

Tennessee State University

Digital Scholarship @ Tennessee State University

Computer Science Faculty Research

Department of Computer Science

11-16-2021

Robust feature space separation for deep convolutional neural network training

Ali Sekmen

Tennessee State University

Mustafa Parlaktuna

Tennessee State University

Ayad Abdul-Malek

Tennessee State University

Erdem Erdemir

Tennessee State University

Ahmet Bugra Koku

Middle East Technical University

Follow this and additional works at: <https://digitalscholarship.tnstate.edu/computerscience>



Part of the [Computer Engineering Commons](#)

Recommended Citation

Sekmen, A., Parlaktuna, M., Abdul-Malek, A. et al. Robust feature space separation for deep convolutional neural network training. *Discov Artif Intell* 1, 12 (2021). <https://doi.org/10.1007/s44163-021-00013-1>

This Article is brought to you for free and open access by the Department of Computer Science at Digital Scholarship @ Tennessee State University. It has been accepted for inclusion in Computer Science Faculty Research by an authorized administrator of Digital Scholarship @ Tennessee State University. For more information, please contact XGE@Tnstate.edu.

Robust feature space separation for deep convolutional neural network training

Ali Sekmen¹ · Mustafa Parlaktuna¹ · Ayad Abdul-Malek¹ · Erdem Erdemir¹ · Ahmet Bugra Koku²

Received: 30 September 2021 / Accepted: 1 November 2021

Published online: 16 November 2021

© The Author(s) 2021 [OPEN](#)

Abstract

This paper introduces two deep convolutional neural network training techniques that lead to more robust feature subspace separation in comparison to traditional training. Assume that dataset has M labels. The first method creates M deep convolutional neural networks called $\{\text{DCNN}_i\}_{i=1}^M$. Each of the networks DCNN_i is composed of a convolutional neural network (CNN_i) and a fully connected neural network (FCNN_i). In training, a set of projection matrices $\{\mathbf{P}_i\}_{i=1}^M$ are created and adaptively updated as representations for feature subspaces $\{\mathcal{S}_i\}_{i=1}^M$. A rejection value is computed for each training based on its projections on feature subspaces. Each FCNN_i acts as a binary classifier with a cost function whose main parameter is rejection values. A threshold value t_i is determined for i^{th} network DCNN_i . A testing strategy utilizing $\{t_i\}_{i=1}^M$ is also introduced. The second method creates a single DCNN and it computes a cost function whose parameters depend on subspace separations using the geodesic distance on the Grassmannian manifold of subspaces \mathcal{S}_i and the sum of all remaining subspaces $\{\mathcal{S}_j\}_{j=1, j \neq i}^M$. The proposed methods are tested using multiple network topologies. It is shown that while the first method works better for smaller networks, the second method performs better for complex architectures.

Keywords Deep Convolutional Neural Networks · Subspace Separation · Robust Deep Learning

1 Introduction

There is an explosion of deep learning applications since the reintroduction of Convolutional Neural Networks (CNNs) in image classification [1] in 2012 and successive years to ImageNet dataset [2–4]. It has been successfully applied in self-driving cars for traffic related objects and person detection and classification [5], face recognition for social media platforms [6], natural language processing [7], and symbolic mathematics [8]. A deep convolutional neural network architecture was developed in [9] for classification of skin lesions using large set of images for training. Another architecture was developed in [10] for detection of diabetic retinopathy in retinal fundus photographs. Predicting 3-D structure of a protein using only amino acid sequences is a challenging task. DeepMind recently announced that their algorithm called AlphaFold can predict protein structures with atomic accuracy using deep learning [11, 12].

This research is supported by DoD Grants W911NF-15-1-0495 and W911NF-20-1-0284.

✉ Erdem Erdemir, erdemir@tnstate.edu; Ali Sekmen, asekmen@tnstate.edu; Mustafa Parlaktuna, mparlaktuna@tnstate.edu; Ayad Abdul-Malek, aabdulmalek@tnstate.edu; Ahmet Bugra Koku, kbugra@metu.edu.tr | ¹Department of Computer Science, Tennessee State University, Nashville, TN 37209, USA. ²Department of Mechanical Engineering, Middle East Technical University, Ankara, Turkey.



There is a growing interest to explain the mathematical foundation of CNNs. The work in [13] uses wavelet theory to explain computational invariants in convolutional layers. It attempts to predict kernel parameters without the need for training. There are other attempts to establish mathematical foundation of deep convolutional neural networks [14, 15].

As *manifold assumption* hypotheses, in many real-world problems, high-dimensional data approximately lies on a lower dimensional manifolds [15]. It has been shown that the trajectories observed in F video frames of M independently moving rigid bodies come from a union of M 4-dimensional subspaces of \mathbb{R}^{2F} [16]. It is experimentally shown that face images of a person with the same facial expression under different illumination approximately lies in a 9-dimensional subspace [17]. A general framework for clustering of data that comes from a union of independent subspaces is given in [18, 19] and a practical algorithm is given in [20]. A detailed treatment of subspace segmentation problem can be found in [21]. Auto-encoder based deep learning is also applied to subspace clustering [22, 23].

The popularity of CNN stems from the fact that it acts as an automatic feature extractor with its cascaded layers that can generate increasingly complex features from input datasets. As opposed to unnatural aspects of some existing feature extractors, such as SIFT [24] for vision data and MFCC [25] for audio data, the final layer of a CNN typically generates feature vectors that is linearly separable by a Fully Connected Neural network (FCNN). A typical Deep Convolutional Neural Network (DCNN) is trained using Stochastic Gradient Descent (SGD) based algorithm such as Adam [26]. The work in [27] uses manifold learning to improve feature representations of a transfer learning model. The work in [28] uses local linear embedding on output of each convolutional layer for particularly recognizing actions. Some other related work are in [29, 30]

This research proposes two novel DCNN architectures and associated training methods with the main goal of converting input data into feature vectors on more separable manifolds (or subspaces) at CNN output. The first method creates multiple DCNNs, each of which adaptively generates a projection matrix for each feature subspace [31, 32]. A rejection value is computed for each label based on its projections on feature subspaces. The second method is based on the idea of maximizing the geodesic distance between a feature subspace and the sum of the remaining feature subspaces [33].

1.1 Paper contributions

- This work develops a classification method with using M deep convolutional neural networks in parallel. In training, a set of projection matrices is created and adaptively updated as representations for feature subspaces. A rejection value is computed for each training based on its projections on feature subspaces. A threshold value is determined for each network and a testing strategy utilizing all thresholds is also introduced.
- This work also develops another classification method using a single DCNN what minimizes a cost function whose parameters depend on subspace separations using the geodesic distance on the Grassmannian manifold of a feature subspace and the sum of all remaining feature subspaces.
- Experiments on real data (datasets on digits, alphabets, and fashion products) are performed to justify the proposed architectures. Five different deep convolutional network topologies are used to show that the proposed technique works better. The proposed methods are tested using five network topologies. It is shown that while the first method works better for smaller networks, the second method performs better for complex architectures.
- A new matrix rank estimation technique is introduced.

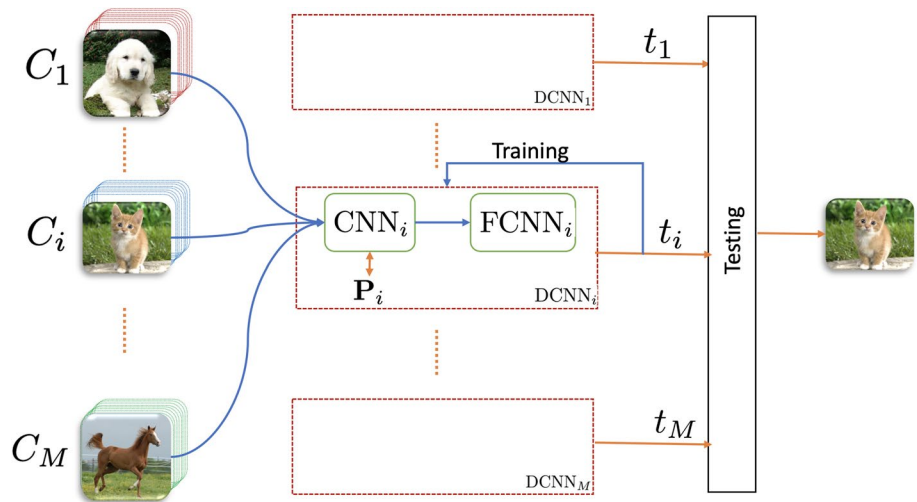
1.2 Layout

Section gives a detailed treatment of the first novel deep convolutional network with multiple CNNs. Section introduces the second novel network with maximum subspace separation based on principal angles between subspaces. The numerical experiments are presented in Section and some future work is motivated in Section .

2 Feature space separation—first approach

Figure 1 shows the network architecture. Let $\{C_i\}_{i=1}^M$ be the sets of M input classes. For each input class, a DCNN is constructed. Let $\{\text{DCNN}_i\}_{i=1}^M$ be the mentioned DCNNs. The kernel (filter) parameters of $\{\text{CNN}_i\}_{i=1}^M$ and weight and biases of $\{\text{FCNN}_i\}_{i=1}^M$ are randomly initialized. During training, a set of projection matrices $\{\mathbf{P}_i\}_{i=1}^M$ are created and iteratively

Fig. 1 System architecture for first approach



updated using feature subspaces $\{S_i\}_{i=1}^M$. Let x_j^n be n^{th} input in class C_j and let f_j^n be the corresponding feature vector at the CNN output. Let $d_{j,i}^n$ be the distance between f_j^n and S_i , which is computed at each iteration as in Equation (1).

$$d_{j,i}^n = \|(\mathbf{I} - \mathbf{P}_i)\mathbf{f}_j^n\|_2. \tag{1}$$

The objective of training is to minimize $d_{j,i}^n$ if $j = i$ and maximize it otherwise. A set of threshold values $\{t_i\}_{i=1}^M$ are generated as a result of trained DCNNs and these thresholds are used for testing when an unknown input is provided. Depending on some test topologies (as used in Section), FCNN $_i$ may be omitted.

2.1 Network training

Algorithm 1 summarizes high-level overall training and generation of threshold values. All network parameters are randomly initialized and training is performed using the steps described in the subsequent subsections.

Algorithm 1: Training for $\{DCNN_i\}_{i=1}^M$.

Data: Input data sets $\{C_i\}_{i=1}^M$
Result: Filter parameters for $\{CNN_i\}_{i=1}^M$, weight and biases for $\{FCNN_i\}_{i=1}^M$, corresponding thresholds $\{t_i\}_{i=1}^M$

- 1 Initialize filter parameters, weights, and biases randomly
- 2 **for** $i: 1 \rightarrow M$ **do**
- 3 Train Network $DCNN_i$
- 4 Calculate threshold value t_i
- 5 **end**

2.2 Singular value decomposition for subspace approximation

A set of feature subspaces $\{S_i\}_{i=1}^M$ are created by forward feeding of $\{C_i\}_{i=1}^M$ into CNN_i . Since all filter parameters are randomly initialized, those subspaces are expected to be not very good to start with. In order to match a subspace for i th feature space, all input data in C_i is passed through CNN_i and a data matrix \mathbf{W}_i , whose columns are the feature vectors at the CNN output for C_i , is created. Singular Value Decomposition (SVD) of \mathbf{W}_i is taken and its rank r_i is estimated. In this work, a new rank estimation algorithm was developed as described in Algorithm 2. Some other rank estimation techniques can be found in [34, 35].

Algorithm 2: Effective rank estimation.

Data: $m \times n$ matrix \mathbf{W} and a threshold κ
Result: Effective-rank(\mathbf{W})

- 1 Compute SVD of $\mathbf{W} = U\Sigma V^T$. Find singular values of \mathbf{W} . They are the diagonal entries $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r \geq 0$ of Σ , where $r \leq \min(m, n)$.
- 2 **for** $i: 1 \rightarrow r$ **do**
- 3 | energy(i) = $\sum_{j=i}^r \sigma_j^2$
- 4 **end**
- 5 **for** $i: 1 \rightarrow r - 1$ **do**
- 6 | $\Delta_{\text{Energy}}(i) = \text{energy}(i) - \text{energy}(i + 1)$
- 7 | **if** $\Delta_{\text{Energy}}(i) < \kappa$ **then**
- 8 | | $\Delta_{\text{Energy}}(i) = 0$
- 9 **end**
- 10 Find the peaks of Δ_{Energy}
- 11 Effective-rank(\mathbf{W}) is the index of the last peak.

Let the effective rank of \mathbf{W}_i be r_i . Then,

$$\mathbf{W}_i = \mathbf{U}_i \Sigma_i \mathbf{V}_i^T \tag{2}$$

The subspace $\mathcal{S}_i = \text{span}\{\mathbf{U}_i[1 : r_i]\}$ and the projection matrix is given as

$$\mathbf{P}_i = \hat{\mathbf{U}}_i \hat{\mathbf{U}}_i^T. \tag{3}$$

where $\hat{\mathbf{U}}_i = \mathbf{U}_i[1 : r_i]$.

2.3 DCNN_i training

Algorithm 3 gives the details of training for each DCNN_i. Note that the rejection is defined as $\mathbf{r}_{j,i}^n = (\mathbf{I} - \mathbf{P}_i)\mathbf{f}_j^n$ and it is fed into FCNN_i.

Algorithm 3: DCNN_i training.

Data: Input data sets $\{C_i\}_{i=1}^M$
Data: I : # of iterations and K : subspace update rate
Result: Filter parameters for $\{\text{CNN}_i\}_{i=1}^M$, weight and biases for $\{\text{FCNN}_i\}_{i=1}^M$

for ($e: 1 \rightarrow I$) **do**

Compute \mathbf{P}_i as in Equation (3)

for $k: 1 \rightarrow K$ **do**

for Each input \mathbf{x}_j^n in $\{C_j\}_{j=1}^M$ **do**

Feedforward \mathbf{x}_j^n through CNN_i to obtain \mathbf{f}_j^n

Find rejection $\mathbf{r}_{j,i}^n = (\mathbf{I} - \mathbf{P}_i)\mathbf{f}_j^n$

Find distance $d_{j,i}^n$

Feedforward $\mathbf{r}_{j,i}^n$ into FCNN_i

Update all network parameters using the following as penalty in backpropagation

if $j = i$ **then**

| $d_{j,i}^n$

else

| $1/d_{j,i}^n$

end

end

end

Compute class separation values

if full class separation **then**

| break

end

2.4 Computing class separation

Assume all input set \mathcal{C}_j is feedforwarded via CNN_i . Let $\mathbf{r}_{j,i}$ be a vector whose entries are the norms of rejection values, i.e., distances $d_{j,i}^n$ for all n input vectors in \mathcal{C}_j . Let $\min(\mathbf{r}_{j,i})$, $\max(\mathbf{r}_{j,i})$, and $\text{mean}(\mathbf{r}_{j,i})$ be the minimum, maximum, and mean values of $\mathbf{r}_{j,i}$. Then, $\max(\mathbf{r}_{i,i})$ is compared with $\min(\min(\mathbf{r}_{j,i})_{j=1, j \neq i}^M)$ to assess if full class separation was achieved. Algorithm 4 summarizes the steps.

Algorithm 4: Computing separation for i^{th} class

Data: Input data sets $\{\mathcal{C}_i\}_{i=1}^M$
Result: Class separation values
Result: Full separation check
for $j: 1 \rightarrow M$ **do**
 Feed forward $\{\mathcal{C}_j\}_{j=1}^M$ through DCNN_i
 Compute $\mathbf{r}_{j,i}$
 Store $\min(\mathbf{r}_{j,i})$, $\max(\mathbf{r}_{j,i})$, and $\text{mean}(\mathbf{r}_{j,i})$
end
if $\min(\min(\mathbf{r}_{j,i})_{j=1, j \neq i}^M) \geq \max(\mathbf{r}_{i,i})$ **then**
 Full separation is achieved
else
 Full separation is not achieved
end

2.5 Training algorithm—faster

In order to speed up training process, it is possible to use only non-separated input for the next training iteration. The class separation values are computed every K iterations and the input sets is updated accordingly. The entire dataset of the corresponding class is still used for computing \mathbf{P}_i . Algorithm 5 presents the steps.

Algorithm 5: Faster training for i^{th} class.

Data: Input data sets $\{\mathcal{C}_i\}_{i=1}^M$
Data: The rejection vectors for all classes - $\{\mathbf{r}_{j,i}\}_{j=1}^M$
Result: Input set for next training
for $r_{i,i}^n$ in $\mathbf{r}_{i,i}$ **do**
 if $r_{i,i}^n \geq \min(\min(\mathbf{r}_{j,i})_{j=1, j \neq i}^M)$ **then**
 Add \mathcal{C}_i^n to the next input set
 end
for $j: 1 \rightarrow M, j \neq i$ **do**
 for $r_{j,i}^n$ in $\mathbf{r}_{j,i}$ **do**
 if $r_{j,i}^n \leq \max(\mathbf{r}_{i,i})$ **then**
 Add \mathcal{C}_j^n to next training input set
 end
 end
end

2.6 Computing thresholds

The set of threshold values $\{t_i\}_{i=1}^M$ are computed after completion of training. Even though there are multiple ways that can be considered, this work uses three approaches as listed below. Each threshold is used to determine if an unknown input belongs to a particular class. Let \mathbf{x} be an unknown input that is passed through DCNN_i . If the rejection value of \mathbf{x} is less than t_i , then \mathbf{x} belongs to i^{th} class.

1. $t_i = \max(\mathbf{r}_{i,i})$

2. $t_i = \min(\mathbf{r}_{j,i}); j : 1 \rightarrow M, j \neq i$
3. $t_i = \text{mean}(\max(\mathbf{r}_{i,i}) + [\min(\mathbf{r}_{j,i}); j : 1 \rightarrow M, j \neq i])$

2.7 Numerical results

In order to find the label of an input image x , it is feedforwarded via all DCNNs and associated rejection values are computed and they are compared with corresponding threshold values. 0 or 1 is used to show x 's connection to that class. If there is only one network connection, that x is labeled as that class. If there does not exist a connection, the rejection values are ranked and x is labeled as the class with the minimum rejection value. Algorithm 6 summarizes the steps to determine the class label of an unknown image.

Algorithm 6: Classification of test data \mathbf{x}

Data: Unclassified data \mathbf{x}
Result: Classification index k for \mathbf{x}

- 1 Create empty sets \mathbf{A}, \mathbf{B}
- 2 **for** $i: 1 \rightarrow M$ **do**
- 3 Feedforward \mathbf{x} via DCNN $_i$ to get rejection r_i
- 4 $\mathbf{A.add}(r_i \leq t_i)$
- 5 $\mathbf{B.add}(r_i)$
- 6 **end**
- 7 **if** $\text{sum}(\mathbf{A}) == 1$ **then**
- 8 $k = \text{index}(\mathbf{A} == 1)$
- 9 **if** $\text{sum}(\mathbf{A}) == 0$ **then**
- 10 $k = \text{arg min}(\mathbf{B})$
- 11 **if** $\text{sum}(\mathbf{A}) > 1$ **then**
- 12 $k = \text{arg min}(\mathbf{B}[\text{index}(\mathbf{A} == 1)])$

3 Feature space separation—second approach

A CNN, after training, transfers input classes $\{C_i\}_{i=1}^M$ into feature spaces, typically manifolds or subspaces $\{S_i\}_{i=1}^M$ at its output layer. The goal of our second approach is to maximize separation of each feature subspace S_i with the sum of the remaining feature subspaces $\{S_j\}_{j=1, j \neq i}^M$. It creates a single DCNN and computes a cost function whose parameters depend on subspace separations using the geodesic distance on the Grassmannian manifold of subspaces S_i and $\sum_{j=1, j \neq i}^M S_j$.

Fig. 2 System architecture for second approach

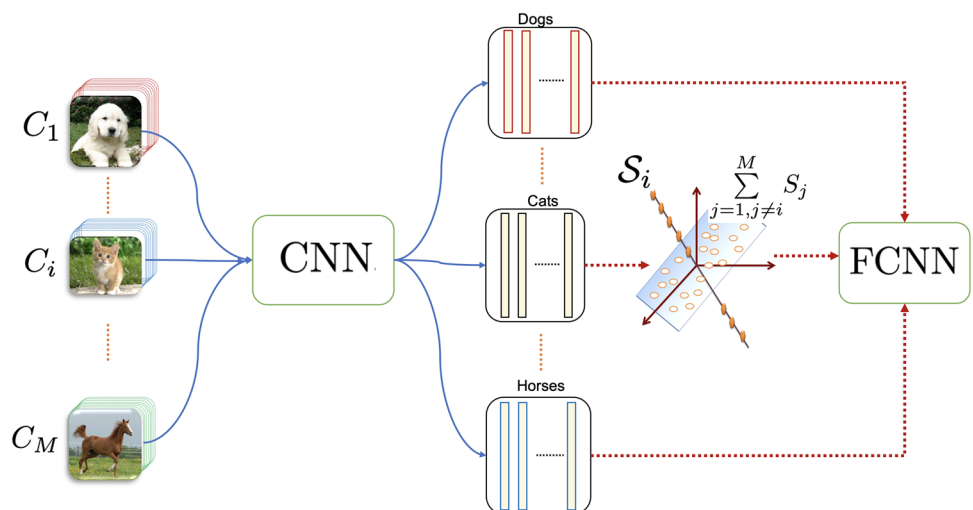


Figure 2 illustrates the approach. We consider a single DCNN and train it using all available data. This is called pre-training. After the pre-training, each input class \mathcal{C}_i is passed via CNN and a data matrix \mathbf{W}_i whose columns are features corresponding each input data in \mathcal{C}_i is constructed. Using SVD of \mathbf{W}_i , a subspace is matched to \mathbf{W}_i and it is called \mathcal{S}_i . Then, separation of \mathcal{S}_i from the sum of the remaining subspaces is computed and the network parameters are updated to minimize the separation. Algorithm 7 gives the steps for training of DCNN.

Algorithm 7: Increasing separation of feature subspaces.

Data: Input data sets $\{\mathcal{C}_i\}_{i=1}^M$
Data: Epochs: e
Data: Separation epochs: k
Result: Filter parameters for $\{\text{CNN}_i\}_{i=1}^M$, weight and biases for $\{\text{FCNN}_i\}_{i=1}^M$

```

1 Pre-training
2 for  $n : 1 \rightarrow e$  do
3   for  $i : 1 \rightarrow M$  do
4     for  $j : 1 \rightarrow M, j \neq i$  do
5       Feedforward  $\mathcal{C}_j$  via DCNN
6       Construct feature matrix  $\mathbf{W}_j$ 
7       Estimate effective rank  $r_j$  of  $\mathbf{W}_j$  using Algorithm 2
8       Compute SVD of  $\mathbf{W}_j = \mathbf{U}_j \boldsymbol{\Sigma}_j \mathbf{V}_j^T$ 
9       Find subspace  $\mathcal{S}_j = \text{span}\{\mathbf{U}_j[1 : r_j]\}$ 
10    end
11    Let  $\mathcal{U} = \sum_{j=1, j \neq i}^M \mathcal{S}_j$ 
12    Concatenate  $\mathbf{U}_j[1 : r_j]$  for  $j = 1, j \neq i$  to obtain a basis for  $\mathcal{U}$ 
13     $\mathbf{U} = [\mathbf{U}_1[1 : r_1] \dots \mathbf{U}_M[1 : r_M]]$ 
14     $\mathcal{U} = \text{span}\{\mathbf{U}\}$ 
15    for  $m : 1 \rightarrow k$  do
16      Feedforward  $\mathcal{C}_i$  via DCNN to compute  $\mathbf{W}_i$ 
17      Estimate effective rank  $r_i$  of  $\mathbf{W}_i$ 
18      Compute SVD of  $\mathbf{W}_i = \mathbf{U}_i \boldsymbol{\Sigma}_i \mathbf{V}_i^T$ 
19      Find subspace  $\mathcal{S}_i = \text{span}\{\mathbf{U}_i[1 : r_i]\}$ 
20      Update network parameters to maximize separation between  $\mathcal{S}_i$  and  $\mathcal{U}$ 
21    end
22  end
23 end
24 Train fully connected FCNN part only

```

There are different measures for separation of subspaces. Each subspace can be represented as a point in a Grassmannian manifold [36] and various distances such as geodesic arc length, chordal distance, or projection distance can be considered. In this work, the projection distance is considered as follows:

$$d^2(\mathcal{S}_i, \mathcal{U}) = \sum_{k=1}^p \sin^2(\theta_k) \quad (4)$$

where $\theta_1, \theta_2, \dots, \theta_p$ are the principal angles between \mathcal{S} and \mathcal{U} . The principal angles are calculated as follows (using concepts from Algorithm 7).

- Let \mathbf{U} and $\mathbf{U}_i[1 : r_i]$ be an orthonormal basis matrices for $\mathcal{U} = \sum_{j=1, j \neq i}^M \mathcal{S}_j$ and \mathcal{S}_i , respectively.
- If $1 \geq \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{r_i} \geq 0$ are the singular values of $\mathbf{U}^T \mathbf{U}_i[1 : r_i]$, then the principle angles are given by

$$\theta_k = \arccos(\sigma_k) \quad k = 1, \dots, r_i. \quad (5)$$

4 Results

4.1 Results for first method

MNIST [37] (handwritten digits - Figure 3), and Fashion-MNIST [38] (fashion products - Figure 4) datasets are used for testing. The MNIST data that support the findings of this study are available from the MNIST Database [<http://yann.lecun.com/exdb/mnist/>]. The Fashion-MNIST data that support the findings of this study are available in github repository [<https://github.com/zalando-research/fashion-mnist/tree/master/data/fashion>].

In order to measure impacts of size on performance, we constructed three network topologies. Another topology with dropout layer to FCNN is also considered. Finally, a network topology with only CNN without FCNN is tested. Table 1 provides more details. The proposed method is tested with five different topologies (as shown in Table 1) and compared to the results obtained with traditional DCNN approach. Out of five, three topologies are to assess the effect of size on the performance, one topology topology adds dropout layers to the fully connected network and one topology has only CNN but not FCNN. Since the last topology does not have label categories layer, it cannot be tested traditionally. The results are shown in Table 2. Our first method performs better with small sized networks. This can be observed with Topologies 2 and 3. Our method performs better with smaller number of filters. The overfitting problem for traditional DCNN training is addressed with introduction of dropout layer at Topology 4. With this, performance is improved without any impacting the new approach. This greatly improves the performance while it does not have an effect on the new approach. Due to having an iteratively refined subspace to describe a class, the subspace overfits to features that occur more. In Topology 5, the CNN achieves a high accuracy close to the one with FCNN in Topology 3. In other words, CNN is able to generate features that can be separated without an FCNN.

4.2 Results for second method

In this part, EMNIST dataset [39], that includes digits and letters, is used for testing purposes. The EMNIST (Extended MNIST) data that support the findings of this study are available from Western Sydney University repository [<https://rds.westernsydney.edu.au/Institutes/MARCS/BENS/EMNIST/emnist-gzip.zip> [39]]. We considered five network topologies to reflect different complexities and impact of dropout layer. The network topologies and the performance results are shown respectively in Table 3 for and Table 4. The experimental results show that the angle based approach performs better for all topologies. It should be noted that in order to rule out that improvements were due to retraining of FCNN, the same extra training for Network-2 was used and 96.02% performance was obtained. In other words, traditional training (CNN + FCNN) generates 94.24%, traditional training with additional FCNN training generates 96.02%, and the new architecture generates 98.37%.

Fig. 3 MNIST sample images



Fig. 4 Fashion MNIST sample images

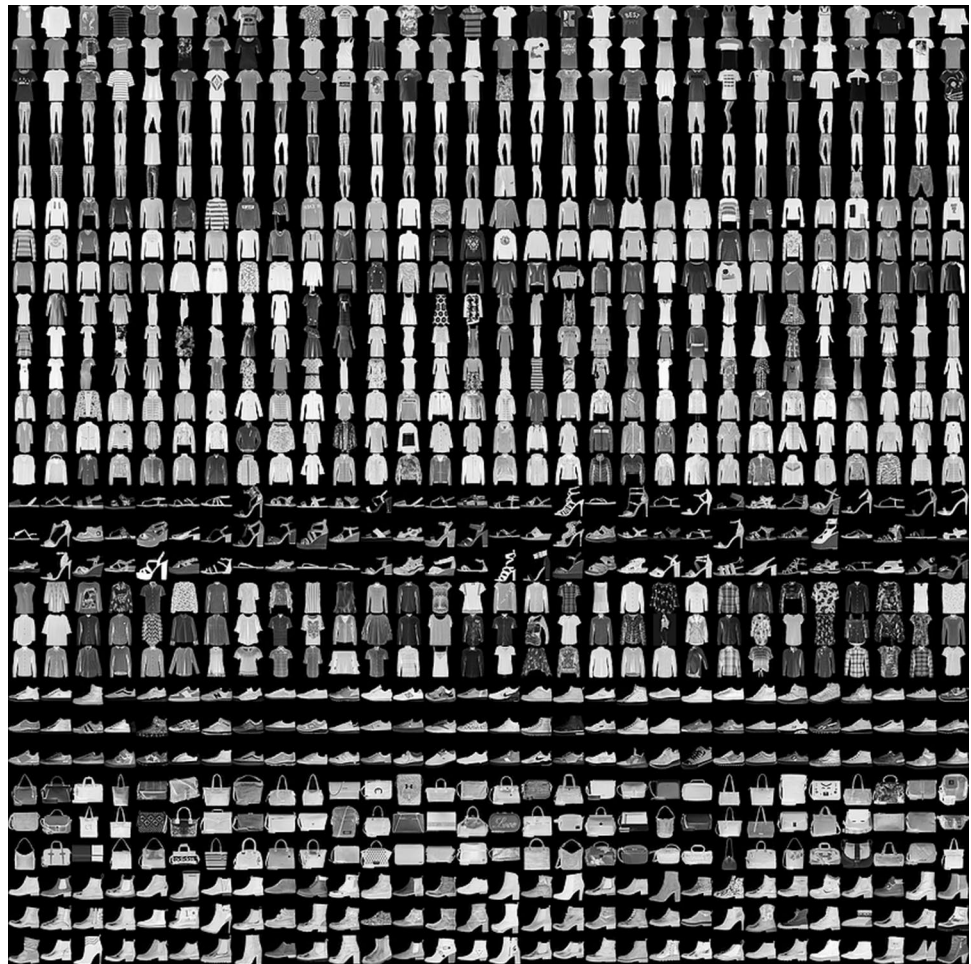


Table 1 Network topologies—first method

Topology	CNN _i Layer-1	CNN _i Layer-2	FCNN _i Layer-1	FCNN _i Layer-2
1	16 (2, 2, 1)	16 (2, 2, 16)	784 → 10	N/A
2	16 (5, 5, 1)	16 (5, 5, 16)	784 → 10	N/A
3	32 (5, 5, 1)	32 (5, 5, 32)	1568 → 10	N/A
4	32 (5, 5, 1)	64 (5, 5, 32)	3136 → 1024 DO	1024 → 10 DO
5	32 (5, 5, 1)	32 (5, 5, 32)	N/A	N/A

Table 2 Performances—first method

Topology	MNIST Traditional	MNIST New Arch.	Fash.-MNIST Traditional	Fash.-MNIST New Arch.
1	87.86%	96.48%	86.49%	90.21%
2	89.47%	98.37%	87.29%	90.77%
3	99.36%	99.3%	87.29%	91.04%
4	99.5%	99.27%	87.59%	90.79%
5	N/A	99.21%	N/A	91.23%

Table 3 Network topologies—second method

Topology	CNN Layer-1	CNN Layer-2	CNN Layer-3	FCNN Layer-1	FCNN Layer-2
1	32 (3, 3, 1)	64 (3, 3, 32)	N/A	3136 → 512	512 → 10
2	32 (5, 5, 1)	64 (5, 5, 32)	N/A	3136 → 512	512 → 10
3	32 (3, 3, 1), DO	64 (3, 3, 32), DO	N/A	3136 → 512	512 → 10
4	32 (5, 5, 1)	32 (5, 5, 32)	64 (5, 5, 32)	3136 → 512	512 → 10
5	32 (3, 3, 1)	32 (4, 4, 32)	64 (5, 5, 32)	3136 → 512	512 → 10

Table 4 Performances—second method

Topology	Traditional Perf.	New Arch. Perf.
1	94.15%	97.50%
2	94.24%	98.37%
3	94.27%	96.99%
4	94.34%	97.47%
5	91.77%	92.94%

5 Conclusions

This paper introduced two methods that aim at enhancing feature subspace separation during the training process. The first method creates multiple deep convolutional neural networks and network parameters are optimized based on projection of data on subspaces. In the second method, the geodesic distances on Grassmanian manifolds of subspaces is minimized for a particular feature subspaces and the sum of all remaining feature subspaces. As a future work, other subspace based methods, representing each feature subspace by some orthogonal subspaces and training to keep orthogonality as new data training data arrives should also improve accuracy. Such a network training may be more robust especially for adversarial effects.

6 Materials and methods

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request. The EMNIST (Extended MNIST) data that support the findings of this study are available from Western Sydney University repository [<https://rds.westernsydney.edu.au/Institutes/MARCS/BENS/EMNIST/emnist-gzip.zip> [39]]. The MNIST data that support the findings of this study are available from the MNIST Database [<http://yann.lecun.com/exdb/mnist/>]. The Fashion-MNIST data that support the findings of this study are available in github repository [<https://github.com/zaladoresearch/fashion-mnist/tree/master/data/fashion>].

Declarations

Competing interests The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th international conference on neural information processing systems, vol 1. Curran Associates Inc.; 2012. p. 1097–1105.
2. He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE international conference on computer vision (ICCV).
3. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR). 2016. p. 770–778.
4. Litjens GJ, Kooi T, Bejnordi BE, Setio AA, Ciompi F, Ghafoorian M, van der Laak JA, van Ginneken B, Sánchez CI A survey on deep learning in medical image analysis. *CoRR*, [arXiv:abs/1702.05747](https://arxiv.org/abs/1702.05747). 2017.
5. Angelova A, Krizhevsky A, Vanhoucke V, Ogale A, Ferguson D. Real-time pedestrian detection with deep network cascades. In: Proceedings of BMVC 2015. 2015.
6. Parkhi OM, Vedaldi A, Zisserman A. Deep face recognition. In: Proceedings of the British machine vision conference (BMVC). 2015.
7. Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. *CoRR*, [arXiv:abs/1708.02709](https://arxiv.org/abs/1708.02709). 2017.
8. Lample G, Charton F. Deep learning for symbolic mathematics. *CoRR*, [arXiv:abs/1912.01412](https://arxiv.org/abs/1912.01412). 2019.
9. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*. 2017;542(7639):115–8.
10. Gulshan V, Peng L, Coram M, Stumpe MC, Wu D, Narayanaswamy A, Venugopalan S, Widner K, Madams T, Cuadros J, Kim R, Raman R, Nelson PQ, Mega J, Webster D. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*. 2016;316(22):2402–10.
11. Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with alphafold. *Nature*. 2021;596:583–9.
12. Yang J, Anishchenko I, Park H, Peng Z, Ovchinnikov S, Baker D. Improved protein structure prediction using predicted interresidue orientations. *Proc Natl Acad Sci*. 2020;117(3):1496–503.
13. Stéphane M. Understanding deep convolutional networks. *Philos Trans R Soc Lond A Math Phys Eng Sci*. 2016;374(2065):20150203.
14. Zhou D-X. Theory of deep convolutional neural networks: Downsampling. *Neural Netw*. 2020;124:319–27.
15. Berner J, Grohs P, Kutyniok G, Petersen P. The modern mathematics of deep learning. *CoRR*, [arXiv:abs/2105.04026](https://arxiv.org/abs/2105.04026). 2021.
16. Kanatani K, Matsunaga C. Estimating the number of independent motions for multibody motion segmentation. In: 5th Asian conference on computer vision. 2002. p. 7–9.
17. Georgiades AS, Belhumeur PN, Kriegman DJ. From few to many: illumination cone models for face recognition under variable lighting and pose. *IEEE Trans Pattern Anal Mach Intell*. 2001;23(6):643–60.
18. Aldroubi A, Sekmen A, Koku AB, Cakmak AF. Similarity matrix framework for data from union of subspaces. *Appl Comput Harmon Anal*. 2018;45(2):425–35.
19. Aldroubi A, Hamm K, Koku AB, Sekmen A. Cur decompositions, similarity matrices, and subspace clustering. *Front Appl Math Stat*. 2019;4:65.
20. Aldroubi A, Sekmen A. Nearness to local subspace algorithm for subspace and motion segmentation. *IEEE Signal Process Lett*. 2012;19(10):704–7.
21. Vidal R. A tutorial on subspace clustering. *IEEE Signal Process Mag*. 2010;28:52–68.
22. Huang Q, Zhang Y, Peng H, Dan T, Weng W, Cai H. Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning. *Neurocomputing*. 2020;404:340–50.
23. Lv J, Kang Z, Lu X, Xu Z. Pseudo-supervised deep subspace clustering. *CoRR*, [arXiv:abs/2104.03531](https://arxiv.org/abs/2104.03531), 2021.
24. Lowe DG. Distinctive image features from scale-invariant keypoints. *Int J Comput Vis*. 2004;60(2):91–110.
25. Davis SB, Mermelstein P. Readings in speech recognition. Chapter comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. San Francisco, CA: Morgan Kaufmann Publishers Inc.; 1990. p. 65–74.
26. Kingma D, Ba J. Adam: a method for stochastic optimization. In: International conference on learning representations, 12 2014.
27. Zhu R, Dornaika F, Ruicheck Y. Semi-supervised elastic manifold embedding with deep learning architecture. *Pattern Recognit*. 2020;107:107425.
28. Chen X, Weng J, Wei L, Jiaming X, Weng J-S. Deep manifold learning combined with convolutional neural networks for action recognition. *IEEE Trans Neural Netw Learn Syst*. 2018;29(9):3938–52.
29. Dorfer M, Kelz R, Widmer G. Deep linear discriminant analysis. *CoRR*, [arXiv:abs/1511.04707](https://arxiv.org/abs/1511.04707). 2015.
30. Chan T-H, Jia K, Gao S, Lu J, Zeng Z, Ma Y. Pcanet: A simple deep learning baseline for image classification? *CoRR*, [arXiv:abs/1404.3606](https://arxiv.org/abs/1404.3606). 2014.
31. Parlaktuna M, Sekmen A, Koku AB, Abdul-Malek A. Enhanced deep learning with improved feature subspace separation. In: 2018 international conference on artificial intelligence and data processing (IDAP). 2018. p. 1–5.
32. Parlaktune M. Enhanced deep learning with improved feature subspace separation. Master's thesis, Tennessee State University, 2018.
33. Abdul-Malek A. Deep learning and subspace segmentation: theory and applications. PhD thesis, Tennessee State University, 2019.
34. Vidal R, Ma Y, Sastry S. Generalized principal component analysis (GPCA). *IEEE Trans Pattern Anal Mach Intell*. 2005;27(12):1945–59.
35. Roy O, Vetterli M. The effective rank: a measure of effective dimensionality. In: 2007 15th European signal processing conference. 2007. p. 606–610
36. Zhang J, Zhu G, Heath Jr. RW, Huang K. Grassmannian learning: embedding geometry awareness in shallow and deep learning. *CoRR*, [arXiv:abs/1808.02229](https://arxiv.org/abs/1808.02229). 2018.
37. Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proc IEEE*. 1998;86(11):2278–324.
38. Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. 2017.
39. Cohen G, Afshar S, Tapson J, van Schaik A. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, [arXiv:abs/1702.05373](https://arxiv.org/abs/1702.05373). 2017.