FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 44 (2019) No. 3
ISSN 0867-6356
DOI: 10.2478/fcds-2019-0015 e-ISSN 2300-3405

sciendo

On Intra-Class Variance for Deep Learning of Classifiers

Rafał Pilarczyk, Władysław Skarbek *

Abstract. A novel technique for deep learning of image classifiers is presented. The learned CNN models higher offer better separation of deep features (also known as embedded vectors) measured by Euclidean proximity and also no deterioration of the classification results by class membership probability. The latter feature can be used for enhancing image classifiers having the classes at the model's exploiting stage different from from classes during the training stage. While the Shannon information of SoftMax probability for target class is extended for mini-batch by the intra-class variance, the trained network itself is extended by the Hadamard layer with the parameters representing the class centers. Contrary to the existing solutions, this extra neural layer enables interfacing of the training algorithm to the standard stochastic gradient optimizers, e.g. AdaM algorithm. Moreover, this approach makes the computed centroids immediately adapting to the updating embedded vectors and finally getting the comparable accuracy in less epochs.

Keywords: deep learning of image classifier, convolutional neural network, Shanon information measure, intra-class variance, image embedding

1. Introduction

Artificial Neural Networks (ANN) are developed in science and engineering since late 1950s – the first report in 1957 (Rosenblatt [12]). Initially, as a algorithmic tool for computer-aided decision making (perceptron), then as a universal mechanism for function approximation (multilayer perceptron) – since 1980s when the error back-propagation was published (cf. Werbos' pioneer paper [20] and Rumelhart et al. [13]) using a low dimensional data for their classification and regression, and nowadays, since about 2010, as Deep Neural Networks (DNN) equipped with specialized operations (e.g. convolutions), operating on large multidimensional signals (also known as tensors) and dozens of processing layers to extract their implicit tensor features.

*Warsaw University of Technology {r.pilarczyk, w.skarbek}@ire.pw.edu.pl

A comprehensive survey of ANN history with a large collection of references can be found in Schmidhuber's article [14].

Deep neural networks algorithms embrace a broad class of ANN algorithms for data model building in order to solve the hard data classification and regression problems. However, now due to unprecedented progress in computing technology, both tasks can accept digital media signals approaching complexity, the human can brain may deal with. Moreover, due to the high quality of DNN models, nowadays digital media systems and application significantly improved their performance, comparing to the research status at the turn of the century when the existing media standards (JPEG-x, MPEG-x) were being established.

Discriminant analysis is a procedural concept related to finding, in general hidden features of the analyzed class of objects which make them different from features of other classes of objects being considered. To this goal, usually a separation measure is defined which is used to find a data model supporting the extraction of such discriminant features.

The classical linear discriminant analysis (LDA) plays the prominent role in pattern recognition systems. Essentially it is based on the Fisher's class separation loss function defined as the ratio of within-class variance to between-class variance (e.g. [3], [18]). The linear transformation which minimizes the Fisher's discriminant measure is used as the feature extractor which is only a part of end-to-end classifier. Interestingly, the LDA features can be used to recognize objects from classes having no representatives in the training stage of LDA model, e.g. for recognition of persons from a new facial database.

In the case image classifiers defined by DNN models feature extraction is performed by its CNN (convolutional neural network), which significantly improved state-of-the in computer vision problems like image classification [11, 6, 16], image segmentation [1, 5, 2], and face modeling [22, 7].

In DNN modeling about separation of feature classes mainly decides the loss function. The most popular objective function is the one which combines the SoftMax function with Shannon information measure. While the former function converts real valued scores into a distribution of probabilities (PDF), the latter one is obtained by using the Kullback-Leibler (KL) divergence measure[1] to this pdf and the crisp membership value for the class the training element belongs to. However, the cross entropy measure is focused on the discrimination of probability distributions which are computed by DNN on its output – not on the minimization of intra-class variances relatively to inter-class variances for deep features (also known as embeddings) which are computed by DNN at the end of feature extraction pipeline of network layers[2].

Embeddings have received immense interest in the research community. The methodology of embeddings has shown a superiority and become an important tool for representation of data for many generic tasks, such as speaker verification, speaker diarization in the audio domain and also for face verification, classification, similarity measurements in the image domain.

[1] It is known to be equal to cross entropy in this special case.

[2] Usually it is the output of the nonlinear activation following the last convolutional layer in the processing pipeline, preceding full connection layer(s).

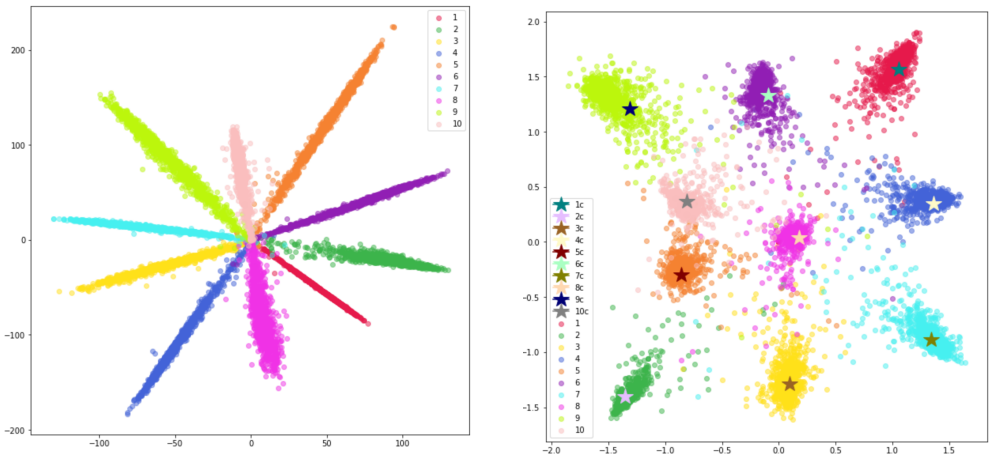

Figure 1. Deep 2D features (embeddings) for MNIST training data and the loss function: (left) SoftMax followed by the Shanon information function, (right) augmented by intra-class variance term computed within mini-batch.

Embeddings modeled as deep neural network and trained on huge amount of data has shown ability to encode different properties of target signal. Enhancements of data embeddings can improve performance the current and potential application of this tool. Intra-class variances extending probability divergence measures support development of deep features since then we can approximate the class dependent weighted Euclidean norm by the class independent regular Euclidean norm. It is experimental observation which can be explained by a symmetrical clustering of deep features as the point cloud representing the embedded vectors of the same class (cf. Figure 1).

In the classical optimization to determine variances for the combined loss functions, the complete set of training vectors is used for each optimization step. In deep learning practice, the data sets are huge and therefore stochastic optimization is a reasonable tool and then computing of class separation measure is replaced by its estimation modified either for each data training sample or for a mini-batch of training data samples.

In the existing solutions [19] used for face recognition the exponential weighting is performed outside of gradient back-propagation while the centroid learning requires the modification of standard gradient flow what imposes the extra complexity for the training algorithm. In the paper we show how to re-define the loss functions for both techniques in order to use standard stochastic gradient descent (SGD) optimizers for model training without degrading of its performance.

The research is presented in three sections. In Section 2 the mathematical concepts related to the discriminant analysis are defined and the basic properties summarized including loss function definitions. In Section 3 the architectures are presented in symbolic notation for tensor neural networks. In Section 4 the experimental results

are reported in relevant tables and figures. The paper is closed by the conclusion, the acknowledgment, and the references.

2. Discriminant analysis – concepts and basic properties

In this section mathematical concepts related to the discriminant analysis are summarized: Kullback-Leibler divergence of probability distribution functions and the data scatter analysis, and the loss functions compared in the research.

2.1. Information measure for target class at SoftMax random data source – the reference loss function

Let x be the input tensor of CNN neural network designed for solving of K class problem. The output $p_k \doteq P(x)[k]$ is the probability of x to be in class $k = 1, \ldots, K$. We assume that probability distribution $p \doteq P(x) \in \mathbb{R}^K$ is computed from the scores $s \doteq S(x) \in \mathbb{R}^K$ which in turn are processed from the feature vector $f \doteq F(x) \in \mathbb{R}^N$, i.e. from the embedding of x into N dimensional space [3]:

$$x \mapsto F(x) \mapsto s = S(x) \mapsto p = P(x) \longrightarrow p_k \doteq \frac{e^{s_k}}{\sum_{i=1}^{K} e^{s_i}}, \quad k = 1, \ldots, K \tag{1}$$

If the training input x belongs to the class k then the value of the *reference loss function* $\mathcal{L}_0(x)$ equals to the Shannon information measure[4] $(-\log P(x)[k]) = -\log p_k$ of class k with respect to SoftMax random data source:

$$\text{input } x \text{ belongs to class } k \longrightarrow \mathcal{L}_0(x) \doteq -\log P(x)[k] = -\log p_k \tag{2}$$

The above definition refers to the single training example x. The Shanon information for the mini-batch is the average information for all mini-batch elements:

$$\mathcal{L}_0\left(x_{i:i+N_b}\right) = -\frac{1}{N_b} \sum_{j \in [i, i+N_b)} \log(p_{y_j}) \tag{3}$$

It is really interesting that the definition of $\mathcal{L}_0$ via Shannon information measure is not used in the literature. Instead we have two other popular definitions: Kullback-Leibler divergence[5] $D_{KL}(\mathbf{1}_k||p)$ and cross entropy $H(\mathbf{1}_k, p)$. [6] It easy to see

[3] e^{s_k} is $exp(s_k)$

[4] Information measure of symbol k according Shannon theory of information.

[5] By $\mathbf{1}_k$ we denote the probability distribution q such that $q_k = 1$.

[6] p_k is computed probability distribution for k-th class, q_k - target probability distribution

the equivalence of those definitions:

$$
\begin{aligned}
&D_{KL}(q||p) \doteq \sum_{k=1}^{K} q_k \log \frac{q_k}{p_k} \longrightarrow D_{KL}(\mathbf{1}_k||p) = -\log p_k = \mathcal{L}_0(x) \\
&H(q,p) \doteq \sum_{k=1}^{K} q_k \log \frac{1}{p_k} = H(q) + D_{KL}(q||p) \longrightarrow \underbrace{H(\mathbf{1}_k,p))}_{0} + \underbrace{D_{KL}(\mathbf{1}_k||p)}_{-\log p_k} = \mathcal{L}_0(x)
\end{aligned}
\tag{4}
$$

2.2. Variability measures

There are two concepts related to data variability in sets of data samples: variance and scatter. It seems that average squared distance between all pairs of elements in the given set, also known as the *set scatter*, is more appealing to daily intuition than the average squared distance to the set's center. Intuitively, whenever we transform data samples before they are assigned to classes, we would like to make their scatter between classes high and within class low. In this subsection we show that both concepts are mathematically equivalent when computed for the same set of data samples. Moreover the scatter of the sets union embraces their within class and between class variability. We use $\doteq$ as *is equal by definition* term.

Variance and scatter of vectorial data set

The variance of a set (sample) X having discrete probability distribution $p(x)$ normalized to "one", $\sum_{x\in x} p(x) = 1$	$\mathrm{var}(X) \doteq \sum_{x\in X} p(x)\|x - \bar{x}\|^2.$
The scatter of a set (sample) X is the mean of squared Euclidean distance between any pair of sample elements[7].	$\mathrm{SCATTER}(X) \doteq \frac{1}{2}\sum_{x\in X}\sum_{y\in X} p(x)p(y)\|x-y\|^2$
Property: Concept of scatter is equivalent to concept of variance.	$\mathrm{SCATTER}(X) \equiv \mathrm{var}(X)$

The variability properties, we discuss here, do not dependent on the name we give to values $p(x)$. The name could be probability or weight – their positivity and summation to one is important. In context of loss functions to be used for DNN modeling, the weights are interpreted rather as forgetting factors than probabilities.

Class mean and grand mean

Let the sample X consists of disjoint subsets X_k including samples drawn from classes[8] $k, k \in [K]$. Let $p_X(x)$ be the probability of x within X while $p_{X_k}(x)$ is the

[8] $[K]$ denotes any set of K indexes, e.g. zero-based indexing $\{0,\ldots,K-1\}$ or one-based indexing $\{1,\ldots,n\}$.

probability of x within X_k. Then

class mean for sample X_k*:*	$\bar{x}^k \doteq \sum_{x \in X_k} p_{X_k}(x)x$
global (grand) mean from sample X*:*	$\bar{x} \doteq \sum_{x \in X} p_X(x)x$
Property: The grand mean is the average of class means.	$P_k \doteq \sum_{x \in X_k} p_X(x) \longrightarrow \bar{x} = \sum_{k \in [K]} P_k \bar{x}^k$

Within class variance and covariance

Within class (intra-class) variance is important to control separation via attracting elements of the same class. The intra-class covariance matrix can be used to evaluate covariances of within-class embedding errors and then visualize them via initial PCA components, computed after each training epoch.

Variance of sample X_k :	$\mathrm{var}(X_k) \doteq \sum_{x \in X_k} p_{X_k}(x)\|x - \bar{x}^k\|^2$
Scatter of sample X_k:	$\mathrm{SCATTER}(X_k) \doteq \frac{1}{2} \sum_{x,y \in X_k} p_{X_k}(x)p_{X_k}(y)\|x - y\|^2$
Within-class variance is the average of variances for class samples $X_1, \ldots, X_K$:	$\mathrm{var}_w(X) \doteq \sum_{k=1}^{K} P_k \mathrm{var}(X_k)$
Covariance matrix $R(X_k)$ for sample X_k of class k	$R(X_k) \doteq \sum_{x \in X_k} p_{X_k}(x) \left(x - \bar{x}^k\right) \left(x - \bar{x}^k\right)^\intercal$
Within-class covariance matrix is the average of matrix covariances in classes:	$R_w(X) \doteq \sum_{k \in [K]} P_k R(X_k)$

Within class variance and covariance – properties

The properties show how the within-class variance and covariance matrix are related and suggest how covariances of within-class error can be reduced by DNN learning of embedding function without compromising the minimization of within-class variance.

Property: Within-class variance equals to the trace of within class covariance matrix:	$\mathrm{var}_w(X) = \mathtt{tr}\left[R_w(X)\right]$
Property: The scatter of class sample X_k equals to the variance of class sample X_k:	$\mathrm{SCATTER}(X_k) = \mathrm{var}(X_k),\ k \in [K]$
Property: The within-class scatter, i.e. the average of scatters for class samples equals to the within-class variance:	$\sum_{k\in[K]} P_k \cdot \mathrm{SCATTER}(X_k) = \mathrm{var}_w(X)$
Property: The within-class variance is variance for class centered samples:	$\mathrm{var}_w(X) = \sum_{x\in X} p_X(x)\|x^{(c)}\|^2$, where $x^{(c)} \doteq x - \bar{x}^k$, if $x \in X_k$.
Property: The within-class covariance matrix equals to the covariance matrix for class centered samples:	$R_w(X) = \sum_{x\in X} p_X(x) x^{(c)} (x^{(c)})^\intercal$, where $x^{(c)} \doteq x - \bar{x}^k$, if $x \in X_k$.

2.3. Loss function extension by intra-class variance

Before adding any new term to the reference loss function $\mathcal{L}_0$ we should describe the stochastic optimization context. The contemporary DNN optimizers are based on stochastic gradient descent/ascent algorithm (SGD/SGA). The optimizers themselves are standalone procedures working in the mini-batch mode, i.e. expecting the gradient $\nabla_W \mathcal{L}\left(x_{i:i+N_b}, y_{i:i+N_b}\right)$ for the loss function $\mathcal{L}$ which depends on N_b inputs x_j, the N_b outputs y_j drawn from the training sequence, $j \in [i, \ldots, I + N_b)$, $i = 0, N_b, 2N_b, \ldots$, and optionally the inner layer(s) outputs(s). The gradient itself is computed by DNN engine using a type of the AutoGrad algorithm. Its responsibility of the programmer to deliver a differentiable loss function $\mathcal{L}$ which directly or indirectly depends on network's input data, on the current model (represented by the parameters W), and on the desired (by the teacher) network's outputs.

Usually, the programmer computes the *instant* loss function for all inputs in the current mini-batch and then makes an aggregation of the results, e.g. averaging, in order to obtain the *mini-batch* loss function value. However, there are engines which require the vector of loss function values, the kind of aggregation (reducing) method, and then back-propagate the error into DNN instances, created for all mini-batch elements.

In order to handle learning of centroids $C = [C_1, \ldots, C_k] \in \mathbb{R}^{N\times K}$ we join an additional Hadamard $\mathbb{H}$ layer[9] with the constant input $\mathbf{1}_{N\times K}$, parameters matrix C,

[9]The Hadamard layer multiplies the input tensor X by the weight tensor W : $H = X * W$.

and the following functionality for the loss term $\mathcal{L}_{var}$:

$$H = \mathbb{H}(\mathbf{1}_{N\times K}, C) \doteq C, y_{i:i+N_b} \in [K]^{N_b} - \text{ class indexes for the mini-batch} \longrightarrow$$
$$\mathcal{L}_{var}(x_{i:i+N_b}) \doteq \frac{1}{N_b} \sum_{j\in[i,i+N_b)} \|x_j - C_{y_j}\|^2 \tag{5}$$

The whole loss function $\mathcal{L}_1 \doteq \mathcal{L}_0 + \mathcal{L}_{var}$. The SGD theory (for instance AdaM [8]) the iterations over the whole batch, i.e. the whole training sequence, lead to the local minimum of $\mathcal{L}_1$ for certain parameters W^* in the original part of DNN and for some C^* in the extra part. Since $\mathcal{L}_{var}$ is nonnegative, additive term in $\mathcal{L}_1$, the local minimum of the whole function implies the local minimum for each. However, the whole batch, i.e. whole training sequence can be divided into K subsequences J_k with elements from the same class. Then over the subsequences $k \in [K]$ we have also a minimum for the variance of class k:

$$C_k^* = \arg\min_{C_k} \left[\sum_{j\in J_k} \|x_j - C_k\|^2 \right] \tag{6}$$

Since x_j depends on the current model W_j, the embedded data (deep features) are changing in time, the above reasoning is just intuition – not really a proof of convergence.

We get more insight if we notice that minimization of the function $f(C_k) \doteq \sum_{j\in J_k} \|x_j - C_k\|^2$ always leads to the centroid C_k^* and the minimum value equals to $|J_k| \cdot \mathrm{var}(\{x_j : j \in J_k\})$. Moreover this minimum is global. Since SGD (AdaM [8]) is proved to be convergent with probability one to a local minimum of $\mathcal{L}_1$, on its part $\mathcal{L}_{var}$, it is also convergent with probability one to all class centroids giving as the result variances on all class subsequences $J_k, k \in [K]$.

Apparently the idea to join classifier loss function with intra-class variance was proposed by Wen et al. [19]. Their approach to update class centroids was defined outside the error back-propagation framework. We present the update in another form which gives us conclusion that the update is the exponential, biased weighting of means for embedded vectors x_j from the same class $k \in [1, K]$, occurring in the same mini-batch i:

$$J'_k \doteq \{j \in [i, i+N_b) : y_j = k\},\ n_k \doteq |J'_k|,\ n_k \neq 0 \longrightarrow$$
$$C_k := (1-\alpha)\cdot C_k + \frac{\alpha}{n_k} \cdot \sum_{j\in J'_k} x_j \tag{7}$$

There is also another possibility to make weighting of embedded vectors and centroids one by one without averaging them in mini-batches (n_k term) ([9]):

$$j \in [i, i+N_b),\ y_j = k \longrightarrow C_k := (1-\alpha)\cdot C_k + \alpha \cdot x_j \tag{8}$$

This is also biased, exponential weighting of embedded vectors. However the embeddings defined by function F are modified after each batch. Moreover, having the

centroids iterated outside of the gradient framework we cannot use for them the conclusions on stochastic convergence to local minimum by SGD optimization method (e.g. AdaM algorithm [8]). Our updates for centroids are integrated with SGD framework with learning factors modified according embedding data evolution what is the main reason of provable stochastic convergence.

Though in this paper we do not present the loss function option with the subtracted term for inter-class variance, the concept of nonlinear discriminant analysis in principle is possible to produce yet another embeddings for various applications. In such extension the SGD optimizer is also doing the main work for updates not only for class centroids but also it updates the grand mean centroid while keeping bounded the values of the total variance for all embedded vectors.

3. CNN architectures in symbolic notation

3.1. Elements of symbolic notation

The symbolic notation for DNN architectures follows the definitions of Symbolic Tensor Neural Networks presented for the first time in the tutorial [17] of the second author. The selected rules of this notation are given in the table below:

Symbol	Description
$\boxed{c} \mapsto \mathcal{I}_{a}^{b}$	Input layer of name (a) with tensor of signal axes (b) and feature axis (c), e.g. $\boxed{image} \mapsto \mathcal{I}_{yx}^{1}$
$\mathbb{C}_{c\,a}^{d\,b}$	Convolutional layer with (b) kernels of dimensions and strides (if any) (a) having options (c) followed by activation(s) (d), e.g. ${}_{p}\mathbb{C}_{5}^{br\,64}$ where there are 64 convolutions with kernels of size 5 at each signal axis, zero padding option is required, and the batch normalization and ReLU activation is performed, afterwards.
${}_{b}\mathbb{P}_{a}$	Pooling layer which aggregates feature values in blocks of size (a) with optional striding (a) according to a technique (b), e.g. ${}_{m}\mathbb{P}_{2}$ – max pooling in blocks of size 2 at each signal axis is performed.
$\mathbb{F}_{b}^{a}$	Full connection layer with (a) output features followed by activation(s) (b), e.g. $\mathbb{F}_{br}^{4096}$ where feature vector with 4096 components is computed by linear operation on layer's tensor input. b in this example is batch-normalization, r - ReLU activation function
$\mathbb{D}_{a}$	The layer drop-outs randomly (a) percents of weights in the next layer. It is only used at training, e.g.$\mathbb{D}_{50}$ makes zeroing for 50% of weights.

3.2. Face descriptor – CNN architecture with centroid layer

Descriptor extraction for face recognition fits to the first above category. The solution proposed by Jacek Naruniec and Marek Kowalski of Warsaw University of Technology (KN-FR [9]) is a combination of convolutional neural network with loss function aggregating SoftMax loss with intra-class variance, proposed by Wen, Zhang et al. in [19]: *A Discriminative Feature Learning Approach for Deep Face Recognition.*

In the architecture described below we added the extra centroid layer which is referenced from the loss function – the term for intra-class variance. Therefore the normalized embeddings into the Euclidean space $\mathbb{R}^{1024}$ are different then the embeddings in [19] and [9].

$$
\begin{array}{l}
\boxed{image} \mapsto \mathcal{I}^{1}_{yx}\ {}_{p}\mathbb{C}^{64}_{br5}\ {}_{p}\mathbb{C}^{64}_{br5}\ {}_{p}\mathbb{C}^{64}_{br5}\ {}_{m}\mathbb{P}_{2}\ {}_{p}\mathbb{C}^{128}_{br5}\ {}_{p}\mathbb{C}^{128}_{br5}\ {}_{p}\mathbb{C}^{128}_{br5}\ {}_{m}\mathbb{P}_{2} \\
{}_{p}\mathbb{C}^{256}_{br5}\ {}_{p}\mathbb{C}^{256}_{br5}\ {}_{p}\mathbb{C}^{256}_{br5}\ {}_{p}\mathbb{C}^{256}_{br5}\ {}_{m}\mathbb{P}_{2}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{m}\mathbb{P}_{2} \\
{}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{p}\mathbb{C}^{512}_{br5}\ {}_{m}\mathbb{P}_{2}\ \mathbb{F}^{4096}_{br}\ \mathbb{F}^{1024} \mapsto \boxed{x} \\
\boxed{x/\|x\|} \mapsto \boxed{norm}\ \mathbb{F}^{P} \mapsto \boxed{scores} \qquad \boxed{ones} \mapsto \boxed{\mathbf{1}\cdot C} \mapsto \boxed{centers} \\
\hline
\overset{knFaRec}{\mathcal{NET}} \left(\begin{array}{lcl} image & := & 112_{yx}; \\ optima & := & [loss, MomentumSGD, \\ & & ShannonInfo + IntraClassVar] \end{array} \right)
\end{array}
$$

Note that embedded vectors are normalized here in order to get the unit length.

3.3. MNIST descriptors – CNN architecture with centroid layer

In order to illustrate intuitively the concept of embeddings controlled by learned centroids we consider the generic MNIST problem for handwritten digits recognition and Fashion MNIST recognition for fashion images. For both problems the following CNN architecture is trained.

$$
\begin{array}{l}
\boxed{image} \mapsto \mathcal{I}_{yx}\ \mathbb{C}^{16}_{r3}\ \mathbb{C}^{32}_{r3}\ {}_{m}\mathbb{P}_{2}\ \mathbb{D}_{50}\ \mathbb{C}^{64}_{r3}\ \mathbb{C}^{64}_{r3}\ {}_{m}\mathbb{P}_{2}\ \mathbb{D}_{50}\ \mathbb{F}^{n}_{r} \mapsto \boxed{x} \\
\boxed{x} \mapsto \mathbb{F}^{10} \mapsto \boxed{scores} \qquad \boxed{ones} \mapsto \boxed{\mathbf{1}\cdot C} \mapsto \boxed{centers} \\
\hline
\overset{embed-n}{\mathcal{NET}} \left(\begin{array}{l} image := 28_{yx}; \\ optima := [loss, AdaM, ShannonInfo + IntraClassVar] \end{array} \right)
\end{array}
$$

The feature (embedded) vector $x \in \mathbb{R}^n$ in this study is normalized or not-normalized. In our experiments, the embedding space dimensionality is low and equals to $n = 2, 4$.

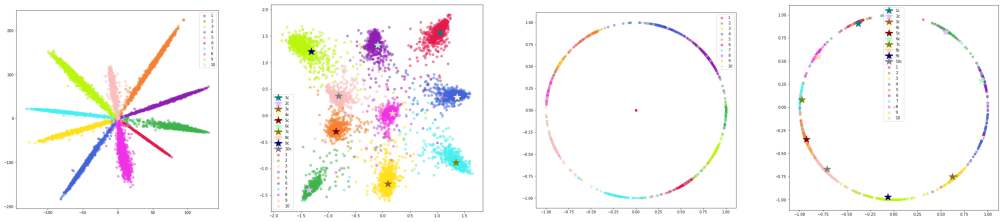

Figure 2. MNIST 2-dim embeddings visualization.

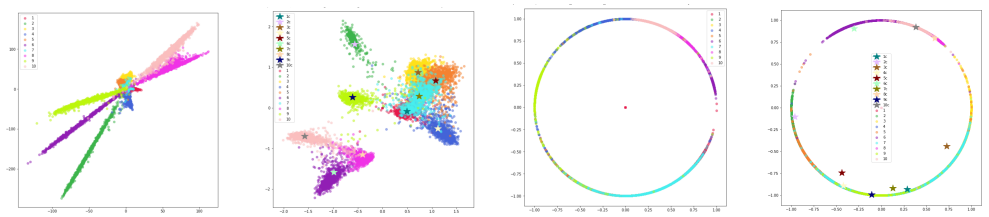

Figure 3. Fashion MNIST 2-dim embeddings visualization .

4. Experiments

Our experiments are carried out on convolutional neural network, changing the size of the layers used to extract embeddings. MNIST [10] and Fashion-MNIST [21] datasets are used. The comparative analysis is aimed at indicating potential factors in which the loss function should be modified in order to achieve optimal results for embeddings similarity.

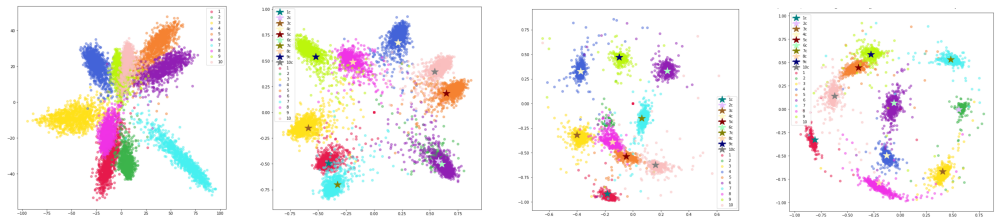

Figure 4. MNIST 4-dim embeddings visualization using 2-component PCA.

Most of prior works focus on large-scale problems, where training hyperparameters can have as significant impact as loss function selection, architecture or weights initialization. Therefore in this paper we focus on small-scale problem followed by many experiments with the same initial parameters to see where large-scale problems like face recognition can have possible bottlenecks and how to improve it.

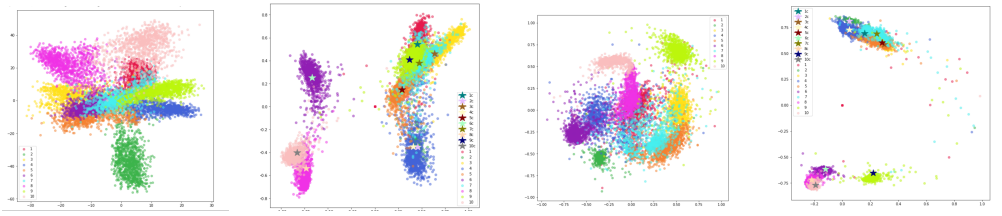

Figure 5. Fashion MNIST 4-dim embeddings visualization using 2-component PCA.

4.1. Implementation details

For training we use AdaM optimizer with learning rate 0.001, mini-batch size of 256. Each network is trained over 20 epochs without any additional learning rate drop. We change the size of bottleneck embedding layer and also optionally apply normalization of features in experiments. For visualization purposes of more than 2 dimensional space we use 2-component PCA decomposition.

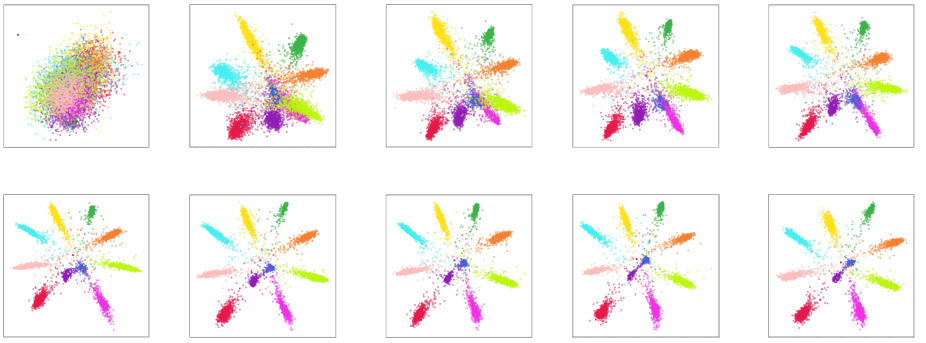

Figure 6. Step-by-step visualization of centroids for 2 dimensional embeddings on MNIST dataset for intra-class variance ($\lambda = 0.05$) + cross-entropy loss function (epochs 1, 2, 3, 4, 5, 8, 11, 14, 17, 20).

4.2. Performance metrics

The *recognition accuracy* takes the simple average success rate as the final score, counted by

$$accuracy = \frac{number\ of\ correct\ decisions}{total\ number\ of\ decisions}$$

We use this metric both for both, the nearest-centroid evaluation and SoftMax probability evaluation. Obviously, at testing, the decision based on the maximum score

can replace the evaluation of the SoftMax probability distribution and its maximum. Namely, we get the predicted class on test datasets by t(a) the maximum score, (b) the nearest centroid in the embedding space. The centroids C_k are learned from training data using the proposed Hadamard $\mathbb{H}$ layer.

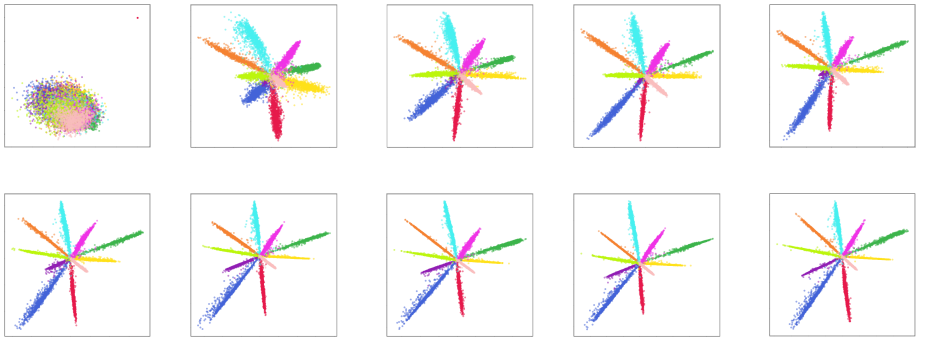

Figure 7. Step-by-step visualization of centroids for 2 dimensional embeddings on MNIST dataset for cross-entropy loss function (epochs 1, 2, 3, 4, 5, 8, 11, 14, 17, 20).

4.3. Results

We prepare visualization for two and four dimensional embeddings[10] to show an impact of intra-class variance loss on centroids for two-dimensional embeddings on MNIST dataset (Figure 2), Fashion-MNIST (Figure 3) and also for 4 dimensional space embeddings on both datasets (Figure 4, 5).

We can visually notice the impact of intra-class variance loss on overall distribution of embeddings (Figure 7). *Long-tailed* embeddings obtained from entropy loss are packed into small clusters (Figure 6).

The metrics evaluation is carried out on test datasets both for nearest centroid (Table 1) and SoftMax probability (Table 2) metrics. We observe that SoftMax probability gives similar performance on the same architecture both for the normalized and the non-normalized cases.

Results (Table 1) show that using non-normalized features intra-class variance loss outperforms cross entropy loss for 2-dimensional and 4-dimensional embedding vectors, which we indicate using **bold** font to indicate best result for each vector size. Using smaller λ allows to achieve better results in nearest centroid metric, so it indicates that main loss component should be cross-entropy. In addition, the introduction of an intra-variance loss function does not worsen the results of maximum score metric (Table 2),

[10]For each row of presented embedding visualisation (Figure 2, 4, 3, 5) we have the following order: non-normalized SoftMax embeddings, non-normalized intra-class variance loss embeddings, normalized SoftMax embeddings and normalized intra-class variance embeddings.

Table 1. Results for minimum distance to the centroid (class mean).

	Nearest Centroid				
	Non-Norm.		**Norm.**		
Emb. Dim.	2x1	4x1	2x1	4x1	
Cross entropy loss					
MNIST	0.909	0.975	0.984	0.988	
Fashion-MNIST	0.746	0.869	0.814	0.894	
Intra-class variance					λ
MNIST	0.973	0.992	0.980	0.988	1.0
Fashion-MNIST	**0.880**	0.891	0.715	0.884	
MNIST	0.971	0.991	**0.985**	0.988	0.5
Fashion-MNIST	0.870	0.902	0.738	0.889	
MNIST	0.972	**0.992**	0.984	0.990	0.05
Fashion-MNIST	0.867	**0.910**	0.820	0.891	

Table 2. Results for maximum score (softmax).

	Highest Score				
	Non-Norm.		**Norm.**		
Emb. Dim.	2x1	4x1	2x1	4x1	
Shannon information					
MNIST	0.986	0.994	0.986	0.994	
Fashion-MNIST	0.895	0.924	0.894	0.920	
intra-class variance					λ
MNIST	0.985	0.994	0.981	0.995	1.0
Fashion-MNIST	0.882	0.904	0.884	0.915	
MNIST	0.986	0.995	0.982	0.995	0.5
Fashion-MNIST	0.890	0.918	0.889	0.913	
MNIST	0.986	0.995	0.985	0.994	0.05
Fashion-MNIST	0.895	0.927	0.891	0.926	

5. Discussion

We observ that for embeddings the normalization in Euclidean norm, i.e. the projection on to the unit sphere, is not effective for intra-class variance objective. The Shannon information based embeddings exhibit better results only after normalization, however for non-normalized embeddings learned by intra-class variance the success rate is higher .

Presented loss function and Hadamard layer should be tested on more difficult multi-class problems and larger training sets. Searching hyperparameter space for complex computational algorithms is expensive, so our example can be used as a starting point. This is undoubtedly the disadvantage of each algorithm - selection: how to choose size of the embedding vector and $\lambda_\mathbf{i}$ coefficients for each component of the loss function.

Conclusion

We conducted many experiments using convolutional neural network and proposed contribution to intra-class variance minimization using standard optimization techniques through Hadamard $\mathbb{H}$ layer. Embeddings extracted using CNN and trained using intra-class variance are less sparse than embeddings extracted directly from classifier trained with entropy objective function.
Intra-class variance enables to get better results for nearest-centroid evaluation for both test dataset and embedding dimensions (Table 1):

1. 2-dim: **0.880**, **0.985**
2. 4-dim: **0.992**, **0.910**

Intra-variance loss also does not worsen the classification results with the use of softmax (2.

This work shows potential usability of intra-class variance loss (or regularization term) for classifiers that can be trained in supervised fashion without any additional arbitrary strategy for selection of training examples or on-the-fly statistics calculations, as you can find for works referring to triplet-loss [15] or contrastive-loss [4].

Acknowledgment

The presented research was conducted by both authors within the Statutory Works of Faculty of Electronics and Computer Technology, Warsaw University of Technology, Poland.

References

[1] Badrinarayanan V., Kendall A., and Cipolla R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *arXiv:1511.00561 [cs]*, Nov. 2015. arXiv: 1511.00561.

[2] Chen L.-C., Papandreou G., Kokkinos I., Murphy K., and Yuille A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *arXiv:1606.00915 [cs]*, June 2016. arXiv: 1606.00915.

[3] Fisher R. A. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics*, 7(2):179–188, Sept. 1936.

[4] Hadsell R., Chopra S., and LeCun Y. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1735–1742, Washington, DC, USA, 2006. IEEE Computer Society.

[5] He K., Gkioxari G., Dollár P., and Girshick R. Mask R-CNN. *arXiv:1703.06870 [cs]*, Mar. 2017. arXiv: 1703.06870.

[6] He K., Zhang X., Ren S., and Sun J. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[7] Jourabloo A. and Liu X. Large-Pose Face Alignment via CNN-Based Dense 3d Model Fitting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4188–4196, Las Vegas, NV, USA, June 2016. IEEE.

[8] Kingma D. P. and Ba J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Kowalski M. and Naruniec J. (Warsaw University of Technology – personal communication).

[10] Lecun Y. and Cortes C. The MNIST database of handwritten digits.

[11] Ren S., He K., Girshick R., and Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, June 2017.

[12] Rosenblatt F. The perceptron–a perceiving and recognizing automaton. report 85-460-1. Technical report, Cornell Aeronautical Laboratory, 1957.

[13] Rumelhart D. E., Hinton G. E., and Williams R. J. Learning representations by back-propagating errors. In Anderson J. A. and Rosenfeld E., editors, *Neurocomputing: Foundations of Research*, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[14] Schmidhuber J. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[15] Schroff F., Kalenichenko D., and Philbin J. FaceNet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, Boston, MA, USA, June 2015. IEEE.

[16] Simonyan K. and Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Sept. 2014. arXiv: 1409.1556.

[17] Skarbek W. Symbolic tensor neural networks for digital media - from tensor processing via bnf graph rules to creams applications. *Preprint stored in Cornell University Archive*, abs/1809.06582, 2018.

[18] Skarbek W., Kucharski K., and Bober M. Dual lda for face recognition. *Fundam. Inform.*, 61:303–334, 07 2004.

[19] Wen Y., Zhang K., Li Z., and Qiao Y. A Discriminative Feature Learning Approach for Deep Face Recognition. In Leibe B., Matas J., Sebe N., and Welling M., editors, *Computer Vision – ECCV 2016*, volume 9911, pages 499–515. Springer International Publishing, Cham, 2016.

[20] Werbos P. J. Applications of advances in nonlinear sensitivity analysis. In Drenick R. F. and Kozin F., editors, *System Modeling and Optimization*, pages 762–770, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[21] Xiao H., Rasul K., and Vollgraf R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, Aug. 2017. arXiv: 1708.07747.

[22] Zhang K., Zhang Z., Li Z., and Qiao Y. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct. 2016.

This paper is a revised and extended version of work originally presented at the 16th International Symposium New Trends in Audio and Video - NTAV2018, 11-13 October 2018, Poznań, Poland

Received 29.01.2019, Accepted 23.04.2019