used to produce statistical vectors needed for i-vector extraction. In [17], a DNN is used to extract a low-dimensional representation similar to i-vectors. Also, to extract bottleneck features used for i-vector extraction, DNNs are utilized in [18]. And in [19], DNNs are employed to manipulate post-processed i-vectors for speaker recognition.

In [20] DBNs are used with raw[3] i-vector features to model discriminatively the target and impostor i-vectors in a speaker verification scenario and in [21], DBNs and DNNs are combined together for single and multi-session speaker recognition.

The methods mentioned above that use raw i-vectors with DNNs, pursue adaptation purposes or have used DBNs which are trained in an unsupervised manner. Others, use the processed i-vectors as an input.

3. I-VECTOR FEATURES

3.1. Theoretical background

An i-vector refers to vectors in a low-dimensional space called Total Variability Space (TVS). The TVS models variabilities encountered with both the artist and song [7] where, the song variability defines as the variability exhibited by a given artist from one song to another.

TVS is created using a matrix $\mathbf{T}$ known as *TVS matrix*. This matrix is obtained by applying Factor Analysis (FA) on the adapted means of a Gaussian Mixture Model (GMM) known as Universal Background Model (UBM). This UBM is trained on the acoustic features of a sufficient amount of data. The means of UBM are then adapted to each song and then are used for the FA procedure explained in [1].

In the TVS, a given song is represented by an **i-vector** which indicates the directions that best separate different artists.

[3] I-vectors that are not encountered with post-processing methods such as LDA or WCCN.

A GMM mean supervector $\mathbf{M}$ adapted to a song from artist α can be decomposed as follows:

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{y} \tag{1}$$

where $\mathbf{m}$ is the GMM mean supervector and $\mathbf{T}\mathbf{y}$ is an offset. $\mathbf{y}$ is a latent variable with the standard normal prior. The i-vector is defined as the MAP estimate of $\mathbf{y}\mathbf{M}$ is assumed to be normally distributed with mean vector $\mathbf{m}$. The obtained i-vector is an artist and song dependent vector. The matrix $\mathbf{T}$ is used to extract i-vectors from statistical supervectors (known as $\mathbf{N_s}$ and $\mathbf{F_s}$) of songs which are computed using UBM.

We calculate statistical supervectors for a specific song s using UBM. These supervectors are $\mathbf{N_s}$ and $\mathbf{F_s}$ of song s:

$$\mathbf{N_s}^c = \sum_{t=1}^{L} \gamma_t(c), \quad \mathbf{F_s}^c = \sum_{t=1}^{L} \gamma_t(c) Y_t \tag{2}$$

where $\gamma_t(c)$ is the posterior probability of Gaussian component c of UBM for frame t and Y_t is the MFCC feature vector at frame t.

After calculating $\mathbf{N_s}$ and $\mathbf{F_s}$, using the statistical supervectors of songs in training set we learn the $\mathbf{T}$ matrix via an Expectation Maximization (EM) algorithm as follows: E-step, computes the probability of $P(w|X)$ where X is the given song and w is its i-vector. M-step, optimizes $\mathbf{T}$ by updating the following equation:

$$\mathbf{w} = (\mathbf{I} + \mathbf{T}^t \mathbf{\Sigma}^{-1} \mathbf{N(s)} \mathbf{T})^{-1} \mathbf{T}^t \mathbf{\Sigma}^{-1} \mathbf{F(s)} \tag{3}$$

where $\mathbf{N(s)}$ and $\mathbf{F(s)}$ are diagonal matrices with $\mathbf{N_s}^c\mathbf{I}$ and $\mathbf{F_s}^c\mathbf{I}$ on diameter. $\mathbf{I}$ is the identity matrix and $\mathbf{\Sigma}$ is the diagonal covariance matrix. The actual computation of an i-vector $\mathbf{w}$ for a given song s can be done using (3) after training $\mathbf{T}$. The curious reader is referred to [1, 22] for more information about the training procedure of $\mathbf{T}$.

3.2. I-vector post-processing and scoring

As explained before, i-vectors contain both artist and song variability. The song variability can be reduced by applying post-processing techniques such as LDA [23] and WCCN [24]. These techniques project i-vectors into a space which minimizes the song variability and maximizes the artist variability. In this section, we describe three techniques (LDA, WCCN and Length Normalization) that are frequently used in i-vector pipeline for post-processing. Also we describe a scoring method for classifying the i-vectors.

WCCN: provides a linear projection with an effective compensation. WCCN scales the i-vector space in the opposite direction of its inter-class covariance matrix, so that directions of intra-artist variability are improved for i-vector scoring.

LDA: yields a linear projection that tries to find a orthogonal basis with a better discrimination between different classes. LDA projection maximizes the between-class and minimize the within-class covariance of the data.

Length Normalization: Length (amplitude) of i-vectors are in correlation with negative effects such as song variability. For this reason, an iterative length-normalization for i-vectors is proposed in [25] where suggest to divide each i-vector by its length (norm). It is suggested to apply length-normalization before each post processing, also before feeding to the classifier/scoring step.

Cosine Scoring (CS): In the TVS, a simple cosine scoring has been successfully used to compare two i-vectors, as described in [26]. To predict the artist label for an i-vector, the artist with

the highest cosine score is chosen as the label where the score is defined as the cosine score of the given i-vector and class-averaged i-vectors[4].

3.3. I-vector pipeline

In Figure 1 (top), a diagram of an i-vector based system is shown. As you can see, first the frame-level features are extracted and then the i-vector models (such as UBM and **T** matrix) are trained and then i-vectors are computed. Further, these raw i-vectors are encountered with post-processing methods. First LDA is applied and the resulting i-vectors are projected using WCCN. Finally, LDA-WCCN projected i-vectors are used for scoring via Cosine Scoring.

4. THE PROPOSED NETWORK

In this section, we introduce our proposed NN for music artist recognition using i-vector features. As we show in Figure 1 (bottom), instead of using post-processing methods such as LDA and WCCN, and scoring methods such as cosine-scoring, our network is able to use raw i-vectors directly as an input.

Our experiments show that linear projections such as LDA and WCCN play a significant role in the performance of i-vector pipeline. Hence, we would like to replace such linear projections with a NN. We use linear activation function (LIN) in our NN. The reason to choose LIN is that other layer activation functions such as rectify units discard all the negative values by replacing them with zero Because i-vectors have a mean value close to zero, by using a rectified activation function [27], the layer's output activations that still have negative values, might be forced to throw away the information related to negative values of i-vector features.

Instead of the common loss functions used with NNs such as Mean Squared Error (MSE) and Categorical Cross-Entropy (CCE), we introduce a novel Cosine-distance based loss function for multi-class classification tasks using i-vector features. Our Cosine-Distance Based Neural Network (CDB-Net) is described in details in the following.

Architecture: Our CDB-Net consists of 4 layers with 1 hidden layer. The first hidden layer is a dense (fully connected) layer with linear activation function. It is followed by a batch-normalization layer [28] then a drop-out layer [29] and finally a dense output layer with linear activation function. Using the drop-out, prevents the network from over-fitting and let each feature to be learned, without relying on other dimensions. During training a NN the parameters of a layer change, and consequently the distribution of each layer's outputs changes as well. Batch-normalization layer normalizes each layer output for each training mini-batch. This allows us to use higher learning rates and be less careful about initialization. The aim of the first hidden layer is to learn a linear projection that improves the i-vector representation. We expect that since the i-vector pipeline benefits from LDA and WCCN linear projections, our CDB-Net also should be able to learn a projection with similar characteristics. Our first hidden layer has 400 hidden neurons (the same as the i-vectors dimensionality).

The output activations of this layer and one-hot encoding of the correct class are used to calculate a cosine loss. In the output layer, we use the same number of hidden neurons as our classes to produce a score for each class given an i-vector.

Cosine loss function: For the loss optimization, we use a novel Cosine-distance based loss. This loss is the cosine distance of the activations of the output layer with one-hot encoding of the correct class. The calculation of our proposed loss is as follows.

For a C classes problem, a one-hot encoding of class i ($i = 1, \ldots, C$) is one at the index i and zero otherwise. The cosine loss of the network for a batch size of b is defined as follows:

$$loss_{cos} = 1 - \frac{1}{b}\sum_{n=1}^{b} cos(out_n, l_n) \tag{4}$$

where out_n is the output activation of the output layer and l_n is the one-hot encoding for the correct class. Also, cos is a standard cosine similarity [26]. Then maximum of the cosine loss is equal to 1. Also, each one-hot encoding is orthogonal to the others. So the labels have the maximum distance from each other. Hence, minimizing the cosine loss of the output activations to the correct one-hot encoding, will increase the discrimination power of the network. The network tries to minimize this loss by using stochastic gradient descent. Then it back-propagates the error through the previous layers to update the weight of the layers.

This is not the first time that cosine-distance is used as loss in a NN. In [30], a NN optimized with a similar cosine distance-based loss function is used for the task of signature verification. Although the cosine loss function in [30] is used to process both imposter and target signature features. Some of the differences between our CDB-Net loss and loss used in [30] can be explained as: 1) the loss definitions are different. The loss in [30] is defined for a special NN called "Siamese" containing two identical sub-networks with a special training procedure. This network is designed to compare two signatures for signature verification. The loss defined in our CDB-Net can be used in any multi-class classification task. 2) The Siamese network training tries to minimize the cosine distance of two output activations from two different signatures. CDB-Net minimizes the cosine distance of the output activations of a given i-vector with its correct one-hot encoding label in a discriminative manner.

Network distance metric: The proposed network is forced to use the cosine distance metric in its optimization because of two reasons: 1) the input raw i-vectors are length-normalized. So the network can not distinguish between different classes using a distance metric such as Euclidean distance that relies on the amplitude of the output activations. 2) The loss function is a cosine distance between output layer activation and their respective one-hot encoding class label. So the network's objective function is defined by a cosine metric.

5. EXPERIMENTS

5.1. Data

Similar to the data used in [12] for music artist recognition, we used a subset of Million Song Dataset [31] (MSD) in the artist recognition experiments. We follow a similar procedure as used in [12]. We first removed the duplications from MSD by using the official duplication list provided in MSD website which reduced the number of songs from one million to around 900,000. Then we selected all the artists with more than 100 songs. From these artists, we selected top 50 artists with more songs and further selected 100 random songs from each artist (in sum, 5,000 songs)

[4] class-averaged i-vectors are defined as the average of i-vectors in each class, in training set.

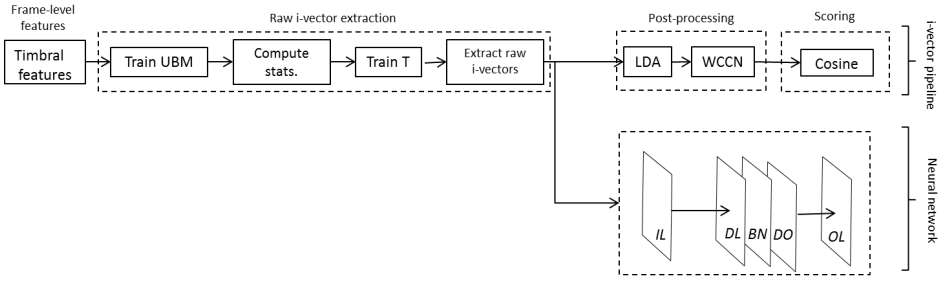

Figure 1: *The block-diagram of i-vector based artist recognition system. On the top, classic state-of-the-art i-vector pipeline. On the bottom, the proposed artist recognition method using a deep network.* ***IL****: input layer.* ***DL****: dense layer.* ***BN****: batch-normalization layer.* ***DO****: dropout layer.* ***OL****: output layer.*

as our dataset for the experiments in this paper. 80% of the songs are used for training and the rest are used for validation (10%) and testing (10%). We trained our models on training set, optimized using validation set and reported the results on the testing set.

The results reported in [12] (35.74% accuracy) can not be compared with the performance of our network because of two reasons:

1) in [12] both timbre and chroma features are used together. and 2) the performance reported in [12] is in a bar level (which is smaller than a song), but i-vectors are song-level features and our performance is reported in the song level.

5.2. Features

Using the features available through The Echo Nest API[5], we extracted i-vectors from Echo Nest Analyzer's timbre feature. Each Echo Nest Analyzer's timbre feature consists of a vector that includes 12 unbounded values roughly centered around 0. Those values are high level abstractions of the spectral surface, ordered by degree of importance such as the average loudness of the segment, brightness, the flatness of a sound and attack [32]. As meta-data, we used the artist ids provided in MSD in our artist recognition experiments. To extract i-vectors, all the Echo Nest timbre features for each song are used.

5.3. Setup

I-vector extraction: For i-vector extraction, a 1024 components GMM is trained as UBM on Echo Nest timbre features. Then, **T** matrix with 400 dimensions is learned using the statistics computed from Echo Nest timbre features and UBM. The training set is used to train UBM and **T**. Further i-vectors are extracted using UBM and **T** for both training and testing sets. All the i-vectors are length normalized.

The **T** matrix is trained using matlab MSR identity toolbox [33].

CDB-Net: We used 400 neurons in our hidden layer and 50 neurons in the output layer with 50. The hidden layer is followed by batch normalization and 50% droup-out.

We apply learning-rate schedule during training and decrease the learning rate by its half after each 10 epochs. The initial learning rate is 1.0 the learning-rate starts to decrease at the 100^{th} epoch out of 200 epochs used for training.

[5]http://the.echonest.com/

A stochastic gradient descent (SGD) with back-propagation algorithm and a momentom of 0.9 is used with the batch size of 500 samples. The one-hot coding of the labels are computed to be used in the cosine-loss calculation.

For all of our experiments with NNs, the open-source python library *Lasagne* [34] is used. Our network is implemented in Python using *Theano* [35].

We used a PC running on Linux with a NVIDIA Titan X GPU card, an Intel Core i7 CPU and 16 GB of RAM for our experiments. All the NN experiments are optimized on GPU.

We use the averaged F-measure to compare the performance of different methods. This measurement is calculated by averaging the F-measures of all the classes in each experiment.

Baselines: Four baseline methods are used in this work. We use the i-vector pipeline method and three NNs similar to our CDB-Net as baselines. The difference between our CDB-Net and other NN baselines is the loss function and the activation functions of the hidden layer. Our first baseline is the Cosine Scoring (CS) used in i-vector pipeline. CS first projects i-vectors using LDA and then by WCCN. Further, it uses the cosine distance to calculate a score for each given testing i-vector and class-averaged i-vectors from training set. Finally, it classifies testing i-vectors by minimizing the cosine score. Similar to the architecture of CDB-Net, our second baseline (CCE-REC-Net) is a 4 layers feed-forward network with 1 hidden layer of 400 neurons which uses rectified activation function. The hidden layer is followed by batch-normalization layer and a drop-out layer with 50% drop outs. At the output layer, CCE-REC-Net uses a soft-max activation function. CCE-REC-Net is optimized using a CCE loss function.

Our third baseline (CCE-TAN-Net) is a a 4 layers feed-forward network with 1 hidden layer of 400 neurons. CCE-TAN-Net has exactly the same architecture as CCE-REC-Net, only uses $\tanh$ activation function instead of rectified activation function. Our fourth baseline (CCE-LIN-Net) is also a a 4 layers feed-forward network with 1 hidden layer of 400 neurons. CCE-LIN-Net has exactly the same architecture as CCE-REC-Net, only uses linear activation function instead of rectified activation function.

Experiment design: We examine the performance of our CDB-Net in three different experiments: 1) dealing with LDA-WCCN projected i-vectors, 2) dealing with raw i-vectors and 3) The effect of weight initialization in NNs.

In our first experiment, we compare the performance of CDB-Net and our baselines on processed i-vectors. Since the i-vector pipeline uses LDA and then WCCN projections for post-

processing, we use the LDA, then WCCN projected i-vectors in our first experiment. Our first experiment reveals how different networks–as well as the i-vector pipeline–deal with processed i-vectors. The results of this experiment are provided in Table 1

Our second experiment compares the CDB-Net with the baselines encountering raw i-vectors. This experiment is the core of this paper and shows how our CDB-Net performs compared to all the other baselines.

Finally, in our third experiment, we study the effect of weight initialization in the hidden layer of the baseline NNs as well as our CDB-Net. In [36] the importance of layer's weight initialization in feed-forward NNs is discussed in details. In the experiments 1 and 2, we initialize the weight of our hidden layer from a uniform distribution as explained in [36]. In experiment 3, we would like to compare this initialization with an initialization using the LDA-WCCN projection matrix which is used in the i-vector pipeline for i-vector post-processing.

Even though we are aware that initializing the hidden layer with LDA-WCCN projection matrix is similar to use processed i-vectors, we would like to provide proof that our CDB-Net is able to find an optimum point that other NNs are unable to find using SGD without a proper initialization. If we initialize the hidden layer's weight, or use processed i-vectors, other netowrks are able to reach that optimum point.

Using a LDA-WCCN projection matrix, we initialize the hidden layer's weight matrix in all of our baseline NNs as well as our CDB-Net. Then we feed all the networks with raw i-vectors. The LDA-WCCN projection matrix is computed by multiplying LDA and WCCN projection matrices. In [26] a procedure is described about how to combine LDA and WCCN projection matrix to be used with i-vector features and cosine scoring in an efficient way. We follow the same procedure to compute our LDA-WCCN projection matrix.

5.4. Results

The performance of NNs with LDA-WCCN projected i-vectors can be found in Table 1. Also, the performance of the i-vector pipeline can be found under (CS) name.

It can be seen that using LDA-WCCN projected i-vectors, all the networks achieved similar performances to the i-vector pipeline. Also, it can be observed that CCE-LIN-Net and CDB-Net achieved better performances than CCE-REC-Net and CCE-TAN-Net. It shows that rectified and $\tanh$ activation functions were not useful, as expected.

As the main experiment of this paper, in Table 2 we compare the CDB-Net with other baselines using raw i-vectors. As can be seen, the performance of the i-vector pipeline is very poor without LDA-WCCN projection step. This shows the importance of the post-processing for i-vector features. Also, looking at the other baseline NNs, it can be seen that all the other baseline networks also could not achieve a F-measure of more than 41.46%. This show that similar to i-vector pipeline, post-processing is very effective to process i-vector features using NNs optimized with CCE loss function. The proposed CDB-Net could achieve the performance of **57.86%** and outperformed all the baselines. The good performance of CDB-Net reveals that even though no weight initialization using LDA-WCCN projections were used, also i-vectors were not processed with such linear projections, using cosine loss function was very effective to optimize the network. From the second experiment, we can observe that changing the loss function from CCE to cosine, the NN's behavior changes and it can find much better optimum points which leads to achieving much higher performances.

In our final complimentary experiment (exp. 3) we studied the effect of weight initialization with LDA-WCCN projection matrix in the hidden layer. In Table 3 it can be seen that as expected, by initializing the weight of the hidden layer with the projection matrices of LDA-WCCN, all the baseline methods and the proposed method achieved similar performances to exp. 1 that processed i-vectors were used.

By looking at 3 baselines used in experiment 3, it can be seen that the networks which previously did not perform well with raw i-vectors, now can perform much better if the hidden layer's weight matrix initializes with the LDA-WCCN projection matrix. Even though the results are not surprising as we observed from the performance of our baseline NNs in experiment 1, we learned that it is not necessary to only use processed i-vectors to achieve good performances with NNs. By a right initialization, the performance of a CCE-optimized NN can be significantly improved.

Table 1: ***Experiment 1**- Artist recognition F-measure using **processed (LDA-WCCN projected) i-vector features** and different methods. The method marked with an asterisk (∗) is the i-vector pipeline.*

method	in. dim.	hid. neu.	out. layer	F1 (%)
*CS	49	–	–	56.17
CCE-REC-Net	49	49	SM	54.12
CCE-TAN-Net	49	49	SM	57.45
CCE-LIN-Net	49	49	SM	**59.77**
CDB-Net	49	49	LIN	58.24

Table 2: ***Experiment 2**- Artist recognition F-measure using **raw i-vector features** and different methods.*

method	in. dim.	hid. neu.	out. layer	F1 (%)
CS	400	–	–	26.99
CCE-REC-Net	400	400	SM	37.10
CCE-TAN-Net	400	400	SM	40.26
CCE-LIN-Net	400	400	SM	41.46
CDB-Net	400	400	LIN	**57.86**

Table 3: ***Experiment 3**- Artist recognition F-measure using **raw i-vector features** with **LDA-WCCN weight initialization** for different methods.*

method	in. dim.	hid. neu.	out. layer	F1 (%)
CCE-REC-Net	400	49	SM	53.33
CCE-TAN-Net	400	49	SM	57.14
CCE-LIN-Net	400	49	SM	**58.41**
CDB-Net	400	49	LIN	56.89

6. CONCLUSION

Our experiment results (exp. 1) suggest that feed-forward NNs can be used as a classifier with processed i-vectors and achieve the performance of i-vector pipeline. Also in exp. 2 we showed that the

same NNs that performed well with processed i-vectors, are unable to achieve good performances with raw i-vectors and the performance of these NNs drops significantly when the post-processing step is removed.

To tackle this problem, we introduced a NN with a cosine-distance loss function and linear dense layers. We showed that this network can achieve the performance of the NNs that used processed i-vectors. It demonstrate that our network has the ability to learn similar projections to LDA-WCCN which significantly improve the i-vector representation for music artist recognition.

Our complimentary experiment results (exp. 3) suggest we can improve the performance of the feed-forward NNs by initializing their hidden layer's weights using LDA-WCCN projection matrix.

7. ACKNOWLEDGMENTS

This work was supported by the Austrian Science Fund (FWF) under grant no. Z159 (Wittgenstein Award) and by the Austrian Ministry for Transport, Innovation and Technology, the Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

8. REFERENCES

[1] Najim Dehak, Patrick Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front-end factor analysis for speaker verification," *Audio, Speech, and Language Processing, IEEE Transactions on*, 2011.

[2] Rui Xia and Yang Liu, "Using i-vector space model for emotion recognition.," in *INTERSPEECH*, 2012.

[3] Najim Dehak, Pedro A Torres-Carrasquillo, Douglas A Reynolds, and Reda Dehak, "Language recognition via i-vectors and dimensionality reduction.," in *INTERSPEECH*. Citeseer, 2011.

[4] Benjamin Elizalde, Howard Lei, and Gerald Friedland, "An i-vector representation of acoustic environments for audio-based video event detection on user generated content," in *ISM*. IEEE, 2013.

[5] Anna M Kruspe, "Improving singing language identification through i-vector extraction," in *DAFx*, 2011.

[6] Hamid Eghbal-Zadeh, Markus Schedl, and Gerhard Widmer, "Timbral modeling for music artist recognition using i-vectors," in *EUSIPCO*, 2015.

[7] Hamid Eghbal-zadeh, Bernhard Lehner, Markus Schedl, and Gerhard Widmer, "I-vectors for timbre-based music similarity and music artist classification," in *ISMIR*, 2015.

[8] Daniel PW Ellis, "Classifying music audio with timbral and chroma features," in *ISMIR*, 2007.

[9] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balázs Kégl, "Aggregate features and adaboost for music classification," *Machine learning*, 2006.

[10] Pavel P. Kuksa, "Efficient multivariate kernels for sequence classification," *CoRR*, 2014.

[11] Honglak Lee, Peter Pham, Yan Largman, and Andrew Y Ng, "Unsupervised feature learning for audio classification using convolutional deep belief networks," in *Advances in neural information processing systems*, 2009, pp. 1096–1104.

[12] Sander Dieleman, Philémon Brakel, and Benjamin Schrauwen, "Audio-based music classification with a pretrained convolutional network," in *ISMIR*, 2011.

[13] Philippe Hamel and Douglas Eck, "Learning features from music audio with deep belief networks.," in *ISMIR*. Utrecht, The Netherlands, 2010.

[14] Hamid Eghbal-Zadeh and Gerhard Widmer, "Noise robust music artist recognition using i-vector features," in *ISMIR*, 2016.

[15] Yun Lei, Luciana Ferrer, Moray McLaren, et al., "A novel scheme for speaker recognition using a phonetically-aware deep neural network," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.

[16] Patrick Kenny, Vishwa Gupta, Themos Stafylakis, P Ouellet, and J Alam, "Deep neural networks for extracting baum-welch statistics for speaker recognition," in *Proc. Odyssey*, 2014.

[17] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Jorge Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.

[18] Pavel Matejka, Le Zhang, Tim Ng, HS Mallidi, Ondrej Glembek, Jeff Ma, and Bing Zhang, "Neural network bottleneck features for language identification," *Proc. of IEEE Odyssey*, 2014.

[19] Albert Jiménez Sanfiz, "Deep neural networks for channel compensated i-vectors in speaker recognition," *BA Thesis, Universitat Politécnica De Catalunya*, 2014.

[20] Omid Ghahabi and Juan Hernando, "Deep belief networks for i-vector based speaker recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014.

[21] Omid Ghahabi and Javier Hernando, "Deep learning for single and multi-session i-vector speaker recognition," *arXiv preprint arXiv:1512.02560*, 2015.

[22] Patrick Kenny, "Joint factor analysis of speaker and session variability: Theory and algorithms," *CRIM, Montreal,(Report) CRIM-06/08-13*, 2005.

[23] Bernhard Scholkopft and Klaus-Robert Mullert, "Fisher discriminant analysis with kernels," *Neural networks for signal processing IX*, 1999.

[24] Andrew O Hatch and Andreas Stolcke, "Generalized linear kernels for one-versus-all classification: application to speaker recognition," *Proc. Int. Conf. Acoust. Speech and Signal Process.*, 2006.

[25] Daniel Garcia-Romero and Carol Y Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems.," in *INTERSPEECH*, 2011.

[26] Najim Dehak, Reda Dehak, James R Glass, Douglas A Reynolds, and Patrick Kenny, "Cosine similarity scoring without score normalization techniques.," in *Odyssey*, 2010.

[27] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011.

[28] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, 2014.

[30] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah, "Signature verification using a siamese time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, 1993.

[31] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere, "The million song dataset," in *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*. University of Miami, 2011.

[32] Tristan Jehan and Davis DesRoches, "Analyzer documentation," *The Echo Nest*, 2011.

[33] Seyed Omid Sadjadi, Malcolm Slaney, and Larry Heck, "Msr identity toolbox-a matlab toolbox for speaker recognition research," *Microsoft CSRC*, 2013.

[34] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, SK Sønderby, D Nouri, D Maturana, M Thoma, E Battenberg, J Kelly, et al., "Lasagne: First release," *Zenodo: Geneva, Switzerland*, 2015.

[35] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, 2016.

[36] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, 2010.

[37] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[38] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

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

[40] Mohamad Hasan Bahari, Rahim Saeidi, David Van Leeuwen, et al., "Accent recognition using i-vector, gaussian mean supervector and gaussian posterior probability supervector for spontaneous telephone speech," in *ICASSP*. IEEE, 2013.

[41] Andrew O Hatch, Sachin S Kajarekar, and Andreas Stolcke, "Within-class covariance normalization for svm-based speaker recognition.," in *INTERSPEECH*, 2006.