# Deep Linear Discriminant Analysis

**Matthias Dorfer, Rainer Kelz & Gerhard Widmer** [*]
Department of Computational Perception
Johannes Kepler University Linz
Linz, 4040, AUT
{matthias.dorfer, rainer.kelz, gerhard.widmer}@jku.at

## Abstract

We introduce Deep Linear Discriminant Analysis (*DeepLDA*) which learns linearly separable latent representations in an end-to-end fashion. Classic LDA extracts features which preserve class separability and is used for dimensionality reduction for many classification problems. The central idea of this paper is to put LDA on top of a deep neural network. This can be seen as a non-linear extension of classic LDA. Instead of maximizing the likelihood of target labels for individual samples, we propose an objective function that pushes the network to produce feature distributions which: (a) have low variance within the same class and (b) high variance between different classes. Our objective is derived from the general LDA eigenvalue problem and still allows to train with stochastic gradient descent and back-propagation. For evaluation we test our approach on three different benchmark datasets (MNIST, CIFAR-10 and STL-10). DeepLDA produces competitive results on MNIST and CIFAR-10 and outperforms a network trained with categorical cross entropy (having the same architecture) on a supervised setting of STL-10.

## 1 Introduction

Linear Discriminant Analysis (LDA) is a method from multivariate statistics which seeks to find a linear projection of high-dimensional observations into a lower-dimensional space (Fisher, 1936). When its preconditions are fulfilled, LDA allows to define optimal linear decision boundaries in the resulting latent space. The aim of this paper is to exploit the beneficial properties of classic LDA (low intra class variability, hight inter-class variability, optimal decision boundaries) by reformulating its objective to learn linearly separable representations based on a deep neural network (DNN).

Recently, methods related to LDA achieved great success in combination with deep neural networks. Andrew et al. published a deep version of Canonical Correlation Analysis (DCCA) (Andrew et al., 2013). In their evaluations, DCCA is used to produce correlated representations of multi-modal input data of simultaneously recorded acoustic and articulatory speech data. Clevert et al. propose Rectified Factor Networks (RFNs) which are a neural network interpretation of classic factor analysis (Clevert et al., 2015). RFNs are used for unsupervised pre-training and help to improve classification performance on four different benchmark datasets. A similar method called PCANet – as well as an LDA based variation – was proposed by Chan et al. (2015). PCANet can be seen as a simple unsupervised convolutional deep learning approach. The method proceeds with cascaded Principal Component Analysis (PCA), binary hashing and block histogram computations. However, one crucial bottleneck of their approach is its limitation to very shallow architectures (two stages) (Chan et al., 2015).

Stuhlsatz et. al. already picked up the idea of combining LDA with a neural networks and proposed a generalized version of LDA (Stuhlsatz et al., 2012). Their approach starts with pre-training a stack of restricted Boltzmann machines. In a second step, the pre-trained model is fine-tuned with respect to a linear discriminant criterion. LDA has the disadvantage that it overemphasises large distances at the cost of confusing neighbouring classes. In (Stuhlsatz et al., 2012) this problem is tackled by a heuristic weighting scheme for computing the within-class scatter matrix required for LDA optimization.

---

[*] http://www.cp.jku.at/

## 1.1 MAIN IDEA OF THIS PAPER

The approaches mentioned so far all have in common that they are based on well established methods from multivariate statistics. Inspired by their work, we propose an end-to-end DNN version of LDA - namely *Deep Linear Discriminant Analysis* (*DeepLDA*).

Deep learning has become the state of the art in automatic feature learning and replaced existing approaches based on hand engineered features in many fields such as object recognition (Krizhevsky et al., 2012). DeepLDA is motivated by the fact that when the preconditions of LDA are met, it is capable of finding linear combinations of the input features which allow for optimal linear decision boundaries. In general, LDA takes features as input. The intuition of our method is to use LDA as an objective on top of a powerful feature learning algorithm. Instead of maximizing the likelihood of target labels for individual samples, we propose an LDA eigenvalue-based objective function that pushes the network to produce discriminative feature distributions. The parameters are optimized by back-propagating the error of an LDA-based objective through the entire network. We tackle the feature learning problem by focusing on directions in the latent space with smallest discriminative power. This replaces the weighting scheme of (Stuhlsatz et al., 2012) and allows to operate on the original formulation of LDA. We expect that DeepLDA will produce linearly separable hidden representations with similar discriminative power in all directions of the latent space. Such representations should also be related with a high classification potential of the respective networks. The experimental classification results reported below will confirm this positive effect on classification accuracy, and two additional experiments (Section 5) will give us some first qualitative confirmation that the learned representations show the expected properties.

The reminder of the paper is structured as follows. In Section 2 we provide a general formulation of a DNN. Based on this formulation we introduce DeepLDA, a non-linear extension to classic LDA in Section 3. In Section 4 we experimentally evaluate our approach on three benchmark datasets. Section 5 provides a deeper insight into the structure of DeepLDA's internal represenations. In Section 6 we conclude the paper.

## 2 DEEP NEURAL NETWORKS

As the proposed model is built on top of a DNN we briefly describe the training paradigm of a network used for classification problems such as object recognition.

A neural network with $P$ hidden layers is represented as a non-linear function $f(\Theta)$ with model parameters $\Theta = \{\Theta_1, ..., \Theta_P\}$. In the supervised setting we are additionally given a set of $N$ train samples $\mathbf{x}_1, ...\mathbf{x}_N$ along with corresponding classification targets $t_1, ...t_N \in \{1, ..., C\}$. We further assume that the network output $\mathbf{p}_i = (p_{i,1}, ..., p_{i,C}) = f(\mathbf{x}_i, \Theta)$ is normalized by the softmax-function to obtain class (pseudo-)probabilities. The network is then optimized using Stochastic Gradient Descent (SGD) with the goal of finding an optimal model parametrization $\Theta$ with respect to a certain loss function $l_i(\Theta) = l(f(\mathbf{x}_i, \Theta), t_i)$.

$$\Theta = \arg\min_{\Theta} \frac{1}{N} \sum_{i=1}^{N} l_i(\Theta) \tag{1}$$

For multi-class classification problems, Categorical-Cross-Entropy (CCE) is a commonly used optimization target and formulated for observation $\mathbf{x}_i$ and target label $t_i$ as follows

$$l_i(\Theta) = -\sum_{j=1}^{C} y_{i,j} log(p_{i,j}) \tag{2}$$

where $y_{i,j}$ is 1 if observation $\mathbf{x}_i$ belongs to class $t_i$ ($j = t_i$) and 0 otherwise. In particular, the CCE tries to maximize the likelihood of the target class $t_i$ for each of the individual training examples $\mathbf{x}_i$ under the model with parameters $\Theta$. Figure 1a shows a sketch of this general network architecture.

We would like to emphasize that objectives such as CCE do not impose any direct constraints – such as linear separability – on the latent space representation.

(a) The output of the network gets normalized by a soft max layer to form valid probabilities. The CCE objective maximizes the likelihood of the target class under the model.

(b) On the topmost hidden layer we compute an LDA which produces corresponding eigenvalues. The optimization target is to maximize those eigenvalues.
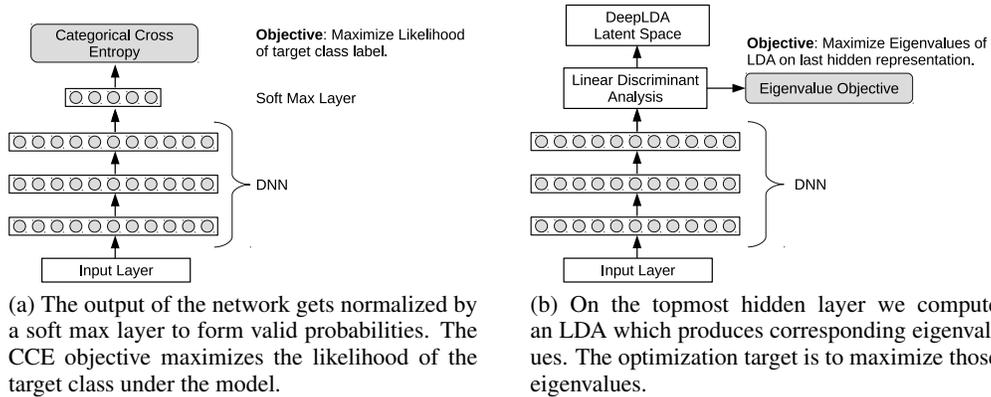
Figure 1: Schematic sketch of a DNN and DeepLDA. For both architectures the input data is first propagated through the layers of the DNN. However, the final layer and the optimization target are different.

## 3 DEEP LINEAR DISCRIMINANT ANALYSIS (DEEPLDA)

In this section we first provide a general introduction to LDA. Based on this introduction we propose DeepLDA, which optimizes an LDA-based optimization target in an end-to-end DNN fashion. Finally we describe how DeepLDA is used to predict class probabilities of unseen test samples.

### 3.1 LINEAR DISCRIMINANT ANALYSIS

Let $\mathbf{x}_1, ..., \mathbf{x}_N = \mathbf{X} \in \mathbb{R}^{N \times d}$ denote a set of $N$ samples belonging to $C$ different classes $c \in \{1, ..., C\}$. The input representation $\mathbf{X}$ can either be hand engineered features, or hidden space representations $\mathbf{H}$ produced by a DNN (Andrew et al., 2013). LDA seeks to find a linear projection $\mathbf{A} \in \mathbb{R}^{l \times d}$ into a lower $l$-dimensional subspace $L$ where $l = C - 1$. The resulting linear combinations of features $\mathbf{x}_i \mathbf{A}^T$ are maximally separated in this space (Fisher, 1936). The LDA objective to find projection matrix $\mathbf{A}$ is formulated as:

$$\arg\max_{\mathbf{A}} \frac{|\mathbf{A}S_b\mathbf{A}^T|}{|\mathbf{A}S_w\mathbf{A}^T|} \tag{3}$$

where $\mathbf{S}_b$ is the between scatter matrix and defined via the total scatter matrix $\mathbf{S}_t$ and within scatter matrix $\mathbf{S}_w$ as $\mathbf{S}_b = \mathbf{S}_t - \mathbf{S}_w$. $\mathbf{S}_w$ is defined as the mean of the $C$ individual class covariance matrices $\mathbf{S}_c$ (Equation (4) and (5)). $\bar{\mathbf{X}}_c = \mathbf{X}_c - \mathbf{m}_c$ are the mean-centered observations of class $c$ with per-class mean vector $\mathbf{m}_c$ ($\bar{\mathbf{X}}$ is defined analogously for the entire population $\mathbf{X}$). The total scatter matrix $\mathbf{S}_t$ is the covariance matrix over the entire population of observations $\mathbf{X}$.

$$\mathbf{S}_c = \frac{1}{N_c - 1} \bar{\mathbf{X}}_c^T \bar{\mathbf{X}}_c \tag{4}$$

$$\mathbf{S}_w = \frac{1}{C} \sum_c \mathbf{S}_c \tag{5}$$

$$\mathbf{S}_t = \frac{1}{N - 1} \bar{\mathbf{X}}^T \bar{\mathbf{X}} \tag{6}$$

The linear combinations that maximize the objective in Equation (3) maximize the ratio of between- and within-class scatter also reffered to as separation. This means in particular that a set of projected observations of the same class show low variance, whereas the projections of observations of different classes have high variance in the resulting space $L$. To find the optimum solution for Equation (3) one has to solve the general eigenvalue problem $\mathbf{S}_b\mathbf{e} = \mathbf{v}\mathbf{S}_w\mathbf{e}$. The projection matrix $\mathbf{A}$ is the set of eigenvectors $\mathbf{e}$ associated with this problem. In the following sections we will cast LDA as an objective function for DNN.

3

## 3.2 DeepLDA Model Configuration

Figure 1b shows a schematic sketch of DeepLDA. Instead of sample-wise optimization of the CCE loss on the predicted class probabilities (see Section 2) we put an *LDA-layer* on top of the DNN. This means in particular that we do not penalize the misclassification of individual samples. Instead we try to produce features that show a low intra-class and high inter-class variability. We address this maximization problem by a modified version of the general LDA eigenvalue problem proposed in the following section. In contrast to CCE, DeepLDA optimization operates on the properties of the distribution parameters of the hidden representation produced by the neural net. As eigenvalue optimization is tied to its corresponding eigenvectors (a linear projection matrix), DeepLDA can be also seen as a special case of a dense layer.

## 3.3 Modified DeepLDA Optimization Target

Based on Section 3.1 we reformulate the LDA objective to be suitable for a combination with deep learning. As already discussed by Stuhlsatz et al. (2012) and Lu et al. (2005) the estimation of $\mathbf{S}_w$ overemphasises high eigenvalues whereas small eigenvalues are estimated as too low. To weaken this effect, Friedman (1989) proposed to regularize the within scatter matrix by adding a multiple of the identity matrix $\mathbf{S}_w + \lambda\mathbf{I}$. Adding the identity matrix has the second advantage of stabilizing small eigenvalues. The resulting eigenvalue problem is then formulated as

$$\mathbf{S}_b\mathbf{e}_i = v_i(\mathbf{S}_w + \lambda\mathbf{I})\mathbf{e}_i \tag{7}$$

where $\mathbf{e} = \mathbf{e}_1, ..., \mathbf{e}_{C-1}$ are the resulting eigenvectors and $\mathbf{v} = v_1, ...v_{C-1}$ the corresponding eigenvalues. Once the problem is solved, each eigenvalue $v_i$ quantifies the amount of discriminative variance (separation) in direction of the corresponding eigenvector $\mathbf{e}_i$. If one would like to combine this objective with a DNN the optimization target would be the maximization of the individual eigenvalues. In particular, we expect that maximizing the individual eigenvalues – which reflect the separation in the respective eigenvector directions – leads to a maximization of the discriminative power of the neural net. In our initial experiments we started to formulate the objective as:

$$\arg\max_{\Theta} \frac{1}{C-1} \sum_{i=1}^{C-1} v_i \tag{8}$$

One problem we discovered with the objective in Equation (8) is that the net favours trivial solutions e. g. maximize only the largest eigenvalue as this produces the highest reward. In terms of classification this means that it maximizes the distance of classes that are already separated at the expense of – potentially non-separated – neighbouring classes. This was already discussed by (Stuhlsatz et al., 2012) and tackled by a weighted computation of the between scatter matrix $\mathbf{S}_b$.

We propose a different solution to this problem and address it by focusing our optimization on the smallest of all $C-1$ available eigenvalues. In particular we consider only the $k$ eigenvalues that do not exceed a certain threshold for variance maximization:

$$\arg\max_{\Theta} \frac{1}{k} \sum_{i=1}^{k} v_i \quad \text{with} \ \ \{v_1, ..., v_k\} = \{v_j | v_j < \min\{v_1, ..., v_{C-1}\} + \epsilon\} \tag{9}$$

The intuition behind this formulation is to learn a net parametrization that pushes as much discriminative variance as possible into all of the $C-1$ available feature dimensions.

We would like to underline that this formulation allows to train DeepLDA networks with backpropagation in end-to-end fashion (see Appendix for a derivative of the loss functions's gradient). Our models are optimized with the Nesterov momentum version of mini-batch SGD. Related methods already showed that mini-batch learning on distribution parameters (in this case covariance matrices) is feasible if the batch-size is sufficiently large to be representative for the entire population (Wang et al., 2015a;b).

## 3.4 Classification by DeepLDA

This section describes how the most likely class label is assigned to an unseen test sample $\mathbf{x}_t$ once the network is trained and parametrized. In a first step we compute the topmost hidden representation

$\mathbf{H}$ on the entire training set $\mathbf{X}$. On this hidden representation we compute the LDA as described in Section 3.1 and 3.3 producing the corresponding eigenvectors $\mathbf{e} = \{\mathbf{e}_i\}_{i=1}^{C-1}$ which form the LDA projection matrix $\mathbf{A}$. We would like to emphasize that since the parameters of the network are fixed at this stage we make use of the entire training set to provide a stable estimate of the LDA projection. Based on $\mathbf{A}$ and the per-class mean hidden representations $\bar{\mathbf{H}}_c = (\bar{\mathbf{h}}_1^T, ..., \bar{\mathbf{h}}_C^T)$ the distances of sample $\mathbf{h}_t$ to the linear decision hyperplanes (Friedman et al., 2001) are defined as

$$\mathbf{d} = \mathbf{h}_t^T \mathbf{T}^T - \frac{1}{2} diag\left(\bar{\mathbf{H}}_c \mathbf{T}^T\right) \text{ with } \mathbf{T} = \bar{\mathbf{H}}_c \mathbf{A} \mathbf{A}^T \qquad (10)$$

where $\mathbf{T}$ are the decision hyperplane normal vectors. The subtracted term is the bias of the decision functions placing the decision boundaries in between the means of the respective class hidden representations (no class priors included). The vector of class probabilities for test sample $\mathbf{x}_t$ is then computed by applying the logistic function $\mathbf{p}'_c = 1/(1 + \mathrm{e}^{-\mathbf{d}})$ and further normalized by $\mathbf{p}_c = \mathbf{p}'_c / \sum p'_i$ to sum to one. Finally we assign class $i$ with highest probability as $\arg\max_i p_i$ to the unseen test sample $\mathbf{x}_t$.

## 4 EXPERIMENTS

In this section we present an experimental evaluation of DeepLDA on three benchmark data sets – namely MNIST, CIFAR-10 and STL-10 (see Figure 2 for some sample images). We compare the results of DeepLDA with the CCE based optimization target as well as the present state of the art of the respective datasets. In addition, we provide details on the network architectures, hyper parameters and respective training/optimization approaches used in our experiments.



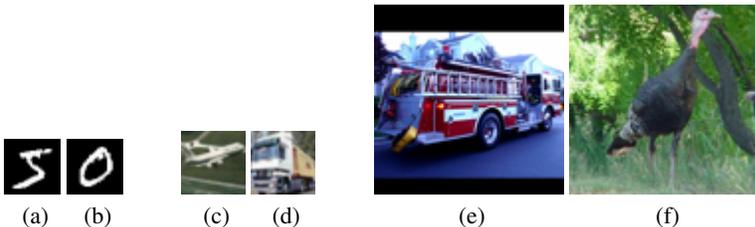|       |       |       |       |       |       |
| (a)   | (b)   | (c)   | (d)   | (e)   | (f)   |

Figure 2: Example images of evaluation data sets (a)(b) MNIST, (c)(d) CIFAR-10, (e)(f) STL-10. The relative size differences between images from the three data sets are kept in this visualization.

### 4.1 EXPERIMENTAL SETUP

The general structure of the networks is similar for all of the three datasets and identical for CIFAR-10 and STL-10. The architecture follows the VGG model with sequences of $3 \times 3$ convolutions (Simonyan & Zisserman, 2014). Instead of a dense classification layer we use global average pooling on the feature maps of the last convolution layer (Lin et al., 2013). We picked this architecture as it leads to well-posed problems for covariance estimation: many samples vs. low feature space dimension. We further apply batch normalization (Ioffe & Szegedy, 2015) after each convolutional layer which (1) helped to increase convergence speed and (2) improved the performance of all our models. Batch normalization has a positive effect on both CCE as well as DeepLDA-based optimization. In Table 1 we outline the structure of our models in detail. All networks are trained using SGD with Nesterov momentum. The initial learning rate is set to 0.1 and the momentum is fixed at 0.9 for all our models. The learning rate is then halved every 25 epochs for CIFAR-10 and STL-10 and every 10 epochs for MNIST. For further regularization we add weight decay with a weighting of 0.0001 on all trainable parameters of the models. The between-class covariance matrix regularization weight $\lambda$ (see Section 3.3) is set to 0.001 and the $\epsilon$-offset for DeepLDA to 1.

One hyper-parameter that varies between the datasets is the batch size used for training DeepLDA. Although a large batch size is desired to get stable covariance estimates it is limited by the amount of memory available on the GPU. The mini-batches for DeepLDA were for MNIST: 1000, for CIFAR-10: 1000 and for STL-10: 200. For CCE training, a batch size of 128 is used for all datasets. The models are trained on an NVIDIA Tesla K40 with 12GB of GPU memory.

Table 1: Model Specifications. BN: Batch Normalization, ReLu: Rectified Linear Activation Function, CCE: Categorical Cross Entropy. The mini-batch sizes of DeepLDA are: MNIST(1000), CIFAR-10(1000), STL-10(200). For CCE training a constant batch size of 128 is used.

| CIFAR-10 and STL-10 | MNIST |
|---|---|
| Input $3 \times 32 \times 32$ ($96 \times 96$) | Input $1 \times 28 \times 28$ |
| $3 \times 3$ Conv(pad-1)-64-BN-ReLu | |
| $3 \times 3$ Conv(pad-1)-64-BN-ReLu | |
| $2 \times 2$ Max-Pooling + Drop-Out(0.25) | |
| $3 \times 3$ Conv(pad-1)-128-BN-ReLu | $3 \times 3$ Conv(pad-1)-96-BN-ReLu |
| $3 \times 3$ Conv(pad-1)-128-BN-ReLu | $3 \times 3$ Conv(pad-1)-96-BN-ReLu |
| $2 \times 2$ Max-Pooling + Drop-Out(0.25) | $2 \times 2$ Max-Pooling + Drop-Out(0.25) |
| $3 \times 3$ Conv(pad-1)-256-BN-ReLu | |
| $3 \times 3$ Conv(pad-1)-256-BN-ReLu | |
| $3 \times 3$ Conv(pad-1)-256-BN-ReLu | |
| $3 \times 3$ Conv(pad-1)-256-BN-ReLu | |
| $2 \times 2$ Max-Pooling + Drop-Out(0.25) | |
| $3 \times 3$ Conv(pad-0)-1024-BN-ReLu | $3 \times 3$ Conv(pad-0)-256-BN-ReLu |
| Drop-Out(0.5) | Drop-Out(0.5) |
| $1 \times 1$ Conv(pad-0)-1024-BN-ReLu | $1 \times 1$ Conv(pad-0)-256-BN-ReLu |
| Drop-Out(0.5) | Drop-Out(0.5) |
| $1 \times 1$ Conv(pad-0)-10-BN-ReLu | $1 \times 1$ Conv(pad-0)-10-BN-ReLu |
| $2 \times 2$ ($10 \times 10$) Global-Average-Pooling | $5 \times 5$ Global-Average-Pooling |
| Soft-Max with CCE or LDA-Layer | |

## 4.2 Experimental Results

We describe the benchmark datasets as well as the pre-processing and data augmentation used for training. We present our results and relate them to the present state of the art for the respective dataset. As DeepLDA is supposed to produce a linearly separable feature space, we also report the results of a linear Support Vector Machine trained on the latent space of DeepLDA (tagged with *LinSVM*). The results of our network architecture trained with CCE are marked as *OurNetCCE*. To provide a complete picture of our experimental evaluation we also show classification results of an LDA on the topmost hidden representation of the networks trained with CCE (tagged with *OurNetCCE(LDA)*).

### 4.2.1 MNIST

The MNIST dataset consists of $28 \times 28$ gray scale images of handwritten digits ranging from 0 to 9. The dataset is structured into 50000 train samples, 10000 validation samples and 10000 test samples. For training we did not apply any pre-processing nor data augmentation. We present results for two different scenarios. In scenario *MNIST-50k* we train on the 50000 train samples and use the validation set to pick the parametrization which produces the best results on the validation set. In scenario *MNIST-60k* we train the model for the same number of epochs as in *MNIST-50k* but also use the validation set for training. Finally we report the accuracy of the model on the test set after the last training epoch. This approach was also applied in (Lin et al., 2013) which produce state of the art results on the dataset.

Table 2 summarizes all results on the MNIST dataset. DeepLDA produces competitive results – having a test set error of $0.29\%$ – although no data augmentation is used. In the approach described in (Graham, 2014) the train set is extended with translations of up to two pixels. We also observe that a linear SVM trained on the learned representation produces comparable results on the test set. It is also interesting that early stopping with best-model-selection (MNIST-50k) performs better than training on MNIST-60k even though 10000 more training examples are available.

Table 2: Comparison of test errors on MNIST

| Method | Test Error |
|---|---|
| NIN + Dropout (Lin et al. (2013)) | 0.47% |
| Maxout (Goodfellow et al. (2013)) | 0.45% |
| DeepCNet(5,60) (Graham (2014)) | 0.31% (train set translation) |
| OurNetCCE(LDA)-50k | 0.39% |
| OurNetCCE-50k | 0.37% |
| OurNetCCE-60k | 0.34% |
| DeepLDA-60k | 0.32% |
| OurNetCCE(LDA)-60k | 0.30% |
| DeepLDA-50k | **0.29**% |
| DeepLDA-50k(LinSVM) | **0.29**% |

Table 3: Comparison of test errors on CIFAR-10

| Method | Test Error |
|---|---|
| NIN + Dropout (Lin et al. (2013)) | 10.41% |
| Maxout (Graham (2014)) | 9.38% |
| NIN + Dropout (Lin et al. (2013)) | 8.81% (data augmentation) |
| DeepCNINet(5,300) (Graham (2014)) | **6.28**% (data augmentation) |
| DeepLDA(LinSVM) | 7.58% |
| DeepLDA | 7.29% |
| OurNetCCE(LDA) | 7.19% |
| OurNetCCE | 7.10% |

### 4.2.2 CIFAR-10

The CIFAR-10 dataset consists of tiny $32 \times 32$ natural RGB images containing samples of 10 different classes. The dataset is structured into 50000 train samples and 10000 test samples. We pre-processed the dataset using global contrast normalization and ZCA whitening as proposed by Goodfellow et al. (2013). During training we only apply random left-right flips on the images – no additional data augmentation is used. In training, we follow the same procedure as described for the MNIST dataset above to make use of the entire 50000 train images.

Table 3 summarizes our results and relates them to the present state of the art. Both OurNetCCE and DeepLDA produce state of the art results on the dataset when no data augmentation is used. Although DeepLDA performs slightly worse than CCE it is capable of producing competitive results on CIFAR-10.

### 4.2.3 STL-10

Like CIFAR-10, the STL-10 data set contains natural RGB images of 10 different object categories. However, with $96 \times 96$ pixels the size of the images is larger, and the training set is considerably smaller, with only 5000 images. The test set consists of 8000 images. In addition, STL-10 contains 100000 unlabelled images but we do not make use of this additional data at this point as our approach is fully supervised. For that reason we first perform an experiment (*Method-4k*) where we do not follow the evaluation strategy described in (Coates et al., 2011), where models are trained on 1000 labeled and 100000 unlabeled images. Instead, we directly compare CCE and DeepLDA in a fully supervised setting. As with MNIST-50k we train our models on 4000 of the train images and use the rest (1000 images) as a validation set to pick the best performing parametrization. The results on the *Method-4k-Setting* of STL-10 are presented in the top part of Table 4. Our model trained with CCE achieves an accuracy of 78.39%. The same architecture trained with DeepLDA improves the test set accuracy by more than 3 percentage points and achieves 81.46%. In our second experiment (*Method-1k*) we follow the evaluation strategy described in (Coates et al., 2011) but without using the unlabelled data. We train our models on the 10 pre-defined folds (each fold contains 1000 train images) and report the average accuracy on the test set. The model optimized with CCE

Table 4: Comparison of test set accuracy on a purely supervised setting of STL-10. (*Method-4k*: 4000 train images, *Method-1k*: 1000 train images.)

| Method-4k | Test Accuracy-4k |
|---|---|
| OurNetCCE(LDA)-4k | 78.50% |
| OurNetCCE-4k | 78.84% |
| DeepLDA-4k | 81.16% |
| DeepLDA(LinSVM)-4k | **81.40**% |

| Method-1k | Test Accuracy-1k |
|---|---|
| SWWAE (Zhao et al. (2015)) | 57.45% |
| SWWAE (Zhao et al. (2015)) | 74.33% (semi-supervised) |
| DeepLDA(LinSVM)-1k | 55.92% |
| OurNetCCE-1k | 57.44% |
| OurNetCCE(LDA)-1k | 59.48% |
| DeepLDA-1k | **66.97**% |

(*OurNetCCE-1k*) achieves $57.44\%$ accuracy on the test set which is in line with the supervised results reported in (Zhao et al., 2015).

Our model trained with DeepLDA achieves $66.97\%$ average test set accuracy. This is a performance gain of $9.53\%$ in contrast to CCE and it shows that the advantage of DeepLDA compared to CCE becomes even more apparent when the amount of labeled data is low. When comparing *DeepLDA-1k* with LDA applied on the features computed by a network trained with CCE (*OurNetCCE(LDA)-1k*, $59.48\%$), we find that the end-to-end trained LDA-features outperform the standard CCE approach. A direct comparison with state of the art results as reported in (Zhao et al., 2015; Swersky et al., 2013; Dosovitskiy et al., 2014) is not possible because these models are trained under semi-supervised conditions using both unlabelled and labelled data. However, the results suggest that a combination of DeepLDA with methods such as proposed by Zhao et al. (2015) is a very promising future direction.

## 5 INVESTIGATONS ON DEEPLDA AND DISCUSSIONS

In this section we provide deeper insights into the representations learned by DeepLDA. We experimentally investigate the eigenvalue structure of representations learned by DeepLDA as well as its relation to the classification potential of the respective networks.

### 5.1 DOES IMAGE SIZE AFFECT DEEPLDA?

DeepLDA shows its best performance on the STL-10 dataset (*Method-4k*) where it outperforms CCE by 3 percentage points. The major difference between STL-10 and CIFAR-10 – apart from the number of train images – is the size of the contained images (see Figure 2 to get an impression of the size relations). To get a deeper insight into the influence of this parameter we run the following additional experiment: (1) we create a downscaled version of the STL-10 dataset with the same image dimensions as CIFAR-10 ($32 \times 32$). (2) We repeat the experiment (*Method-4k*) described in Section 4.2.3 on the downscaled $32 \times 32$ dataset. The results are presented in Figure 3, as curves showing the evolution of train and validation accuracy during training. As expected, downscaling reduces the performance of both CCE and DeepLDA. We further observe that DeepLDA performs best when trained on larger images and has a disadvantage on the small images. However, a closer look at the results on CIFAR-10 (CCE: $7.10\%$ error, DeepLDA: $7.29\%$ error, see Table 3) suggests that this effect is compensated when the training set size is sufficiently large. As a reminder: CIFAR-10 contains 50000 train images in contrast to STL-10 with only 4000 samples.

### 5.2 EIGENVALUE STRUCTURE OF DEEPLDA REPRESENTATIONS

DeepLDA optimization does not focus on maximizing the target class likelihood of individual samples. As proposed in Section 3 we encourage the net to learn feature representations with discrimi-
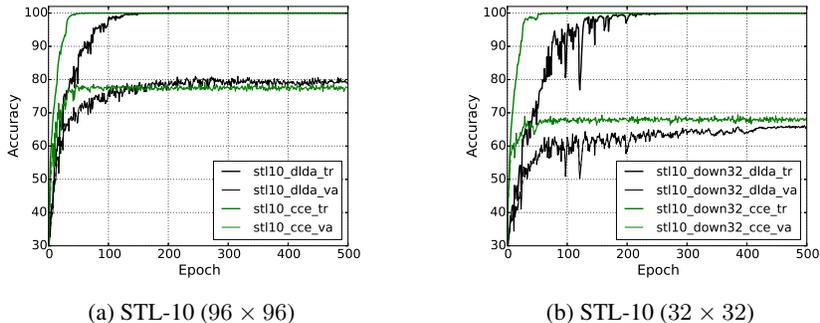
(a) STL-10 (96 × 96)

(b) STL-10 (32 × 32)

Figure 3: Comparison of the learning curves of DeepLDA on the original STL-10 dataset (*Method-4k*) with image size 96 × 96 and its downscaled 32 × 32 version.

native distribution parameters (within and between class scatter). We achieve this by exploiting the eigenvalue structure of the general LDA eigenvalue problem and use it as a deep learning objective. Figure 4a shows the evolution of train and test set accuracy of STL-10 along with the mean value of all eigenvalues in the respective training epoch. We observe the expected natural correlation between the magnitude of explained "discriminative" variance (separation) and the classification potential of the resulting representation. In Figure 4b we show how the individual eigenvalues increase during training. Note that in Epoch 0 almost all eigenvalues (1-7) start at a value of 0. This emphasizes the importance of the design of our objective function (compare Equation (9)) which allows to draw discriminability into the lower dimensions of the eigen-space. In Figure 4c we additionally compare the eigenvalue structure of the latent representation produced by DeepLDA with CCE based training. Again results show that DeepLDA helps to distribute the discriminative variance more equally over the available dimensions. To give the reader an additional intuition on the learned representations we visualize the latent space of STL-10 in our supplemental materials on the final page of this paper.



(a) Eigenvalues vs. Accuracy

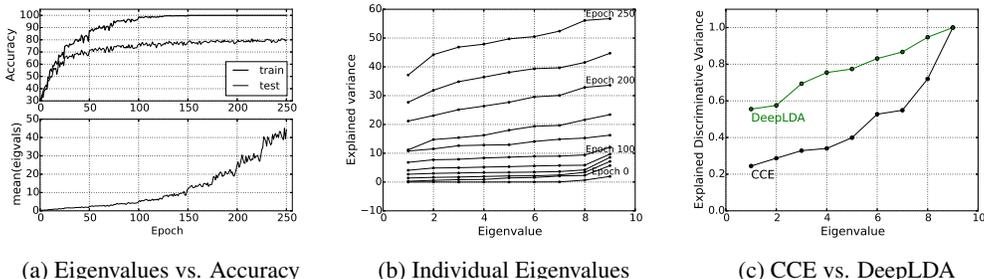(b) Individual Eigenvalues

(c) CCE vs. DeepLDA

Figure 4: The figure investigates the eigenvalue structure of the general LDA eigenvalue problem during training a DeepLDA network on STL-10 (*Method-4k*). (a) shows the evolution of classification accuracy along with the magnitude of explained discriminative variance (separation) in the latent representation of the network. (b) shows the evolution of individual eigenvalues during training. In (c) we compare the eigenvalue structure of a net trained with CCE and DeepLDA (for better comparability we normalized the maximum eigenvalue to one).

## 6 CONCLUSION

We have presented DeepLDA, a deep neural network interpretation of linear discriminant analysis. DeepLDA learns linearly separable latent representations in an end-to-end fashion by maximizing the eigenvalues of the general LDA eigenvalue problem. Our modified version of the LDA optimization target pushes the network to distribute discriminative variance in all dimensions of the latent

feature space. Experimental results show that representations learned with DeepLDA are discriminative and have a positive effect on classification accuracy. Our DeepLDA models achieve competitive results on MNIST and CIFAR-10 and outperform CCE in a fully supervised setting of STL-10 by more than $9\%$ test set accuracy. The results and further investigations suggest that DeepLDA performs best, when applied to reasonably-sized images (in the present case $96 \times 96$ pixel). Finally, we see DeepLDA as a specific instance of a general fruitful strategy: exploit well-understood machine learning or classification models such as LDA with certain desirable properties, and use deep networks to learn representations that provide optimal conditions for these models.

REFERENCES

Andrew, Galen, Arora, Raman, Bilmes, Jeff, and Livescu, Karen. Deep canonical correlation analysis. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1247–1255, 2013.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

Chan, Tsung-Han, Jia, Kui, Gao, Shenghua, Lu, Jiwen, Zeng, Zinan, and Ma, Yi. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24(12): 5017–5032, 2015. doi: 10.1109/TIP.2015.2475625. URL http://dx.doi.org/10.1109/TIP.2015.2475625.

Clevert, Djork-Arné, Unterthiner, Thomas, Mayr, Andreas, Ramsauer, Hubert, and Hochreiter, Sepp. Rectified factor networks. In *Advances in neural information processing systems*, 2015.

Coates, Adam, Ng, Andrew Y, and Lee, Honglak. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pp. 215–223, 2011.

de Leeuw, Jan. Derivatives of generalized eigen systems with applications. 2007.

Dieleman, Sander, Schlueter, Jan, Raffel, Colin, Olson, Eben, Snderby, Sren Kaae, Nouri, Daniel, Maturana, Daniel, Thoma, Martin, Battenberg, Eric, Kelly, Jack, Fauw, Jeffrey De, Heilman, Michael, diogo149, McFee, Brian, Weideman, Hendrik, takacsg84, peterderivaz, Jon, instagibbs, Rasul, Dr. Kashif, CongLiu, Britefury, and Degrave, Jonas. Lasagne: First release., August 2015. URL http://dx.doi.org/10.5281/zenodo.27878.

Dosovitskiy, Alexey, Springenberg, Jost Tobias, Riedmiller, Martin, and Brox, Thomas. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 766–774, 2014.

Fisher, Ronald A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Friedman, Jerome H. Regularized discriminant analysis. *Journal of the American statistical association*, 84(405):165–175, 1989.

Goodfellow, Ian J, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. *arXiv preprint arXiv:1302.4389*, 2013.

Graham, Benjamin. Spatially-sparse convolutional neural networks. *CoRR*, abs/1409.6070, 2014. URL http://arxiv.org/abs/1409.6070.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

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

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *CoRR*, abs/1312.4400, 2013. URL http://arxiv.org/abs/1312.4400.

Lu, Juwei, Plataniotis, Konstantinos N, and Venetsanopoulos, Anastasios N. Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition. *Pattern Recognition Letters*, 26(2):181–191, 2005.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Stuhlsatz, Andre, Lippel, Jens, and Zielke, Thomas. Feature extraction with deep neural networks by a generalized discriminant analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 23(4):596–608, 2012.

Swersky, Kevin, Snoek, Jasper, and Adams, Ryan P. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, pp. 2004–2012, 2013.

Wang, Weiran, Arora, Raman, Livescu, Karen, and Bilmes, Jeff. On deep multi-view representation learning. In *ICML*, 2015a.

Wang, Weiran, Arora, Raman, Livescu, Karen, and Bilmes, Jeff A. Unsupervised learning of acoustic features via deep canonical correlation analysis. In *Proceedings of ICASSP*, 2015b.

Zhao, Junbo, Mathieu, Michael, Goroshin, Ross, and Lecun, Yann. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015.

## APPENDIX A: GRADIENT OF DEEPLDA-LOSS

To train with back-propagation we provide the partial derivatives of optimization target $l(\mathbf{H})$ proposed in Equation (9) with respect to the topmost hidden representation $\mathbf{H}$ (contains samples as rows and features as columns). As a reminder, the DeepLDA objective focuses on maximizing the $k$ smallest eigenvalues $v_i$ of the generalized LDA eigenvalue problem. In particular, we consider only the $k$ eigenvalues that do not exceed a certain threshold for optimization:

$$l(\mathbf{H}) = \frac{1}{k}\sum_{i=1}^{k} v_i \ \text{ with } \ \{v_1, ..., v_k\} = \{v_j | v_j < \min\{v_1, ..., v_{C-1}\} + \epsilon\} \tag{11}$$

For convenience, we change the subscripts of the scatter matrices to superscripts in this section (e.g. $\mathbf{S}_t \rightarrow \mathbf{S}^t$). $\mathbf{S}^t_{ij}$ addresses the element in row $i$ and column $j$ in matrix $\mathbf{S}^t$. Starting from the formulation of the generalized LDA eigenvalue problem:

$$\mathbf{S}^b \mathbf{e}_i = v_i \mathbf{S}^w \mathbf{e}_i \tag{12}$$

the derivative of eigenvalue $v_i$ with respect to hidden representation $\mathbf{H}$ is defined in (de Leeuw, 2007) as:

$$\frac{\partial v_i}{\partial \mathbf{H}} = \mathbf{e}_i^T \left( \frac{\partial \mathbf{S}^b}{\partial \mathbf{H}} - v_i \frac{\partial \mathbf{S}^w}{\partial \mathbf{H}} \right) \mathbf{e}_i \tag{13}$$

Recalling the definitions of the LDA scatter matrices from Section 3.1:

$$\mathbf{S}^c = \frac{1}{N_c - 1} \bar{\mathbf{X}}_c^T \bar{\mathbf{X}}_c \qquad \mathbf{S}^w = \frac{1}{C} \sum_c \mathbf{S}^c \tag{14}$$

$$\mathbf{S}^t = \frac{1}{N - 1} \bar{\mathbf{X}}^T \bar{\mathbf{X}} \qquad \mathbf{S}^b = \mathbf{S}^t - \mathbf{S}^w \tag{15}$$

we can write the partial derivative of the total scatter matrix $\mathbf{S}_t$ (Andrew et al., 2013; Stuhlsatz et al., 2012) on hidden representation $\mathbf{H}$ as:

$$\frac{\partial \mathbf{S}^t_{ab}}{\partial H_{ij}} = \begin{cases} \frac{2}{N-1}\left(H_{ij} - \frac{1}{N}\sum_n H_{nj}\right) & \text{if } a=j, b=j \\ \frac{1}{N-1}\left(H_{ib} - \frac{1}{N}\sum_n H_{nb}\right) & \text{if } a=j, b\neq j \\ \frac{1}{N-1}\left(H_{ia} - \frac{1}{N}\sum_n H_{na}\right) & \text{if } a\neq j, b=j \\ 0 & \text{if } a\neq j, b\neq j \end{cases} \tag{16}$$

The derivatives for the individual class covariance matrices $\mathbf{S}^c$ are defined analogously to Equation (16) for the $C$ classes and we can write the partial derivatives of $\mathbf{S}^w$ and $\mathbf{S}^b$ with respect to the latent representation $\mathbf{H}$ as:

$$\frac{\partial \mathbf{S}^w_{ab}}{\partial H_{ij}} = \frac{1}{C}\sum_c \frac{\partial \mathbf{S}^c_{ab}}{\partial H_{ij}} \quad \text{and} \quad \frac{\partial \mathbf{S}^b_{ab}}{\partial H_{ij}} = \frac{\partial \mathbf{S}^t_{ab}}{\partial H_{ij}} - \frac{\partial \mathbf{S}^w_{ab}}{\partial H_{ij}} \tag{17}$$

The partial derivative of the loss function introduced in Section 3.3 with respect to hidden state $\mathbf{H}$ is then defined as:

$$\frac{\partial}{\partial \mathbf{H}} \frac{1}{k}\sum_{i=1}^{k} v_i = \frac{1}{k}\sum_{i=1}^{k}\frac{\partial v_i}{\partial \mathbf{H}} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{e}_i^T\left(\frac{\partial \mathbf{S}^b}{\partial \mathbf{H}} - v_i\frac{\partial \mathbf{S}^w}{\partial \mathbf{H}}\right)\mathbf{e}_i \tag{18}$$

## APPENDIX B: DEEPLDA LATENT REPRESENTATION

Figure 5 shows the latent space representations on the STL-10 dataset (Method-4k) as *n-to-n* scatter plots of the latent features on the first 1000 test set samples. We plot the test set samples after projection into the $C - 1$ dimensional DeepLDA feature space. The plot suggest that DeepLDA makes use of all available feature dimensions. An interesting observation is that many of the internal representations are orthogonal to each other (which is an implication of LDA). This of course favours linear decision boundaries.
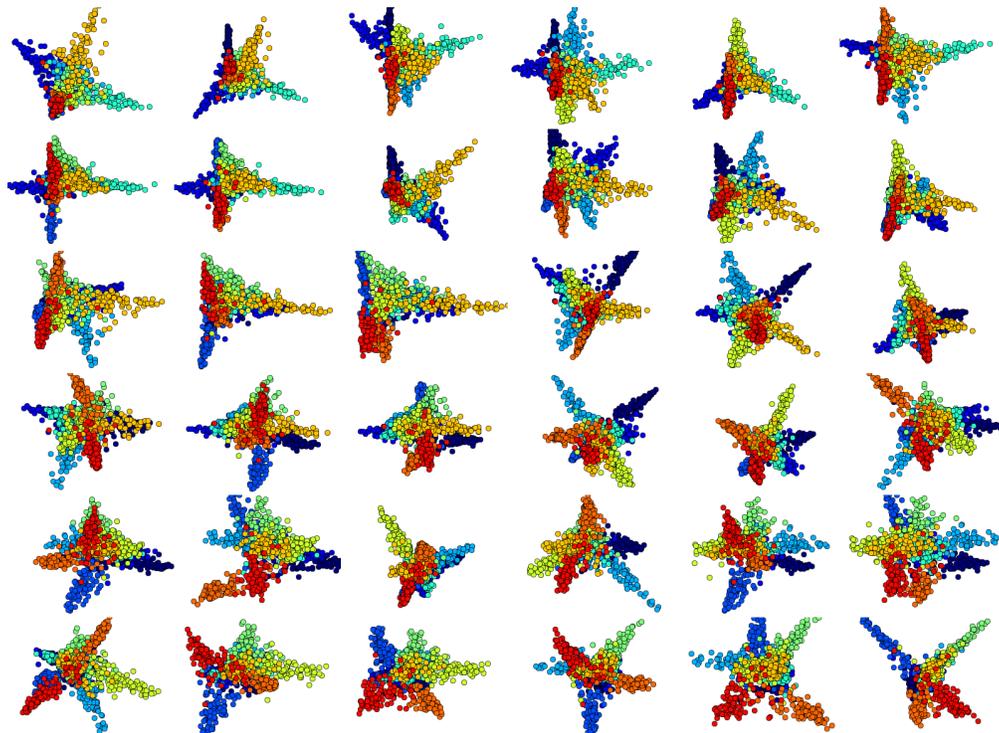


Figure 5: STL-10 latent representation produced by DeepLDA (*n-to-n* scatter plots of the latent features of the first 1000 test set samples. e.g.: top left plot: latent feature 1 vs. latent feature 2).