UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Faculty of Electrical Engineering, Computer Science and Mathematics

Institute for Electrical Engineering and Information Technology
University of Paderborn
Signal and System Theory Group
Prof. Dr. Peter Schreier

Master Thesis

Interpretable Deep Canonical Correlation Analysis Based Neural Networks

by

Yunlyu Wang
Matr.-Nr.:6898496

First Examiner: Prof. Dr. Peter Schreier
University of Paderborn

Second Examiner: Prof. Dr.-Ing. Reinhold Häb-Umbach
University of Paderborn

Supervisors: Dr.-Ing. Tanuj Hasija and Maurice Kuschel
University of Paderborn

Paderborn, December 31, 2022

Erklärung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen worden sind, sind als solche gekennzeichnet.

31.12.2022
Ort, Datum

Yunhyu Wang
Unterschrift

Abstract. We explored unsupervised multiview learning techniques based on correlation maximization. A technique is deep canonically correlated autoencoders (DCCAE). The aim of this thesis is to enhance the interpretability of DCCAE. Two solutions are proposed for this purpose. One is convolutional neural network-based DCCAE and other is visualization. It was found that DCCAE model based on convolutional neural networks outperformed the original feedforward neural network-based model. The use of a convolutional neural network allowed the DCCAE to exploit the dependent information across pixels in MNIST data and hence was able to effectively learn the shared representations between the two views. This resulted in more accurate representations and reduced the overfitting. Three different visualization techniques , which are Saliency Map, SmoothGrad, and Gradient-weighted Class Activation Mapping (GradCAM), were also applied to the proposed model to interpret the learned representations and visualize them over the learning curve of DCCAE. The results showed that Saliency Map and SmoothGrad were more effective for simpler network structures, while GradCAM was particularly useful for complex convolutional neural networks.

Contents

1 Introduction

1.1 Motivation

Multiview analysis is a fast-expanding research direction in machine learning [YHM+21]. It has developed a lot of theoretical foundations and outstanding applications [Sun13]. Multiview means using different sensors to acquire data of a common physical phenomenon. Some of the examples are shown in Fig. 1.1. For instance, simultaneously recorded audio and video of a film [KSE05] shown in Fig. 1.2, different images from the same scene [AMH+05], parallel text in two languages [VCST02], picture and text surrounding the picture, or words and context [PSM14].

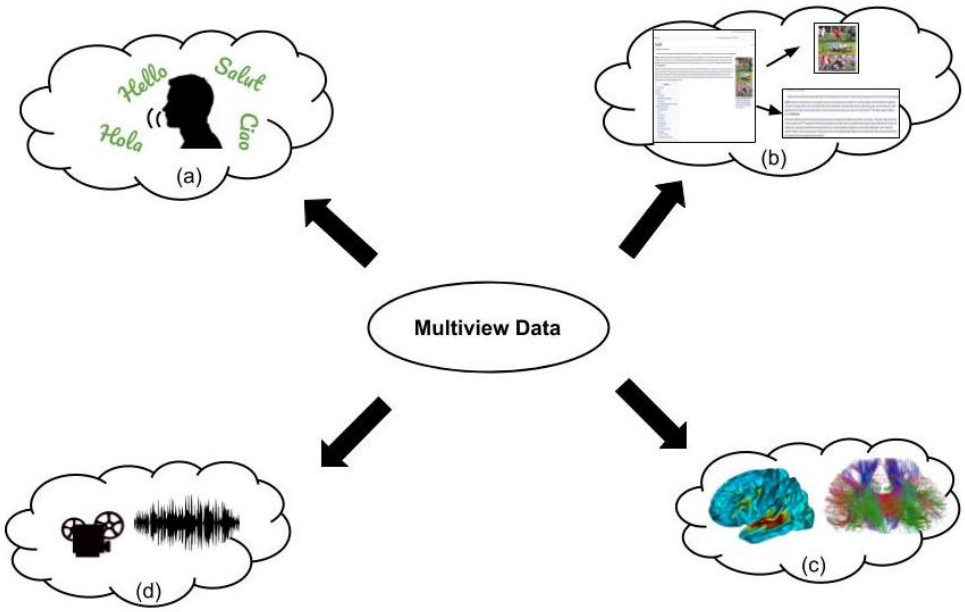

Figure 1.1: Multiview Analysis for Different Applications [Goy18]. (a) Multilingual document classification, (b) Webpage classification based on both textual and image data, (c) Medical Imaging where each brain image is represented by its MRI and t-fMRI images, (d) Multimedia data which is combination of both video and audio signals

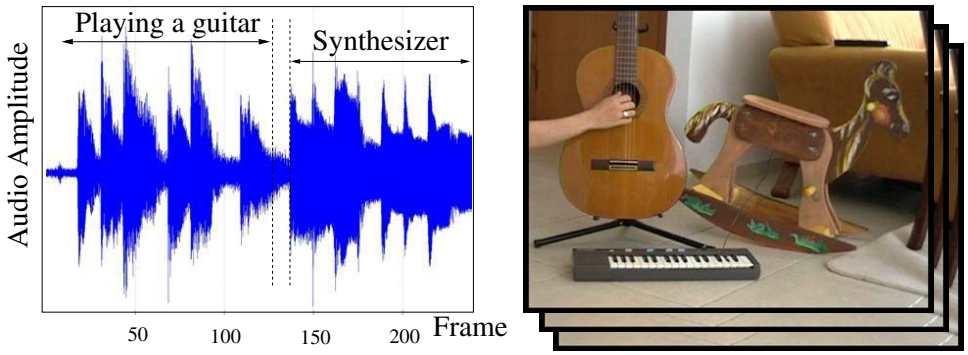

Figure 1.2: Aduio Data and Video Data [KSE05]

Multiview analysis is a learning method that not only makes use of both the information contained in each view but also the associations between different views. Therefore, multiview analysis offers more dimension of information in the data. For example, By finding the correlation between the EEG signal and music signal to create new features that lead to music recommendations [SOH21]. Multiview analysis has so far permeated several different machine learning subfields. For example, dimensionality reduction [ZZPZ16, CHC12, WZSY12] multilabel image classification [LLT+15], multiview classification [Sun11], clustering [BS05, DSGLM10, KD11], semi-supervised learning [SNB05, SR08], supervised learning [FHM+05] and image denoising [ZHJ19, JRHG12, YSYW15].

1.2 Literature Review

According to recent years of multiview analysis research, it can be categorized into two main groups: a alignment-based framework, which looks for pairwise alignment between views, and a factorization-based framework, which seeks an equivalent representation for all points of view [NW20]. In the alignment-based framework, it focuses on find the consensus between views. The consensus principle aims to maximize the agreement between distinct views of data, such as partial least squares [RK05, BR03] which tries to find the maximum covariance between each view and mulitmodel deep boltzmann machines [SS12]. In the factorization-based framework, the complementary principle, which each view may contain information that is not present in other views, is the key mind. For example, multiple kernel learning[PLI+15, GA11, WLY+19] Also, some methods belong to both groups, e.g., multiview uncorrelated locality preserving projection [YS19] and multiview spectral clustering [RS18, KD11, CNHK11]. Another mulitview analysis extension is multiview conditional random fields [SSYY20] which includes the features of multiple views in the conditional random fields by realizing the complementary principle.

For unsupervised learning, unlabeled data is the advantage but also the challenge which is that most data is unlabeled in practical application. As mentioned above, mulitview data are common in real applications and multiview analysis provides more dimensions of information which the representations attained by multiview analysis can be applied in the process. Correlation-max approaches is one of the most widely used methods for reaching the consensus in the alignment-based framework. The consensus

found by these approaches is the maximum correlation between views. With limited available data, correlation-max approaches provide a solution for a mount of applications by reaching the correlation between each view in the data. In cross-domain applications, correlation-max approaches have been proven that they are in good performance [GLWS20, HR20].

Canonical correlation analysis (CCA) [Hot92] is the classic methond in correlation-max approaches which uses a statistical mathematical method to study the correlation between multiview data. In this thesis, I focus on correlation-max approaches. In recent years of correlation-max approaches studies, kernel canonical correlation analysis (KCCA) [Aka06, MRB01, BJ02, HSST04], deep canonical correlation analysis (DCCA) [AABL13] and deep canonically correlated autoencoders(DCCAE) [WALB15a] have been widely used as extensions of CCA. Due to the limitation of CCA's ability to only find the linear projections, KCCA using kernel function project representations to a higher-dimensional non-linear feature space which increases the flexibility of the feature selection. It has been utilized for unsupervised data analysis [HMMBST07, VCST02] and natural language processing [DFU11, HLBKK08]. However, KCCA is only capable of recreating kernel on Hilbert spaces with associated kernels. The time needed to train KCCA does not scale well with the size of the training set since it is a nonparametric approach.

Unlike KCCA, DCCA uses deep neural networks to acquire flexible nonlinear representations. The problems with nonparametric models in KCCA are not present in deep neural networks. As a parameteric method, deep nerual networks can process large amounts of training data more quickly and do not need to refer to the training set when conducting tests. Further inspired by reconstructed inputs [NKK+11], DCCAE is composed of DCCA and autoencoders. It rebuilds both views from the learned representations that are available at test time. Thus, this simultaneous optimization strategy results in a trade-off between information learned across different views and information learned within each view [NW20]. In other words, DCCAE can ensure that the learned representations can well reflect the features of the original views.

1.3 Contribution of this Thesis

In this thesis, I studied correlation-max approaches and extended them specifically to applications where there is dependency in the input data, for example, pixel-to-pixel dependency in images or sample-to-sample dependency in time-series data. My proposed model is a DCCAE model based on Convolutional Neural Networks(CNN) [LBBH98a]. It is an unsupervised learning model based on image multiview processing combined with different neural network structures.

Feedforward neural networks have been shown to not fully leverage the dependency information of the input data such as in images and time-series data despite it is easy to designed and comparatively fast to process. When processing images, feedforward neural network is significantly inferior compared to CNN. CNN offers very remarkable results in the areas of semantic parsing, paraphrase detection, and image and video recognition. Because its features are shared parameters, less number of weights, and without losing spatial information. By taking advantage of CNN in image processing, the proposed model combines DCCAE with CNN should give a better performance in multiview analysis.

Another challenge is the disadvantage of correlation-max approaches which prone to

overfitting, i.e., neural network can blindly maximize correlation between each view by projecting complex nonlinearity without learning the shared properties in the data. I investigated if the CNN-based model leads to a better performance in which there is less overfitting compared to the feedforward neural network model.

Furthermore, the next challenge is how to realize the interpretability of this model. what can be used to visualize the highly correlated learned representation and what the correlation learned from the shared properties in the data by the model is have not yet been well studied. Besides, if the overfitting can be visualized when should the model stop the learning? In this thesis, I proposed multiple techniques for visualization. Saliency Map [MKH+95] is the most basic way to visualize deep learning model. It is usually applied to visualize classifiers. SmoothGrad [STK+17] as a improved approach of Saliency Map can offer a better results by using adding noise to de-noise. To visualize CNN, Gradient-weighted Class Activation Mapping(GradCAM) [SCD+17] is a popular technique. It has been proven to have a good performance in various applications [CCHM20]. It is necessary to figure out the basic thinking in visualization and consider the noise during the experiment. Therefore, Saliency Map and SmoothGrad are applied in this thesis. Since GradCAM has an outstanding performance[AGGK18], the CNN-based model also uses it to reach interpretability.

1.4 Outline of this Thesis

The remainder of this thesis proceeds as follows. Chapter 2 introduces the fundamental theories on correlation-max approaches. In section 2.1, how to find the linear projection by CCA is in detailed presented. Two main solution in CCA are eigenvalue decomposition in subsection 2.1.1 and singular value decomposition in subsection 2.1.2. Taking advantage of nonlinearity offerd by a feedforward neural network, in section 2.2 DCCA overcomes the limitation of CCA. In section 2.3, DCCAE not only consider to maximize the correlation between each view but also minimize the error between the reconstructed views and original views.

Chapter 3 consists in two sections. How this proposed model consists in is summarized in section 3.1. Experimental data is included in subsection 3.1.1. Subsection 3.1.2 explains how to build a CNN. The use of the external classifier and clustering is in subsection 3.1.3 and visualization methods are introduced in section 3.2.

Then in Chapter 4, it analysed the results of the proposed model. The comparison results between two neural network structure based models is summarized in section 4.1. Experiment by tuning the hyper-parameters to reach the optimum settings is included in section 4.2. With multiple visualization techniques, evaluation visualization results are in subsection 4.3.1. Overfitting problem evaluation is in subsection 4.3.2. Eventually, the conclusion and outlook are given in Chapter 5.

2 Theoretical Foundations

In the multi-view analysis, canonical correlation analysis(CCA) [Hot92, Nie02, RSSTG13] is a classical method to find the linear projections which are maximally correlated between the two views. However, linear transformations are restrictive for numerous practical applications. Deep canonical correlation analysis(DCCA) [AABL13, LWB+15, WALB15b, WALS15] is the extension of CCA which can simultaneously learn two nonlinear projections of two views that are maximally correlated. Moreover, deep canonically correlated autoencoders(DCCAE) [WALB15a] is an extension of DCCA which ensures that the learned representations can well reflect the features of the original views.

2.1 Canonical Correlation Analysis

Canonical correlation analysis(CCA) is a statistical method for analysing correlation between two views. CCA finds linear projections of two views with that the projected representations are maximally correlated.

Suppose there are two random vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^n$ with n dimensions. Assume $\boldsymbol{x}_1, \boldsymbol{x}_2$ are mean centered. Project the original data $\boldsymbol{x}_1, \boldsymbol{x}_2$ into a latent space by transformations $\boldsymbol{u}_1, \boldsymbol{u}_2$, to get the new representations z_1, z_2.

$$\begin{aligned} z_1 &= \boldsymbol{u}_1^\intercal \boldsymbol{x}_1 \ , \\ z_2 &= \boldsymbol{u}_2^\intercal \boldsymbol{x}_2 \ . \end{aligned}$$

Now the CCA objective can be defined as

$$\rho = \max_{\boldsymbol{u}_1, \boldsymbol{u}_2} corr(z_1, z_2) \ , \tag{2.1}$$

and this is shown as Fig. 2.1

The correlation in (2.1) Pearson correlation coefficient [Pea96], defined as

$$\begin{aligned} corr(z_1, z_2) &= \frac{\mathbb{E}(z_1 z_2)}{\sqrt{\mathbb{E}(z_1^2)\,\mathbb{E}(z_2^2)}} \\ &= \frac{\boldsymbol{u}_1^\intercal\, \mathbb{E}(\boldsymbol{x}_1 \boldsymbol{x}_2^\intercal) \boldsymbol{u}_2}{\sqrt{\boldsymbol{u}_1^\intercal\, \mathbb{E}(\boldsymbol{x}_1 \boldsymbol{x}_1^\intercal) \boldsymbol{u}_1 \boldsymbol{u}_2^\intercal\, \mathbb{E}(\boldsymbol{x}_2 \boldsymbol{x}_2^\intercal) \boldsymbol{u}_2}} \ , \end{aligned} \tag{2.2}$$

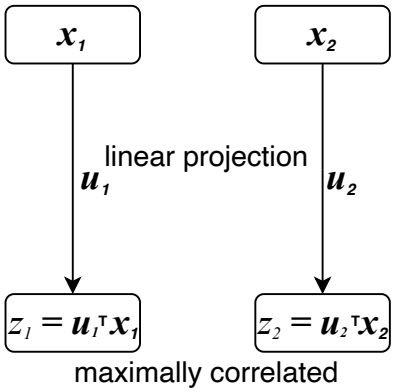

Figure 2.1: Schematic of Canonical Correlation Analysis for handling random vectors

where the $\mathbb{E}(\boldsymbol{x}_1\boldsymbol{x}_1^\intercal)$ and $\mathbb{E}(\boldsymbol{x}_2\boldsymbol{x}_2^\intercal)$ are auto-covariance matrices where correlated projections to obtain, $\mathbb{E}(\boldsymbol{x}_1\boldsymbol{x}_2^\intercal)$ is cross-covariance matrix. To obtain first correlated projections, the augmented covariance matrix $\boldsymbol{\Sigma}$ and other matrices are defined as below

$$\begin{aligned}
\boldsymbol{\Sigma} &= \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} , \\
\boldsymbol{\Sigma}_{11} &= \mathbb{E}(\boldsymbol{x}_1\boldsymbol{x}_1^\intercal) , \\
\boldsymbol{\Sigma}_{22} &= \mathbb{E}(\boldsymbol{x}_2\boldsymbol{x}_2^\intercal) , \\
\boldsymbol{\Sigma}_{12} &= \mathbb{E}(\boldsymbol{x}_1\boldsymbol{x}_2^\intercal) , \\
\boldsymbol{\Sigma}_{21} &= \mathbb{E}(\boldsymbol{x}_2\boldsymbol{x}_1^\intercal) .
\end{aligned} \tag{2.3}$$

So the equation (2.1) can be rewritten as

$$\rho = \max_{\boldsymbol{u}_1,\boldsymbol{u}_2} \frac{\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12}\boldsymbol{u}_2}{\sqrt{\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2}} . \tag{2.4}$$

Moreover, notice that the solution of equation (2.4) is independent with the scalar of the projections $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. For example, if $u_1 = \alpha\boldsymbol{u}_1$ where α is a scalar,

$$\begin{aligned}
\frac{\alpha\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12}\boldsymbol{u}_2}{\sqrt{\alpha\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\alpha\boldsymbol{u}_1 \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2}} &= \frac{\alpha\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12}\boldsymbol{u}_2}{\sqrt{\alpha^2\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2}} \\
&= \frac{\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12}\boldsymbol{u}_2}{\sqrt{\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2}} .
\end{aligned}$$

By setting the projections to unit variance, $\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 = \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2 = 1$, the equation (2.4) now can be written as

$$\begin{aligned}
\rho = &\max_{\boldsymbol{u}_1,\boldsymbol{u}_2} \ \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12}\boldsymbol{u}_2 \\
&\textbf{s.t.} \ \ \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 = \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22}\boldsymbol{u}_2 = 1 .
\end{aligned} \tag{2.5}$$

To obtain next set of projections, $\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}$, for $i = 2, 3, \cdots, n$ the same process of maximal $\rho^{(i)}$ is followed but with a constrain that projections are uncorrelated within

each view

$$\begin{aligned} \rho^{(i)} = \max_{\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}} \; & \boldsymbol{u}_1^{(i)\intercal} \boldsymbol{\Sigma}_{12} \boldsymbol{u}_2^{(i)} \\ \textbf{s.t.} \; & \boldsymbol{u}_1^{(i)\intercal} \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1^{(i)} = \boldsymbol{u}_2^{(i)\intercal} \boldsymbol{\Sigma}_{22} \boldsymbol{u}_2^{(i)} = 1 \; , \\ & \boldsymbol{u}_1^{(i)\intercal} \boldsymbol{u}_1^{(j)} = \boldsymbol{u}_2^{(i)\intercal} \boldsymbol{u}_2^{(j)} = 0 \textbf{ for } i \neq j \; . \end{aligned} \tag{2.6}$$

The optimization of CCA objective can be solved by two methods, eigenvalue decomposition (EVD) [HSST04, And62, VDW77] and the singular value decomposition (SVD) [WPJ+18]

2.1.1 Optimization by Eigenvalue Decomposition

According to the objective formulation (2.5) , the corresponding Lagrangian is

$$\mathcal{L}(\lambda_1, \lambda_2, \boldsymbol{u}_1, \boldsymbol{u}_2) = \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12} \boldsymbol{u}_2 - \frac{\lambda_1}{2} (\boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 - 1) - \frac{\lambda_2}{2} (\boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22} \boldsymbol{u}_2 - 1) \; . \tag{2.7}$$

Taking the derivatives of $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, considering the covariance matrix $\boldsymbol{\Sigma}_{11}$ and $\boldsymbol{\Sigma}_{22}$ are symmetrical, i.e., $\boldsymbol{\Sigma}_{11} = \boldsymbol{\Sigma}_{11}^\intercal$,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_1} = \boldsymbol{\Sigma}_{12} \boldsymbol{u}_2 - \lambda_1 \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 = 0 \; , \tag{2.8}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_2} = \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1 - \lambda_2 \boldsymbol{\Sigma}_{22} \boldsymbol{u}_2 = 0 \; . \tag{2.9}$$

Using $\boldsymbol{u}_1^\intercal$ times the equation (2.8) and $\boldsymbol{u}_2^\intercal$ times the equation (2.9). It can get

$$\begin{aligned} 0 &= \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{12} \boldsymbol{u}_2 - \boldsymbol{u}_1^\intercal \lambda_1 \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 - \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1 + \boldsymbol{u}_2^\intercal \lambda_2 \boldsymbol{\Sigma}_{22} \boldsymbol{u}_2 \\ &= \lambda_2 \boldsymbol{u}_2^\intercal \boldsymbol{\Sigma}_{22} \boldsymbol{u}_2 - \lambda_1 \boldsymbol{u}_1^\intercal \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 \\ &= \lambda_2 - \lambda_1 \; . \end{aligned} \tag{2.10}$$

So $\lambda_1 = \lambda_2$, set $\lambda = \lambda_1 = \lambda_2$. Assuming the $\boldsymbol{\Sigma}_{22}$ is invertible,

$$\boldsymbol{u}_2 = \frac{\boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1}{\lambda} \; . \tag{2.11}$$

Plugging the equation (2.11) into equation (2.8) one gets

$$\begin{aligned} \frac{\boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1}{\lambda} - \lambda \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 = 0 \; , \\ \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1 = \lambda^2 \boldsymbol{\Sigma}_{11} \boldsymbol{u}_1 \; . \end{aligned} \tag{2.12}$$

The equation (2.12) is a generalised eigenproblem of form $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$ [Wil88]. With the solution by finding the eigenvectors of equation (2.12), then the corresponding $\boldsymbol{u}_2$ can be found by substituting $\boldsymbol{u}_1$ into equation (2.11). Assuming $\boldsymbol{\Sigma}_{11}$ is invertible, it can rewritten into a symmetric eigenproblem of the form $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$, i.e.,

$$\boldsymbol{\Sigma}_{11}^{-1} \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-1} \boldsymbol{\Sigma}_{21} \boldsymbol{u}_1 = \lambda^2 \boldsymbol{u}_1 \; . \tag{2.13}$$

The corresponding value $\boldsymbol{u}_1$ can be found by solving the eigenproblem (2.13). Similarly, solving the equation (2.14) can get the solution of $\boldsymbol{u}_2$.

$$\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\boldsymbol{u}_2 = \lambda^2\boldsymbol{u}_2 \ , \tag{2.14}$$

alternately, the value of $\boldsymbol{u}_2$ can be found by equation (2.11) after $\boldsymbol{u}_1$ be found. The following projections $\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}, i = 2, 3, \cdots, n$ can be shown to be eigenvectors of $\boldsymbol{\Sigma_{11}^{-1}\Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}}$ and $\boldsymbol{\Sigma_{22}^{-1}\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}}$ corresponding to the second largest, third largest eigenvalues and so on.

2.1.2 Optimization by Singular Value Decomposition

The singular value decomposition(SVD) of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ can be written as $\boldsymbol{A} = \boldsymbol{LDR}^\intercal$, where $\boldsymbol{D} \in \mathbb{R}^{m\times n}$ is the diagonal matrix, $\boldsymbol{L} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ denote the left and right singular matrix with constrains $\boldsymbol{L}^\intercal\boldsymbol{L} = \boldsymbol{I}$ and $\boldsymbol{R}^\intercal\boldsymbol{R} = \boldsymbol{I}$.

To find the Lagrangian optimization of CCA by SVD, it has to use the feature of orthonormal basis constituted by the left and right singular vectors. Set

$$\boldsymbol{w}_1 = \boldsymbol{\Sigma}_{11}^{-\frac{1}{2}}\boldsymbol{u}_1, \ \boldsymbol{w}_2 = \boldsymbol{\Sigma}_{22}^{-\frac{1}{2}}\boldsymbol{u}_2 \ , \tag{2.15}$$

rewrite the constrains of CCA

$$\begin{aligned} \boldsymbol{u}_1^\intercal\boldsymbol{\Sigma}_{11}\boldsymbol{u}_1 = \boldsymbol{w}_1^\intercal\boldsymbol{w}_1 = 1 \ , \\ \boldsymbol{u}_2^\intercal\boldsymbol{\Sigma}_{22}\boldsymbol{u}_2 = \boldsymbol{w}_2^\intercal\boldsymbol{w}_2 = 1 \ . \end{aligned} \tag{2.16}$$

simultaneously for simplicity, let

$$\boldsymbol{T} = \boldsymbol{\Sigma}_{11}^{-\frac{1}{2}}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-\frac{1}{2}} \ , \tag{2.17}$$

and the objective of CCA now can be written

$$\begin{aligned} \rho = &\max_{\boldsymbol{u}_1,\boldsymbol{u}_2} \ \boldsymbol{w}_1^\intercal\boldsymbol{\Sigma}_{11}^{-\frac{1}{2}}\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-\frac{1}{2}}\boldsymbol{w}_2 \\ = &\max_{\boldsymbol{u}_1,\boldsymbol{u}_2} \ \boldsymbol{w}_1^\intercal\boldsymbol{T}\boldsymbol{w}_2^\intercal \\ &\textbf{s.t. } \boldsymbol{w}_1^\intercal\boldsymbol{w}_1 = \boldsymbol{w}_2^\intercal\boldsymbol{w}_2 = 1 \ , \end{aligned} \tag{2.18}$$

The corresponding Lagrangian is

$$\mathcal{L}(\theta_1, \theta_2, \boldsymbol{w}_1, \boldsymbol{w}_2) = \boldsymbol{w}_1^\intercal\boldsymbol{T}\boldsymbol{w}_2 - \frac{\theta_1}{2}(\boldsymbol{w}_1^\intercal\boldsymbol{w}_1 - 1) - \frac{\theta_2}{2}(\boldsymbol{w}_2^\intercal\boldsymbol{w}_2 - 1) \ . \tag{2.19}$$

Taking the derivatives of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_1} = \boldsymbol{T}\boldsymbol{w}_2 - \theta_1\boldsymbol{w}_1 = 0 \ , \tag{2.20}$$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{w}_2} = \boldsymbol{T}^\intercal\boldsymbol{w}_1 - \theta_2\boldsymbol{w}_2 = 0 \ . \tag{2.21}$$

Using $\boldsymbol{w}_1^\intercal$ times the equation (2.20) and $\boldsymbol{w}_2^\intercal$ times the equation (2.21). It can get

$$\begin{aligned} \boldsymbol{w}_1^\intercal \boldsymbol{T} \boldsymbol{w}_2 - \theta_1 \boldsymbol{w}_1^\intercal \boldsymbol{w}_1 = 0 \ , \\ \boldsymbol{w}_2^\intercal \boldsymbol{T}^\intercal \boldsymbol{w}_1 - \theta_2 \boldsymbol{w}_2^\intercal \boldsymbol{w}_2 = 0 \ . \end{aligned} \tag{2.22}$$

substituting (2.16), it can be

$$\begin{aligned} \boldsymbol{w}_1^\intercal \boldsymbol{T} \boldsymbol{w}_2 - \theta_1 = 0 \ , \\ \boldsymbol{w}_2^\intercal \boldsymbol{T}^\intercal \boldsymbol{w}_1 - \theta_2 = 0 \ . \end{aligned} \tag{2.23}$$

so it can derive $\theta_1 = \theta_2$. Let $\theta = \theta_1 = \theta_2$, rewrite equation (2.20) and (2.21)

$$\begin{bmatrix} -\theta & \boldsymbol{T} \\ \boldsymbol{T}^\intercal & -\theta \end{bmatrix} \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} = \boldsymbol{0} \ . \tag{2.24}$$

Therefore, the question of CCA can be solved by finding the equation (2.24) which is a SVD problem of matrix $\boldsymbol{T}$ when the left and right singular vectors are $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Similarly, $\boldsymbol{u}_1^{(i)}, \boldsymbol{u}_2^{(i)}$ can be obtained as left singular and right singular vector of second largest singular value, third largest value ans so on.

2.1.3 CCA with Samples

Now, assume N i.i.d. samples of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are collected in $\boldsymbol{X}_1 \in \mathbb{R}^{n \times N}$, $\boldsymbol{X}_2 \in \mathbb{R}^{n \times N}$

$$\boldsymbol{X}_1 = [\boldsymbol{x}_1(1), \boldsymbol{x}_1(2), ..., \boldsymbol{x}_1(N)] \ ,$$

$$\boldsymbol{X}_2 = [\boldsymbol{x}_2(1), \boldsymbol{x}_2(2), ..., \boldsymbol{x}_2(N)] \ .$$

The covariance matrices are estimated as

$$\begin{aligned} \hat{\boldsymbol{\Sigma}}_{11} = \frac{1}{N} \boldsymbol{X}_1 \boldsymbol{X}_1^\intercal \ , \\ \hat{\boldsymbol{\Sigma}}_{22} = \frac{1}{N} \boldsymbol{X}_2 \boldsymbol{X}_2^\intercal \ . \end{aligned}$$

CCA can find the linear projections $\boldsymbol{U}_1, \boldsymbol{U}_2$ with that the projected views are maximally correlated, as shown in Fig. 2.2.

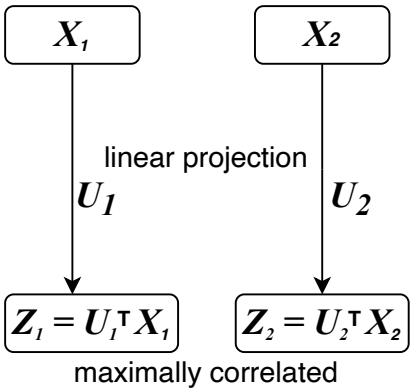

Figure 2.2: Schematic of Canonical Correlation Analysis for handling samples of random vectors

Here $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ contain all N projections as its columns, i.e.

$$\boldsymbol{U}_1 = [\boldsymbol{u}_1^{(1)}, \boldsymbol{u}_1^{(2)}, ..., \boldsymbol{u}_1^{(N)}] \ ,$$

$$\boldsymbol{U}_2 = [\boldsymbol{u}_2^{(1)}, \boldsymbol{u}_2^{(2)}, ..., \boldsymbol{u}_2^{(N)}] \ .$$

The CCA objective function is

$$\begin{aligned} \max_{\boldsymbol{U}_1, \boldsymbol{U}_2} \ & \mathrm{Tr}(\boldsymbol{U}_1^\intercal \boldsymbol{\Sigma}_{12} \boldsymbol{U}_2) \\ \text{s.t.} \ & \boldsymbol{U}_1^\intercal \boldsymbol{\Sigma}_{11} \boldsymbol{U}_1 = \boldsymbol{I} \ , \\ & \boldsymbol{U}_2^\intercal \boldsymbol{\Sigma}_{22} \boldsymbol{U}_2 = \boldsymbol{I} \ , \\ & \boldsymbol{u}_1^{(i)\intercal} \boldsymbol{X}_1 \boldsymbol{X}_2^\intercal \boldsymbol{u}_2^{(j)} = \boldsymbol{0}, \text{for } i \neq j \ . \end{aligned} \tag{2.25}$$

Let it define

$$\boldsymbol{T} \triangleq \boldsymbol{\Sigma}_{11}^{-\frac{1}{2}} \boldsymbol{\Sigma}_{12} \boldsymbol{\Sigma}_{22}^{-\frac{1}{2}} \ , \tag{2.26}$$

and take singular value decomposition of $\boldsymbol{T}$ as

$$\boldsymbol{T} = \boldsymbol{F} \boldsymbol{K} \boldsymbol{G}^\intercal \ .$$

The solution of (2.25) is

$$(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) = (\hat{\boldsymbol{\Sigma}}_{11}^{-\frac{1}{2}} \boldsymbol{F}, \ \hat{\boldsymbol{\Sigma}}_{22}^{-\frac{1}{2}} \boldsymbol{G}) \tag{2.27}$$

When covariance matrix are ill-conditioned, i.e., $\hat{\boldsymbol{\Sigma}}_{11}$ is singular, they can be replaced with their regularized version

$$\begin{aligned} \boldsymbol{\Sigma}_{11} &= \frac{1}{N} \ \boldsymbol{X}_1 \boldsymbol{X}_1^\intercal + r_1 \boldsymbol{I} \ , \\ \boldsymbol{\Sigma}_{22} &= \frac{1}{N} \ \boldsymbol{X}_2 \boldsymbol{X}_2^\intercal + r_2 \boldsymbol{I} \ , \end{aligned} \tag{2.28}$$

where r_1 and r_2 are regularization parameters [DBDM03].

2.2 Deep Canonical Correltion Analysis

Due to the limitation of CCA, its extensions of CCA namely Deep Canonical Correlation Analysis (DCCA) [AABL13, LWB+15, WALB15b, WALS15] and Deep Canonically Correlated Autoencoders (DCCAE) [WALB15a], are introducted in section 2.2 and section 2.3.

Considering more practical applications, there is more complex and nonlinear association in real data, DCCA processes the inputs first into two deep neural networks (DNNs) then applies CCA to get the maximal correlated representations.

Assume the two views data of the form are two random multidimensional variables $\boldsymbol{X}_1 \in \mathbb{R}^{n \times N}$, $\boldsymbol{X}_2 \in \mathbb{R}^{n \times N}$, $\mathbf{f}_1$ and $\mathbf{f}_2$ denote mapping implemented by DNNs, with the corresponding weight matrix $(\boldsymbol{W}_1, \boldsymbol{b}_1)$ represented by $\mathbf{W}_{\mathbf{f}_1}$ and $(\boldsymbol{W}_2, \boldsymbol{b}_2)$ represented by $\mathbf{W}_{\mathbf{f}_2}$, as shown in Fig. 2.3.

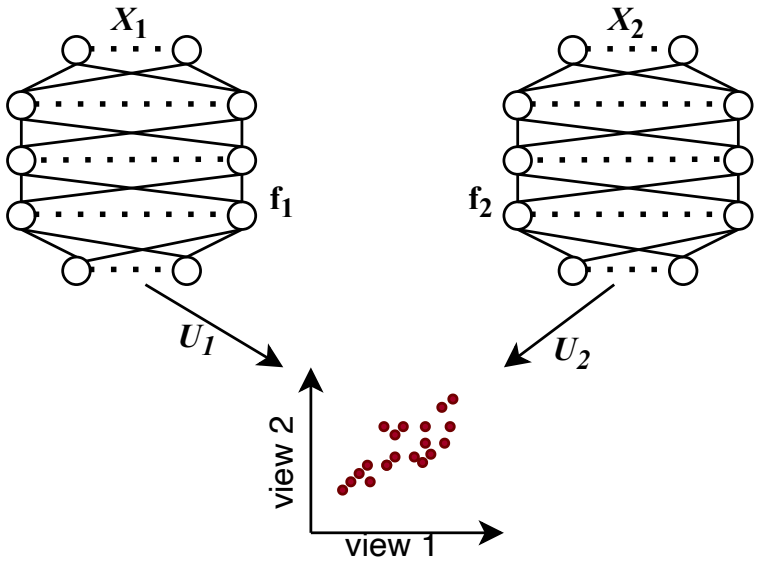

Figure 2.3: Schematic of Deep Canonical Correlation Analysis

The output of the first layer for the example $\boldsymbol{x}_1(1) \in \mathbb{R}^n$ is $\boldsymbol{h}_1^{(1)} \in \mathbb{R}^{c_1}$

$$\boldsymbol{h}_1^{(1)} = s_1(\boldsymbol{W}_1^{(1)}\boldsymbol{x}_1(1) + \boldsymbol{b}_1^{(1)}) \ , \tag{2.29}$$

where $\boldsymbol{W}_1^{(1)} \in \mathbb{R}^{c_1 \times n}$ is the first layer weight matrix, $\boldsymbol{b}_1^{(1)} \in \mathbb{R}^{c_1}$ is the bias vector, and c_1 is the number of first layer output units. $\boldsymbol{h}_1^{(1)}$ is the output of first layer. With the same calculation, $\boldsymbol{h}_1^{(2)}, \boldsymbol{h}_1^{(3)} ..., \boldsymbol{h}_1^{(d_1-1)}$ are the each output of each layers in $\mathbf{f}_1 \in \mathbb{R}^{o_1}$ where d_1 is the layers of $\mathbf{f}_1$ and o_1 is the output units of the last layer. $s_1 : \mathbb{R} \mapsto \mathbb{R}$ represents the nonlinear mapping function, as shown in Fig. 2.4. The last layer ouptut can be represented as

$$\mathbf{f}_1(\boldsymbol{x}_1(1)) = \boldsymbol{h}_1^{(d_1)}(\boldsymbol{x}_1(1)) \ . \tag{2.30}$$

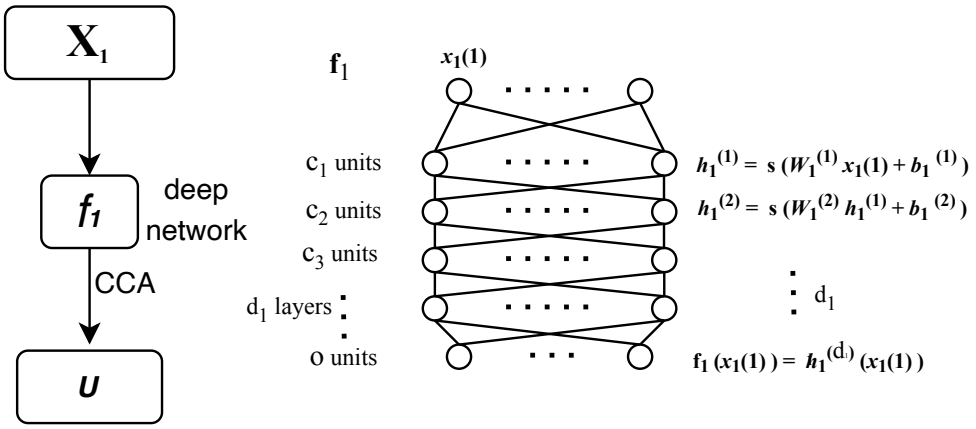

Figure 2.4: Schematic of Deep Neural Network in DCCA

Similarly, an instance $\boldsymbol{x}_2(1) \in \mathbb{R}^n$ from view two $\boldsymbol{X}_2$ is feed into $\mathbf{f}_2 \in \mathbb{R}^{o_2}$ to get the outputs with the corresponding parameters $\boldsymbol{W}_2^{(d_2)}$ and $\boldsymbol{b}_2^{(d_2)}$

$$\begin{aligned} \boldsymbol{h}_2^{(1)} &= s_2(\boldsymbol{W}_2^{(1)}\boldsymbol{x}_2(1) + \boldsymbol{b}_2^{(1)}) \ , \\ \boldsymbol{h}_2^{(2)} &= s_2(\boldsymbol{W}_2^{(2)}\boldsymbol{h}_2^{(1)} + \boldsymbol{b}_2^{(2)}) \ , \\ &\vdots \\ \mathbf{f}_2(\boldsymbol{x}_2^{(1)}) &= \boldsymbol{h}_2^{(d_2)}(\boldsymbol{x}_2(1)) \ . \end{aligned} \tag{2.31}$$

Therefore, the objective of DCCA can be represented as that if there are $\boldsymbol{X}_1 \in \mathbb{R}^{n\times N}$,$\boldsymbol{X}_2 \in \mathbb{R}^{n\times N}$, $\mathbf{f}_1$ and $\mathbf{f}_2$ denote mapping implemented by DNNs, with a corresponding set of learnable parameters, $\mathbf{W}_{\mathbf{f}_1}$ and $\mathbf{W}_{\mathbf{f}_2}$

$$\begin{aligned} \max_{\mathbf{W}_{\mathbf{f}_1},\mathbf{W}_{\mathbf{f}_2},\boldsymbol{U}_1,\boldsymbol{U}_2} \ & \frac{1}{N}\operatorname{Tr}(\boldsymbol{U}_1^\intercal \mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_2(\boldsymbol{X}_2)^\intercal \boldsymbol{U}_2) \\ \text{s.t. } & \boldsymbol{U}_1^\intercal(\frac{1}{N}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_1(\boldsymbol{X}_1)^\intercal + r_1\boldsymbol{I})\boldsymbol{U}_1 = \boldsymbol{I} \ , \\ & \boldsymbol{U}_2^\intercal(\frac{1}{N}\mathbf{f}_2(\boldsymbol{X}_2)\mathbf{f}_2(\boldsymbol{X}_2)^\intercal + r_2\boldsymbol{I})\boldsymbol{U}_2 = \boldsymbol{I} \ , \\ & \boldsymbol{u}_1^{(i)\intercal}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_2(\boldsymbol{X}_2)^\intercal \boldsymbol{u}_2^{(j)} = 0, \text{for } i \neq j \ . \end{aligned} \tag{2.32}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the CCA projections that project the DNN outputs, r_1 and r_2 are regularization parameters for sample covariance estimation.

The new representations after DCCA are

$$\begin{aligned} \tilde{\mathbf{f}}_1(\boldsymbol{X}_1) &= \boldsymbol{U}_1^\intercal \mathbf{f}_1(\boldsymbol{X}_1) \ , \\ \tilde{\mathbf{f}}_2(\boldsymbol{X}_2) &= \boldsymbol{U}_2^\intercal \mathbf{f}_2(\boldsymbol{X}_2) \ . \end{aligned} \tag{2.33}$$

Let $\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}$, $\boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2}$ and $\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_2}$ be the auto-covariance and cross-covariance matrices of the projected views $\tilde{\mathbf{f}}_1(\boldsymbol{X}_1)$ and $\tilde{\mathbf{f}}_2(\boldsymbol{X}_2)$, they are defined as

$$\begin{aligned} \boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1} &= \frac{1}{N}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_1(\boldsymbol{X}_1)^\intercal + r_1\boldsymbol{I} \ , \\ \boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2} &= \frac{1}{N}\mathbf{f}_2(\boldsymbol{X}_2)\mathbf{f}_2(\boldsymbol{X}_2)^\intercal + r_2\boldsymbol{I} \ , \\ \boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_2} &= \frac{1}{N}\mathbf{f}_2(\boldsymbol{X}_2)\mathbf{f}_2(\boldsymbol{X}_2)^\intercal \ . \end{aligned} \tag{2.34}$$

and similar to the definition (2.17) in CCA, $\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1}$ in the DCCA can be written as

$$\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1} = \boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}^{-\frac{1}{2}}\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_2}\boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2}^{-\frac{1}{2}} \ . \tag{2.35}$$

The SVD of $\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1}$ is

$$\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1} = \tilde{\boldsymbol{F}}\boldsymbol{\Lambda}\tilde{\boldsymbol{G}}^\intercal \ , \tag{2.36}$$

where the $\tilde{\boldsymbol{F}}$ and $\tilde{\boldsymbol{G}}$ are its rank N left and right singular matrix, $\boldsymbol{\Lambda}$ is the diagonal matrix with its singular values $\boldsymbol{\sigma}_1, \cdots, \boldsymbol{\sigma}_p$, where $p \leq n$.

The optimum of equation (2.32) is

$$(\boldsymbol{U}_1^*, \boldsymbol{U}_2^*) = (\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}^{-\frac{1}{2}}\tilde{\boldsymbol{F}}, \ \boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2}^{-\frac{1}{2}}\tilde{\boldsymbol{G}}) \ . \tag{2.37}$$

It should be noted that DCCA cannot effectively use stochastic gradient descent (SGD) with too small batch sizes. Observe the constraints in the objective (2.32) of DCCA, the auto-variance matrix are calculated by the N samples. If the number of training samples is too small, it cannot reliably estimate the covariance matrices. Moreover, calculating $\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1}$ in (2.36) requires $\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}$ and $\boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2}$ to be invertible. Regularization parameters r_1 and r_2 promise the invertibility when they are ill conditional.

In DCCA algorithm, to compute the gradient of the objective with respect to $(\mathbf{W}_{\mathbf{f}_1}, \mathbf{W}_{\mathbf{f}_2})$, first get the gradient of $(\mathbf{f}_1(\boldsymbol{X}_1), \mathbf{f}_2(\boldsymbol{X}_2))$ as

$$\begin{aligned} \frac{\partial \sum_{j=1}^{N} \boldsymbol{\sigma}_j}{\partial \mathbf{f}_1(\boldsymbol{X}_1)} &= 2\Delta_{\mathbf{f}_1\mathbf{f}_1}\mathbf{f}_1(\boldsymbol{X}_1) + \Delta_{\mathbf{f}_1\mathbf{f}_2}\mathbf{f}_1(\boldsymbol{X}_1) \ , \\ \text{with } \Delta_{\mathbf{f}_1\mathbf{f}_1} &= -\frac{1}{2}\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}^{-\frac{1}{2}}\tilde{\boldsymbol{F}}\boldsymbol{\Lambda}\tilde{\boldsymbol{F}}^{\intercal}\boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}^{-\frac{1}{2}} \ , \\ \Delta_{\mathbf{f}_1\mathbf{f}_2} &= \boldsymbol{\Sigma}_{\mathbf{f}_1\mathbf{f}_1}^{-\frac{1}{2}}\tilde{\boldsymbol{F}}\tilde{\boldsymbol{G}}^{\intercal}\boldsymbol{\Sigma}_{\mathbf{f}_2\mathbf{f}_2}^{-\frac{1}{2}} \ , \end{aligned} \tag{2.38}$$

where $\tilde{\boldsymbol{F}}, \tilde{\boldsymbol{G}}$ are the SVD of $\boldsymbol{T}_{\mathbf{f}_1\mathbf{f}_1}$ as in the equation (2.36). And $\frac{\partial \sum_{j=1}^{N} \boldsymbol{\sigma}_j}{\partial \mathbf{f}_2(\boldsymbol{X}_2)}$ has an analogous expression [AABL13, WALS15].

In the DCCA algorithm, first start with the forward propagation through two DNNs $\mathbf{f}_1$ and $\mathbf{f}_2$, as shown in Algorithm 1. The weight matrices and bias vectors are given by $(\boldsymbol{W}_1^{l_1}, \boldsymbol{b}_1^{l_1})$, $l_1 = 1, 2, ..., d_1$ and $(\boldsymbol{W}_2^{l_2}, \boldsymbol{b}_2^{l_2})$, $l_2 = 1, 2, ..., d_2$.

Algorithm 1 DCCA - Forward computation through two DNNs

Input: Two views of data $\boldsymbol{X}_1 \in \mathbb{R}^{n\times N}$, $\boldsymbol{X}_2 \in \mathbb{R}^{n\times N}$. Initialization $\mathbf{f}_1$ are given by $(\boldsymbol{W}_1, \boldsymbol{b}_1)$ and $\mathbf{f}_2$ are given by $(\boldsymbol{W}_2, \boldsymbol{b}_2)$

$\boldsymbol{Z}_1^{(0)} = \boldsymbol{X}_1$, $\boldsymbol{Z}_2^{(0)} = \boldsymbol{X}_2$

$l_1 = 1$,$l_2 = 1$

for $l_1 = 1, 2, \cdots, d_1$, $l_2 = 1, 2, \cdots, d_2$ **do**

$\boldsymbol{Z}_1^{(l_1)} = s_1(\mathbf{W}_1^{(l_1)}\boldsymbol{Z}_1^{(l_1-1)} + \boldsymbol{b}_1^{(l_1)})$, $\boldsymbol{Z}_2^{(l_2)} = s_2(\mathbf{W}_2^{(l_2)}\boldsymbol{Z}_2^{(l_2-1)} + \boldsymbol{b}_2^{(l_2)})$

end for

Output: Outputs of two DNNs $\mathbf{f}_1(\boldsymbol{X}_1)$: $\hat{\boldsymbol{Y}}_1 = \boldsymbol{Z}_1^{(l_1)}$ and $\mathbf{f}_2(\boldsymbol{X}_2)$: $\hat{\boldsymbol{Y}}_2 = \boldsymbol{Z}_2^{(l_2)}$

Next it considers the back-propagation. The loss function needs to be a min function. The objective function can directly switch $\max(\circ)$ with $\min(-\circ)$. So (2.32) is rewritten as (2.39).

$$\begin{aligned} \min_{\mathbf{W}_{\mathbf{f}_1}, \mathbf{W}_{\mathbf{f}_2}, \boldsymbol{U}_1, \boldsymbol{U}_2} \quad & -\frac{1}{N}\operatorname{Tr}(\boldsymbol{U}_1^{\intercal}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_2(\boldsymbol{X}_2)^{\intercal}\boldsymbol{U}_2) \\ \text{s.t. } & \boldsymbol{U}_1^{\intercal}(\frac{1}{N}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_1(\boldsymbol{X}_1)^{\intercal} + r_1\boldsymbol{I})\boldsymbol{U}_1 = \boldsymbol{I}, \\ & \boldsymbol{U}_2^{\intercal}(\frac{1}{N}\mathbf{f}_2(\boldsymbol{X}_2)\mathbf{f}_2(\boldsymbol{X}_2)^{\intercal} + r_2\boldsymbol{I})\boldsymbol{U}_2 = \boldsymbol{I}, \\ & \boldsymbol{u}_1^{(i)\intercal}\mathbf{f}_1(\boldsymbol{X}_1)\mathbf{f}_2(\boldsymbol{X}_2)^{\intercal}\boldsymbol{u}_2^{(j)} = 0, \text{for } i \neq j. \end{aligned} \tag{2.39}$$

First propagate the gradient by appling the chain rule to compute the gradient of loss w.r.t. the pre-activation $\boldsymbol{Z}^{(l)}$ at the two outputs nonlinearity $\boldsymbol{g}^{(l)}(\cdot)$, next the gradient w.r.t. the weight matrix $\boldsymbol{W}^{(l)}$ and bias vector $\boldsymbol{b}^{(l)}$ of the last layer, then the gradient

w.r.t the pre-activation $\boldsymbol{Z}^{(l-1)}$ of the next to last layer, etc., until one arrives at the network input. In the two networks $(\mathbf{f}_1, \mathbf{f}_2)$, they follow the same computation rule with corresponding definitions. The computations are summarized in Algorithm 2. $(\hat{\boldsymbol{Y}}_1, \hat{\boldsymbol{Y}}_2)$ are the outputs of two DNNs $(\mathbf{f}_1, \mathbf{f}_2)$. $\boldsymbol{J}$ denotes the loss function (2.39). The gradient $\tilde{\mathbf{g}}_1$ and $\tilde{\mathbf{g}}_2$ are the derivatives of the loss function $\boldsymbol{J}$ w.r.t. a network layer output. It is initially set to the gradient of the loss function $\boldsymbol{J}$ w.r.t. the network output matrices $\hat{\boldsymbol{Y}}_1$ and $\hat{\boldsymbol{Y}}_2$, then overwritten by $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{Z}_1^{(l)}}$ and $\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{Z}_2^{(l)}}$ as traversing the layers.

Algorithm 2 DCCA - Backward computation through two DNNs

$\tilde{\mathbf{g}}_1 := \nabla_{\hat{\boldsymbol{Y}}_1} \boldsymbol{J}$, $\tilde{\mathbf{g}}_2 := \nabla_{\hat{\boldsymbol{Y}}_2} \boldsymbol{J}$

for $l_1 = d_1, d_1 - 1, \cdots, 1$, $l_2 = d_2, d_2 - 1, \cdots, 1$ **do**

Propagate gradient through the layer's nonlinearity to the pre-activation in each DNNs:

$\tilde{\mathbf{g}}_1 := \nabla_{\boldsymbol{Z}_1^{(l_1)}} \boldsymbol{J} = \tilde{\mathbf{g}}_1 \odot (\boldsymbol{g}_1^{(l_1)})'(\boldsymbol{Z}_1^{(l_1)})$, $\tilde{\mathbf{g}}_2 := \nabla_{\boldsymbol{Z}_2^{(l_2)}} \boldsymbol{J} = \tilde{\mathbf{g}}_2 \odot (\boldsymbol{g}_2^{(l_2)})'(\boldsymbol{Z}_2^{(l_2)})$

($\odot$ denotes element-wise multiplication)

Compute gradients on weights and biases:

$\nabla_{\boldsymbol{b}_1^{l_1}} \boldsymbol{J} = \tilde{\mathbf{g}}_1$, $\nabla_{\boldsymbol{W}_1^{l}} \boldsymbol{J} = \tilde{\mathbf{g}}_1 \cdot (\boldsymbol{Z}_1^{(l_1-1)})^\intercal$

$\nabla_{\boldsymbol{b}_2^{l_2}} \boldsymbol{J} = \tilde{\mathbf{g}}_2$, $\nabla_{\boldsymbol{W}_2^{l}} \boldsymbol{J} = \tilde{\mathbf{g}}_2 \cdot (\boldsymbol{Z}_2^{(l_2-1)})^\intercal$

Propagate gradient w.r.t. next lower-level hidden layer's output:

$\tilde{\mathbf{g}}_1 := \nabla_{\boldsymbol{Z}_1^{(l_1-1)}} \boldsymbol{J} = (\boldsymbol{W}_1^{(l_1)})^\intercal \cdot \tilde{\mathbf{g}}_1$

$\tilde{\mathbf{g}}_2 := \nabla_{\boldsymbol{Z}_2^{(l_2-1)}} \boldsymbol{J} = (\boldsymbol{W}_2^{(l_2)})^\intercal \cdot \tilde{\mathbf{g}}_2$

end for

2.3 Deep Canonically Correlated Autoencoders

Using DCCA can overcome the limitation of CCA. However, it leads to a new problem of overfitting, i.e., DCCA always maximizes the correlation between the latent representations close to 1 but the real correlation is unknown.

In order to sovling the problem, deep canonically correlated autoencoders(DCCAE) [WALB15a] takes use of the reconstruction to prevent this problem. If the new representations made by DCCA can be reconstructed back close to the original data, which the correlation between these new representations are more sensible. As shown in Fig. 2.5.

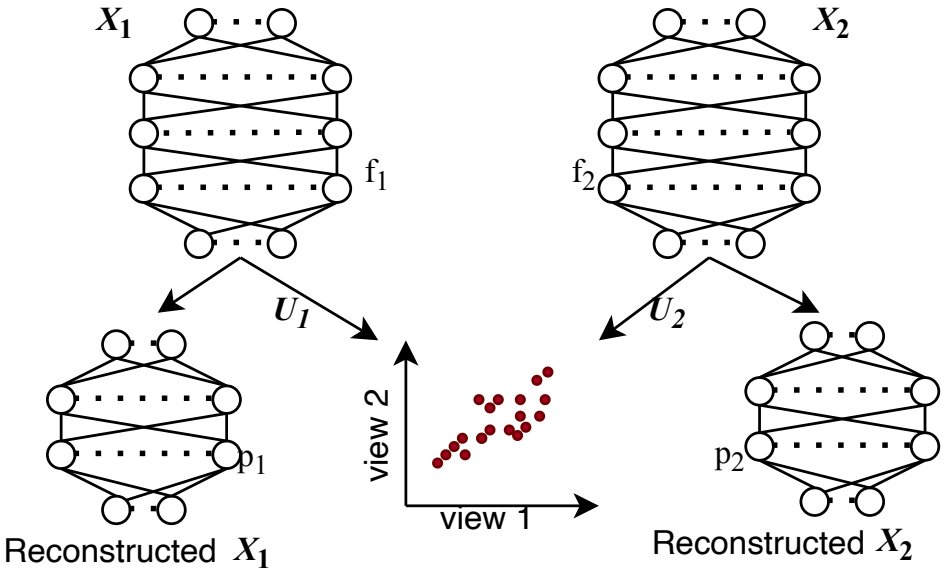

Figure 2.5: Schematic of Deep Canonically Correlated Autoencoders

Therefore, another two DNNs $\mathbf{p}_1$, $\mathbf{p}_2$ are built for reconstructing original data. With the reconstruction, any sample $\boldsymbol{u}_1 = \mathbf{f}_1(\boldsymbol{x}_1^{(1)})$ of output from $\mathbf{f}_1$ feeding into $\mathbf{p}_1$ gets the output $\boldsymbol{x}_1^{(1)'} = \mathbf{p}_1(\boldsymbol{u}_1)$. To minimize the reconstruction error between the original data sample $\boldsymbol{x}_1^{(1)}$ and the reconstructed data $\boldsymbol{x}_1^{(1)'}$ as shown in Fig. 2.6.

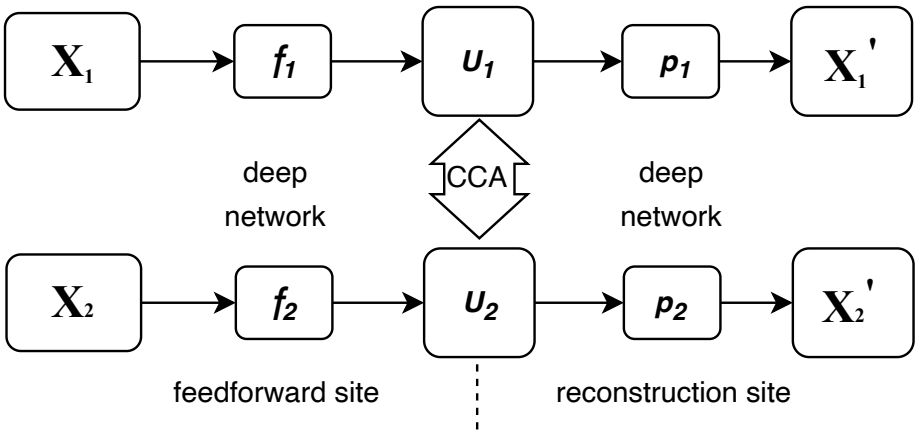

Figure 2.6: DCCAE Framework

It should apply to all the samples. Therefore, the obejctive of DCCAE can be defined as assuming there are two view data $\boldsymbol{X}_1 \in \mathbb{R}^{n \times N}$, $\boldsymbol{X}_2 \in \mathbb{R}^{n \times N}$. $\mathbf{f}_1$, $\mathbf{f}_2$ are the two DNNs of finding the maximal correlated outputs, and $\mathbf{p}_1$, $\mathbf{p}_2$ denote the two DNNs of reconstruction, with a corresponding set of learnable parameters, $\mathbf{W}_{\mathbf{f}_1}$, $\mathbf{W}_{\mathbf{f}_2}$, $\mathbf{W}_{\mathbf{p}_1}$ and

$\mathbf{W}_{\mathbf{p}_2}$.

$$\begin{aligned}\min_{\mathbf{W}_{\mathbf{f}_1},\mathbf{W}_{\mathbf{f}_2},\mathbf{W}_{\mathbf{p}_1},\mathbf{W}_{\mathbf{p}_2},\boldsymbol{U}_1,\boldsymbol{U}_2} \quad & -\frac{1}{N}\operatorname{Tr}(\boldsymbol{U}_1^\intercal \mathbf{f}_1(X_1)\mathbf{f}_2(X_2)^\intercal \boldsymbol{U}_2) \\ & +\frac{\lambda}{N}\sum_{i=1}^{N}(\|\boldsymbol{x}_{1i}-\mathbf{p}_1(\mathbf{f}_1(\boldsymbol{x}_{1i}))\|^2+\|\boldsymbol{x}_{2i}-\mathbf{p}_2(\mathbf{f}_2(\boldsymbol{x}_{2i}))\|^2) \\ \text{s.t. the same constraints in (2.32)} \ , & \end{aligned} \tag{2.40}$$

where $\lambda > 0$ is a trade-off parameter.

3 Implementation

In this thesis, the experimental model is based on convolutional neural networks [LBBH98a]. Previously the DCCAE method makes use of the feedforward neural network to overcome limitation of the linear projection. Since the experimental inputs are images, it has been proven that CNN is good at image processing. Therefore, a new model based on CNN with the DCCAE method should have a better performance on the multiview image data processing. Furthermore, the DCCA and DCCAE algorithms have a characteristic of promoting maximal correlation between the outputs approaching 1, despite the maximal correlation is not known, which leads to overfitting. To further evaluate the issue of overfitting in this proposed model, different visualization techniques are applied in this thesis. Using visualization techniques, the characteristics learned by the neural network model can be visualized, which can more intuitively analyze the correlation between the data and the overfitting problem.

3.1 CNN Based DCCAE Model

The model of this thesis consists of three parts: an encoder, an extracted layer, and a decoder. The encoder part first feeds forward the input data through a CNN, then uses CCA to maximize the correlation between two outputs of CNN. The outputs of CNN, which are maximal correlated representations, are in the extracted layer. Additionally, the decoder part is for reconstruction of the original data, which ensures that the two outputs in the extracted layer can well reflect the features of the original inputs. Fig. 3.1 shows the framework for this thesis model.

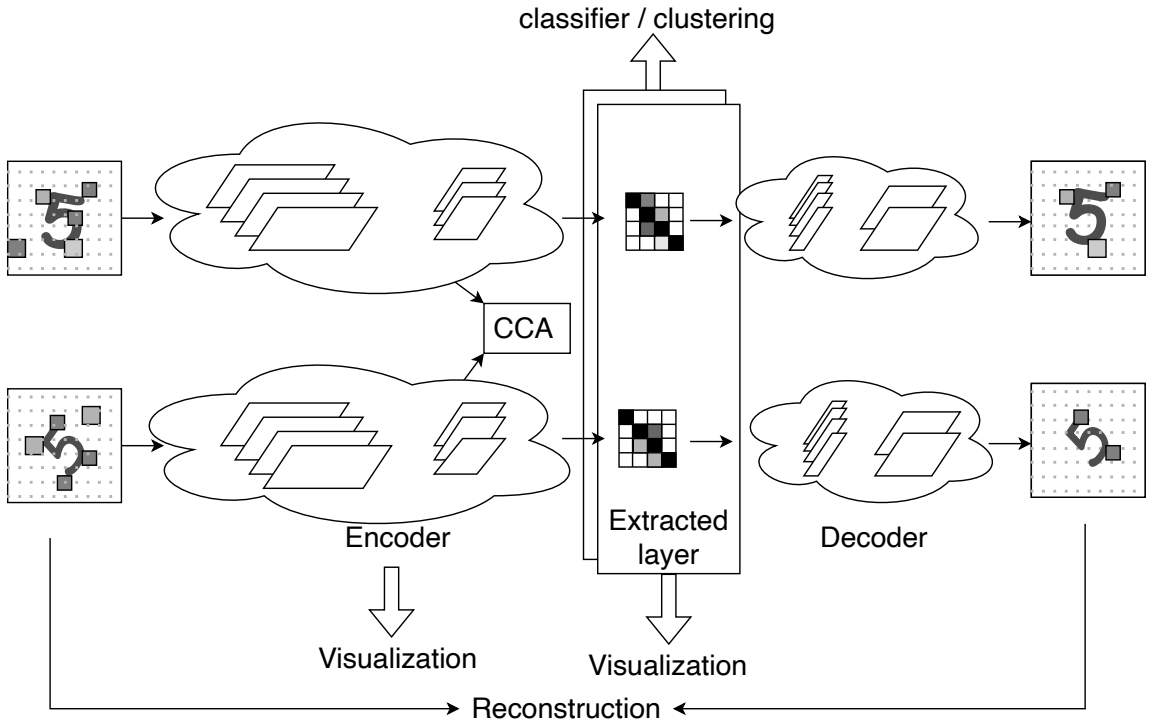

Figure 3.1: Framework of the Proposed Model (CNN based on DCCAE Model).

3.1.1 Data Generation

In the experiment, the input data uses MNIST dataset [LBBH98b]. To evaluate the correlation between two views of MNIST images, it also uses different corrupted MNIST data. Besides, the purpose of this model is to evaluate the correlation between two views by the DCCAE algorithm. Therefore for increasing more structural noise between inputs, box noise will be applied to the MNIST to obtain corrupted MNIST images.

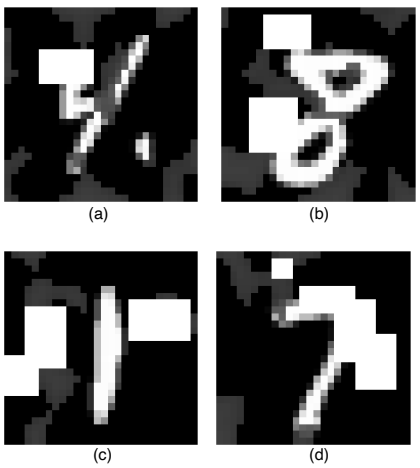

Figure 3.2: Box Nosie Corrupted MNIST by Spatter Samples. (a) Exmaple image with 1 box (b) Exmaple image with 2 boxes (c) Exmaple image with 3 boxes (d) Exmaple image with 4 boxes

Here are some examples of the box noised data, as in Fig. 3.2. It shows that the position and size of the box are random but the number of boxes is customized. Without box, it is called clean data in this experiment. Among clean data, the correlation found by the proposed model should lead to the numbers on the figures. In order to verify the algorithm of DCCAE, box increases the difficulty to distinguish the latent correlation by adding the structural noise between each view. The shared correlation learned by the proposed model should focus on the semantic pixels and not heavily influenced by box.

3.1.2 Convolutional Neural Network

In both DCCA and DCCAE, deep neural networks are used to process nonlinear transformation. The feedforward neural networks, which are composed of fully connected layers, are utilized as the deep neural network structure. However, it needs more parameters and does not consider the spatial information of input due to the characteristic of a fully connected layer.

Convolutional Neural Network was first proposed in 1998 [LBBH98a]. It has been widely applied to image processing and speech recognition. The features of CNN are shared parameters, less number of weights, and without losing spatial information. A simple convolutional neural network usually consists of convolutional layers, pooling layers, and a fully connected layer, as shown in Fig. 3.3.

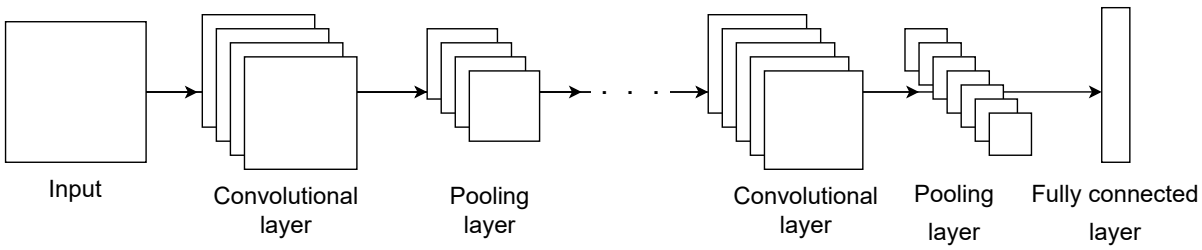

Figure 3.3: Simple Convolutional Neural Network Architecture

- Convolution operation. Convolution is a mathematical operation between two matrices (input and filter matrices) and its output is also a matrix. Hence in convolutional layers, if the input is an image the filter slides over the input image one pixel at a time starting from the top left. This filter multiplies its values with the overlapping values of the image while sliding over it and adds all of them up to output a single value for each overlap. This output matrix is the result matrix of the convolution operation, as explained in Fig. 3.4. The output of convolution can be formulated in eq(3.1).

$$\boldsymbol{S}(i,j) = \sum_m \sum_n \boldsymbol{I}(i+m, j+n)\boldsymbol{K}(m,n) \ , \tag{3.1}$$

  where $\boldsymbol{I}(i,j)$ is the input and the $\boldsymbol{K}(m,n)$ is the filter kernel, m,n is the size of kernel [Sha48].

- Padding and strides. When an input matrix is processed by the CNN kernel, padding is the number of pixels that are added to the matrix. In essence, padding adds a

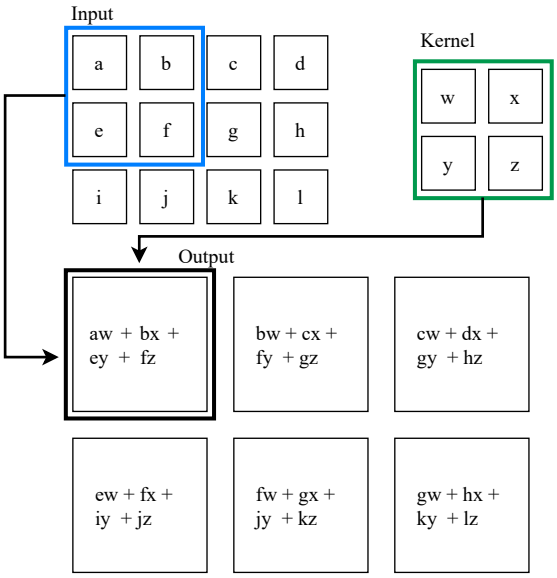

Figure 3.4: Illustration of a 2D Convolution [GBC16]

border to the input image to address two issues. It aids in output reduction but causes a loss of detail in the image's corners. To provide the kernel additional room to cover the image, padding can be applied to the image's frame. The number of pixels shifted across the input matrix is called the stride.

- Pooling layer. Its purpose is to gradually shrink the representation's spatial size to minimize the number of parameters and computations in the network. This helps maintain the process of efficiently training the model by extracting dominating characteristics that are rotational and positional invariant. Max Pooling and Average Pooling are the two classical operations of pooling. Max pooling returns the maximum value from the area that the kernel is responsible for. On the other hand, taking the average value from the area is average pooling. See examples in Fig. 3.5.

3.1.3 Classifier and Clustering

Classifiers [MP43] are commonly used in a variety of applications, such as image and speech recognition [GKP02], to automatically classify data into predefined categories. A simple classifier can be trained using a labeled dataset, where the input samples are associated with the correct class labels. In unsupervised learning, clustering methods [Bon64] are applied to reveal structure in data. The process of clustering involves grouping the data points into several groups so that the data points within each group are more similar to one another than the data points within other groups.

The experiment employs two different classification mechanisms, a classifier and a clustering, to provide additional dimensions for assessing the performance of the proposed model, as shown in the Fig. 3.1. As the experiment employs MNIST image data, the latent representations generated by the proposed model with the maximal correlation should be classified according to the corresponding labels by a classifier. Therefore,

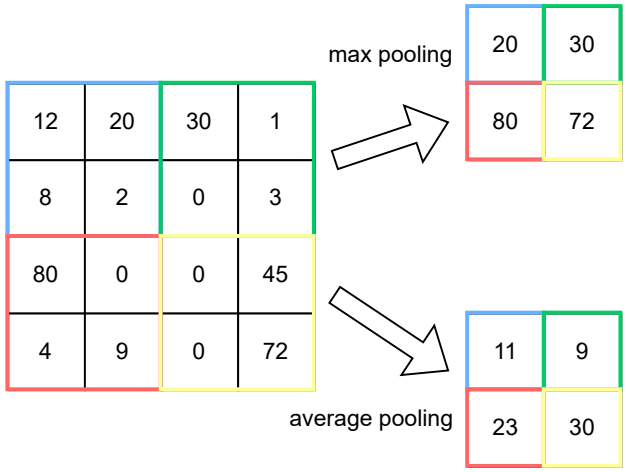

Figure 3.5: Explam of Max Pooling and Average Pooling

when the accuracy of the classifier is increasing with the model running, the correlation between two latent representations is more focused on the semantic pixels which can help to predict the correct labels. If the accuracy is high, the latent representations are good in comparison to low accuracy. Instead, when the accuracy starts to decrease, the model is possibly overfitting according to the feature of DCCA and DCCAE which tends to blindly maximize the correlation between two representations. It cannot accurately classify the latent representations into the corresponding labels. Similar to the classifier, here the latent representations would be fed into a clustering. The accuracy of the clustering can be calculated by using the labels to compare the predicted cluster assignments with the true cluster assignments. The two classification mechanisms as an external monitor in the proposed model, the accuracy of these mechanisms can be used to evaluate the performance of the model.

3.2 Visualization in Neural Networks

Visualization of neural networks can have a clear explanation of which characteristics of inputs the networks learn. Through this method, the tune of regularization parameters and learning parameters of neural networks can be more operational and the overfitting problem is easier to detect for evaluating the validity of the model. To provide interpretability, there are a few classical methods of visualization. Saliency Map(SM) [MKH+95] is the most basic method of visualizing networks. An improved ways of SM is SmoothGrad [STK+17]. SmoothGrad takes use of the technique that removing noise by adding noise to inputs. Besides, GradCAM [SCD+17] is good at visualizing convolutional neural networks.

3.2.1 Saliency Map

For the simple classifier, Saliency Map is attained by directly using the gradient of output backpropagating to the input.

Assume the function $\boldsymbol{S}_c$ of a simple classifier can use a linear function in the neighborhood of $\boldsymbol{I}$ by computing the first-order Taylor expansion to approximate

$$\boldsymbol{S}_c(\boldsymbol{I}) \approx \boldsymbol{\omega}^\intercal \boldsymbol{I}^{'} + \boldsymbol{b} \ ,$$

where the $\boldsymbol{I}$ denotes an image as the input by the matrix form, $\boldsymbol{I}^{'}$ represents the neighborhood of $\boldsymbol{I}$, and $\boldsymbol{\omega}, \boldsymbol{b}$ are the corresponding weight vector and bias of the classifier. The predicted classification is dedicated to the maximal value in the output of this classifier $\boldsymbol{S}_c(\boldsymbol{I})$.

$$\begin{aligned} class(\boldsymbol{I}) &= \underset{c}{\operatorname{argmax}}\, \boldsymbol{S}_c(\boldsymbol{I}) \\ &= \underset{c}{\operatorname{argmax}}(\boldsymbol{\omega}^\intercal \boldsymbol{I}^{'} + \boldsymbol{b}) \ . \end{aligned}$$

By this formula, it notices that the magnitude of elements of $\boldsymbol{\omega}$ is important to influence the output $\boldsymbol{S}_c(\boldsymbol{I})$, which the corresponding pixels $\boldsymbol{I}^{'}$ of $\boldsymbol{I}$ would represent the label information c.

Given a gray scalar image $\boldsymbol{I}_1 \in \mathbb{R}^{w \times h}$ and the corresponding class c_1, the output is $\boldsymbol{o}_1 \in \mathbb{R}^o$ which can be calculated by the classifier function.

$$\begin{aligned} \boldsymbol{o}_1 &= \boldsymbol{S}_c(\boldsymbol{I}_1) \\ &\approx \omega^\intercal \boldsymbol{I}_1' + \boldsymbol{b} \ , \end{aligned} \tag{3.2}$$

where $\boldsymbol{I}_1'$ is the neighbourhood of $\boldsymbol{I}_1$. To get the Saliency Map $\boldsymbol{M} \in \mathbb{R}^{w \times h}$, first it has to get the derivative $\boldsymbol{\omega}$ of $class(\boldsymbol{I}_1)$ with the respect to the image $\boldsymbol{I}_1$ by backpropagation

$$\begin{aligned} \boldsymbol{\omega} &= \frac{\partial class(\boldsymbol{I}_1)}{\partial \boldsymbol{I}_1} \\ &= \frac{\partial \operatorname{argmax}_c \boldsymbol{S}_c(\boldsymbol{I}_1)}{\partial \boldsymbol{I}_1} \ . \end{aligned} \tag{3.3}$$

The number of elements in $\boldsymbol{\omega}$ is equal to the number of pixels in $\boldsymbol{I}_1$. Therefore the Saliency Map $\boldsymbol{M}$ can be taken directly from the absolute value of the $\boldsymbol{\omega}$ with corresponding pixels.

$$\boldsymbol{M}(i,j) = |\boldsymbol{\omega}_{p(i,j)}| \ , \tag{3.4}$$

where the $p(i,j)$ denotes the index of the element of $\boldsymbol{\omega}$ corresponding to the image $\boldsymbol{I}_1$ in the i-th row and j-th column. If the input $\boldsymbol{I}_1$ is a multi-channel image, e.g. **RGB** picture. The Saliency Map $\boldsymbol{M}$ can take the maximal value of the channels h to be obtained.

$$\boldsymbol{M}(i,j) = \max_h |\boldsymbol{\omega}_{p(i,j,h)}| \ , \tag{3.5}$$

where the $p(i,j,h)$ represents the index of the pixel in the (i,j) position and the c color channel of image $\boldsymbol{I}_1$.

3.2.2 SmoothGrad

Saliency Map highlights the important pixels to display what the classifier learned from the input. For example, in MNIST digital images the pixels of the numbers' position is outstanding. However, the sensitive map also detects the noise from the learning process. The example of a Saliency Map for visualizing the classifier also highlights the noise surrounding the numbers' position. When some more complicated models try to use a Saliency Map to visualize, the noise probably has a heavy influence on the displaced results. In the meantime, it does not indicate the noise can not reflect what the model learns from the input. While the visualization with less noise offers research value especially for sensitive models.

From the research [Bis95], it is a common regularization that adding noise in the training data has a de-noise effect on sensitivity maps. Therefore, SmoothGrad offers a de-noised Saliency Map $\tilde{\boldsymbol{I}}$ by adding noise,

$$\tilde{\boldsymbol{I}} = \frac{1}{n}\sum_{1}^{n}(\hat{\boldsymbol{I}}_1 + \cdots + \hat{\boldsymbol{I}}_n) \ , \tag{3.6}$$

where $\hat{\boldsymbol{I}}_n$ denotes the noised $\boldsymbol{I}$ for $n = 1, 2, ..., n$.

Similar to getting a Saliency Map, given a gray scalar image $\boldsymbol{I}_1 \in \mathbb{R}^{w\times h}$ and the corresponding class c_1, the output is $\boldsymbol{o}_1 \in \mathbb{R}^o$ which can be calculated by the classifier function (3.2). Instead of using the gradient $\boldsymbol{\omega}$ in (3.3), it uses the smoothing gradient with a Gaussian kernel. It is hard to directly compute the local average in a high-dimension input. Therefore a straightforward stochastic approximation can be used here. Using n random samples in a neighborhood of an input $\boldsymbol{I}_1$

$$\begin{aligned}
\hat{\boldsymbol{I}}_1 &= \boldsymbol{I}_1 + \mathcal{N}(0, \sigma^2) \ , \\
\hat{\boldsymbol{I}}_2 &= \boldsymbol{I}_1 + \mathcal{N}(0, \sigma^2) \ , \\
&\vdots \\
\hat{\boldsymbol{I}}_n &= \boldsymbol{I}_1 + \mathcal{N}(0, \sigma^2) \ ,
\end{aligned} \tag{3.7}$$

where $\mathcal{N}(0, \sigma^2)$ represents Gaussian noise with standard deviation σ. Then using each noised input can calculate the corresponding noised gradient $\hat{\boldsymbol{\omega}}_1, \hat{\boldsymbol{\omega}}_2, ..., \hat{\boldsymbol{\omega}}_n$ by backpropagation.

$$\hat{\boldsymbol{\omega}}_n = \frac{\partial class(\hat{\boldsymbol{I}}_n)}{\partial \hat{\boldsymbol{I}}_n} = \frac{\partial \operatorname{argmax}_c \boldsymbol{S}_c(\hat{\boldsymbol{I}}_n)}{\partial \hat{\boldsymbol{I}}_n} \ . \tag{3.8}$$

Taking the de-noised thinking in (3.6) to calculate the smooth gradient of input $\boldsymbol{I}_1$.

$$\tilde{\boldsymbol{\omega}} = \frac{1}{n}\sum_{1}^{n}(\hat{\boldsymbol{\omega}}_1 + \hat{\boldsymbol{\omega}}_2 + \cdots + \hat{\boldsymbol{\omega}}_n) \ . \tag{3.9}$$

Same to the Saliency Map, the SmoothGrad map can be computed.

$$\tilde{\boldsymbol{M}}(i, j) = |\hat{\boldsymbol{\omega}}_{p(i,j)}| \ . \tag{3.10}$$

Similarly, if the input is multi-channel image,

$$\tilde{\boldsymbol{M}}(i, j) = \max_h |\hat{\boldsymbol{\omega}}_{p(i,j,h)}| \ . \tag{3.11}$$

3.2.3 GradCAM

GradCAM [STK+17] is the abbreviation of Gradient-weighted Class Activation Mapping which was first proposed in 2017. It is a CNN-based visualization method. GradCAM can be applied to the CNN model families, for example, CNN with fully connected layers model. It focuses on the gradient of the target which predicts the label information to highlight the important pixels. Therefore GradCAM can be an assessment element to evaluate some models' performance especially when they have similar accuracy.

Deeper representations in a CNN are said to capture higher-level visual constructs, according to several previous studies [BCV13, MV16]. Besides, the structure of CNN determines it can retain spatial information which would be lost in a fully connected layers model. The last convolution layers should be the key because they can compromise between the high-level semantic and the spatial information. The neurons in these layers can tell the class label information which looks for the high-level semantics. GradCAM takes use of the gradients backpropagating into the last convolution layers to assign the key values which strongly relate the class label information to each neuron.

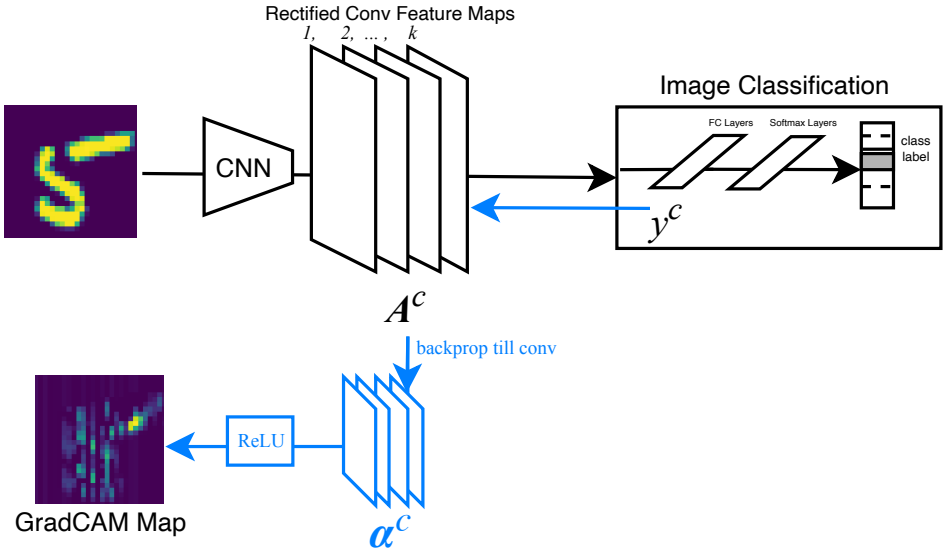

Figure 3.6: Schematic of GradCAM in a Image Classification

In order to get the GradCAM map $\boldsymbol{L}^c_{\text{GradCAM}} \in \mathbb{R}^{u \times v}$ of width u and height v for any class c as shown in Fig. 3.6, it first has to compute the gradient of the score for class c, $\boldsymbol{y}^c$(before the softmax), with respect to feature map activation $\boldsymbol{A}^k$ of a convolutional layer, i.e. $\frac{\partial \boldsymbol{y}^c}{\partial \boldsymbol{A}^k}$. These gradients backpropagating into the convolution layers are global-average-pooled over the width and height dimensions to obtain the neuron importance weights α^c_k.

$$\alpha^c_k = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial \boldsymbol{y}^c}{\partial A^k_{ij}}}_{\text{gradient via backprop}}, \tag{3.12}$$

where A^k_{ij} represents the element value in $\boldsymbol{A}^k$ with corresponding pixel index (i, j) and Z is the sum of all pixels. The gradients with respect to activation would also backpropagate

through all layers during calculation of α_k^c. Up until the final convolution layer that the gradients are being transmitted to, the exact computation consists of successive matrix products of the weight matrices and the gradient with respect to activation functions. Thus, this neuron weights α_k^c catches the key parts of feature map k for a target class c and indicates a partial linearization of the deep network downstream from $\boldsymbol{A}$. Then GradCAM map $\boldsymbol{L}^c_{\text{GradCAM}}$ can be a weighted combination of forward activation maps then be the answers after through a ReLU. The reason why it needs a ReLU to get the GradCAM values is that the useful pixels should have a positive influence on the class label information.

$$\boldsymbol{L}^c_{\text{GradCAM}} = \text{ReLU} \underbrace{\left(\sum_k \alpha_k^c \boldsymbol{A}^k \right)}_{\text{linear combination}} . \tag{3.13}$$

Notice that GradCAM can not only be applied to visual classification CNN like in Fig. 3.6 but also other CNN model, e.g. image captioning and visual question answering, by different definitions $\boldsymbol{y}^c$.

3.2.4 Visualization Classification Example using Clean MNIST

Before taking use of these different visualization techniques into DCCAE based CNN model, it is necessary that evaluating the validity of these techniques and better understanding how they work and their maps look like. In order to achieve this purpose, the experiment applied these visualization methods into two image classifiers with different structures of neural network. Moreover, the input uses the MNIST dataset to make it simpler. The fully connected deep network is on the classifier A's focus, the structure is shown in Table. 3.1. The other classifier B uses the convolutional neural network, shown in Table. 3.2. The accuracy of these two image classifiers is approximatly equal to 96%.

Layers	Output Shape
Input	(None, 784)
Layer 1(Dense)	(None, 1024)
Layer 2(Dense)	(None, 1024)
Output layer(Dense)	(None, 10)

Table 3.1: The Fully Connected Network Classifier A

First test Salieny Map on both classifiers, the results are shown in Fig. 3.7, 3.8. As shown in the figures, Saliency Map can detect some important pixels which lead to the prediction in the neural networks. Notice that based on classifier A, it performs well when the input image has a simple structure like 1, and 7 and also a special structure like 3. While based on classifier B, more spatial information, e.g. background pixels, are detected by Saliency Map. Although from the results of the Saliency Map it seems that the classifier with fully connected layers learns better than classifier B, there are a lot of reasons to answer this observation. For instance, the input MNIST data has a too simple structure.

Layers	Output Shape
Input	(None, 28,28,1)
Layer 1(Conv2D)	(None, 28,28,8)
Layer 2(MaxPooling2D)	(None, 14,14,8)
Layer 3(Conv2D)	(None, 14,14,4)
Layer 4(MaxPooling2D)	(None, 7,7,4)
Layer 5((Flatten))	(None, 196)
Output layer(Dense)	(None, 10)

Table 3.2: The Convolutional Neural Network Classifier B

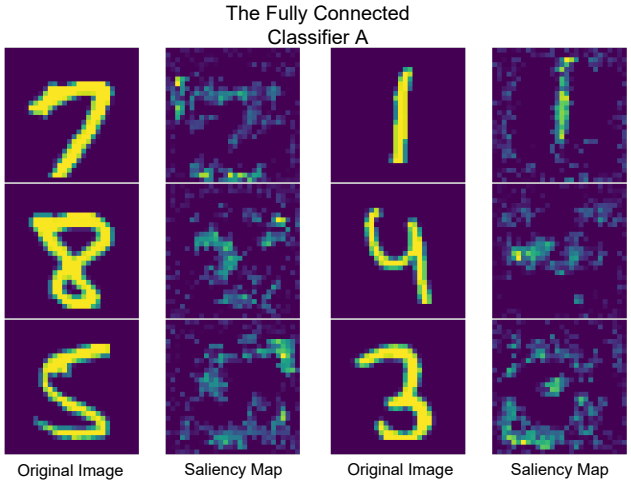

Figure 3.7: SM bases on Classifier A. First and third columns are the original MNIST images. Second and fourth columns are the first and third one corresponding SM maps.

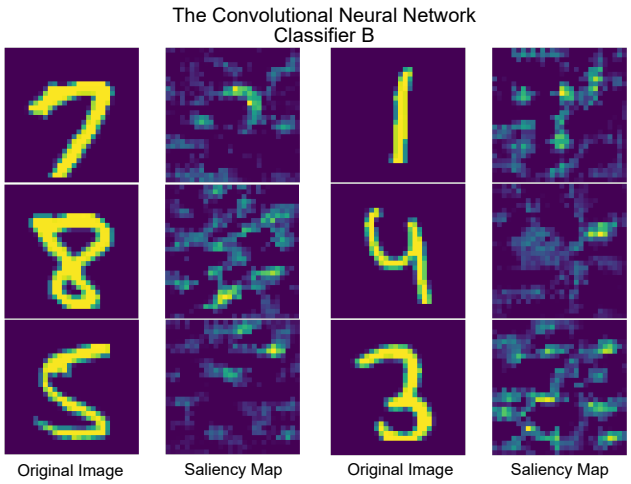

Figure 3.8: SM bases on Classifier B. First and third columns are the original MNIST images. Second and fourth columns are the first and third one corresponding SM maps.

Next test SmoothGrad on these two models, the results are shown in Fig. 3.9,3.10.

In the two comparisons, the experiment took different σ to make normal noise on the original images. With two different σ settings which are 0.2 and 0.4, it shows that the pixels learned by the classifier A are lighter when the noise is stronger. This proves that appropriately increasing the noise of the appropriate intensity can offer clearer learning maps. However, in classifier B SmoothGrad with a smaller noise setting is shown better than it with stronger noise.

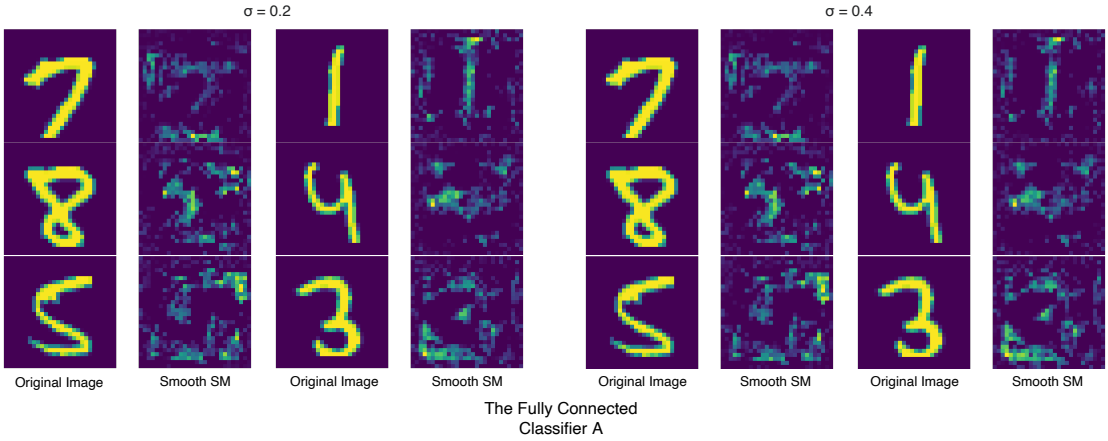

Figure 3.9: SmoothGrad based on Classifier A

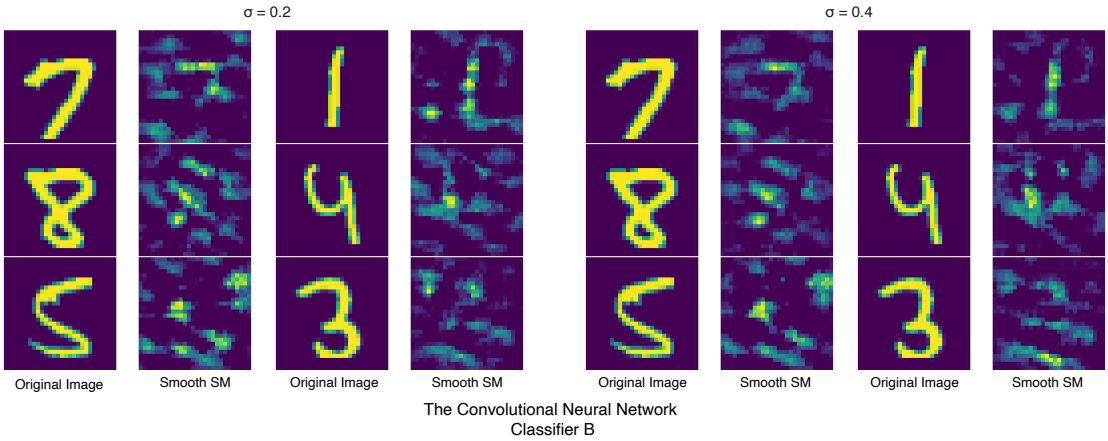

Figure 3.10: SmoothGrad based on Classifier B

Moreover, the SmoothGrad offers a clearer visualization map comparing the results of two different visualization techniques based on two classifiers. More details are showing what these classifiers learn from inputs.

The last visualization method requires a neural network model using convolution layers. Hence it is only applied to visualize classifier B. In Fig. 3.11 there are two similar results. The above one is to use the first convolution layer while the below one uses the second convolution layer, though in the principle GradCAM only takes use of the last convolution layer. Classifier B is not a too complex model in which there are only two convolution layers and five layers in total. The results in Fig. 3.11 prove that they are not

too much different in which convolution layer to apply. Moreover, GradCAM is assessed as having the best performance among the three visualization methods.

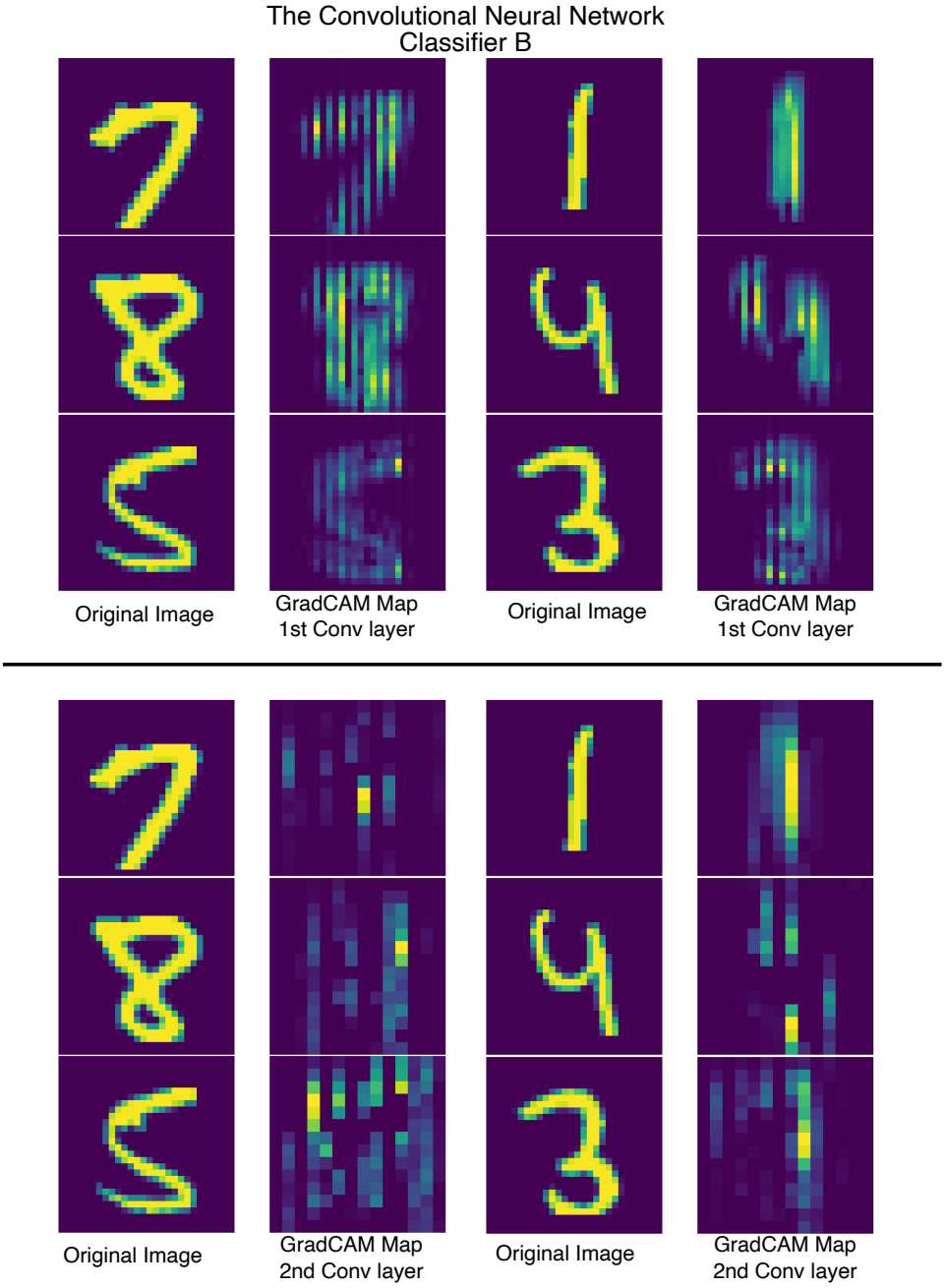

Figure 3.11: GradCAM based on Classifier B

4 Experiments and Results

The goal of experiments is to analyse which neural network structure works better with the same input data and making use of visualization techniques to understand how the network works in the process of maximizing correlated latent representations. In this thesis, the employed prgoramming language is python.

The experiments and results are classified into three groups. Section 4.1 shows the different performances between feedforward neural network and convolutional neural network based DCCAE model. Hyper-parameter settings are summarized in section 4.2. The last section 4.3 shows the results of visualizations and analysis of overfitting.

4.1 Comparison between Two Network Structures

In the experiment, the input data is clean MNIST data with one box. Examples are shown in Fig. 4.1. The deep neural network in the DCCAE model (DNN model) uses the fully connected layers, as shown in Table. 4.1. The proposed model uses the convolutional neural network (CNN model), more specific settings are given in Table. 4.2.

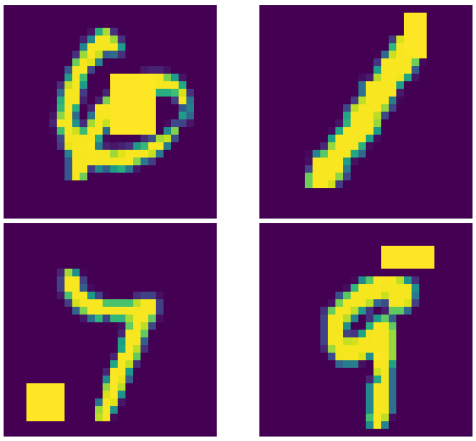

Figure 4.1: Examples of clean MNIST with 1 box

Layers	Output Shape	Activation function
Input	(None, 784)	
Layer 1(Dense)	(None, 1024)	ReLU
Layer 2(Dense)	(None, 1024)	ReLU
Layer 3(Dense)	(None, 1024)	ReLU
Output layer(Dense)	(None, 10)	None

Table 4.1: The Deep Neural Network(DNN)

Layers	Output Shape	Filters	Kernel / Pooling	Stride	Padding	Activation function
Input	(None, 28,28,1)					
Layer 1(Conv2D)	(None, 28,28,8)	8	(5,5)	(1,1)	0	ReLU
Layer 2(MaxPooling2D)	(None, 14,14,8)		(2,2)	(2,2)	no padding	
Layer 3(Conv2D)	(None, 14,14,6)	6	(5,5)	(1,1)	0	ReLU
Layer 4(MaxPooling2D)	(None, 7,7,6)		(2,2)	(2,2)	no padding	
Layer 5(Conv2D)	(None, 7,7,4)	4	(5,5)	(1,1)	0	ReLU
Layer 6(MaxPooling2D)	(None, 3,3,4)		(2,2)	(2,2)	no padding	
Layer 7(Conv2D)	(None, 3,3,4)	2	(5,5)	(1,1)	0	ReLU
Output layer((Flatten))	(None, 18)					

Table 4.2: The Convolutional Neural Network(CNN)

Besides, the latent representation which is projected by the view 1 is the input for the classifier, which classifies the latent representation into the corresponding label of view 1. The classifier uses a support vector machine(SVM) [Vap95] to analyse the performance.

First, the correlations between the two outputs of the two models are shown in Fig. 4.2. The maximal accuracy of SVM in DNN model is 94.4% at 150th epochs, while the maximal accuracy of SVM in CNN model is 93.3% at 1000th epochs. From this figure, it is clear that the DNN model always drives the correlation between two latent representations close to 1 while the true correlation of the original data is likely not equal to 1. With

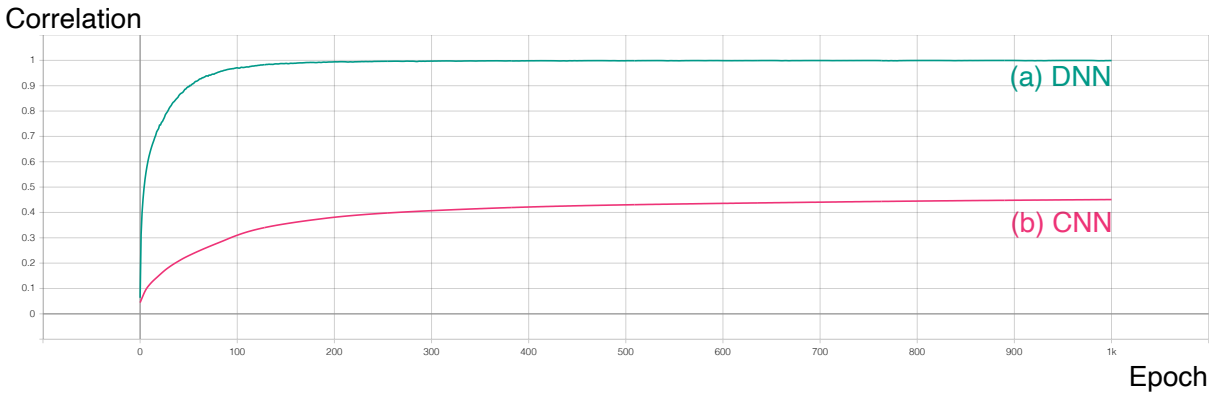

Figure 4.2: Correlation of DNN Model and CNN Model. (a) DNN model, (b) CNN model

the epoch running, the correlation derived by the CNN model is kept under 0.5 which is more reasonable according to the examples of input data. Moreover, the change of SVM accuracy of the two models is shown in Fig. 4.3. It indicates that the accuracy in the DNN model starts to decrease before the accuracy in the CNN model. The accuracy in the CNN model possibly still increases with more epoch running. It explains that the latent representations generated by the CNN model more concentrate on the semantics of the digits, while the outputs from the DNN model probably are disturbed by the box.

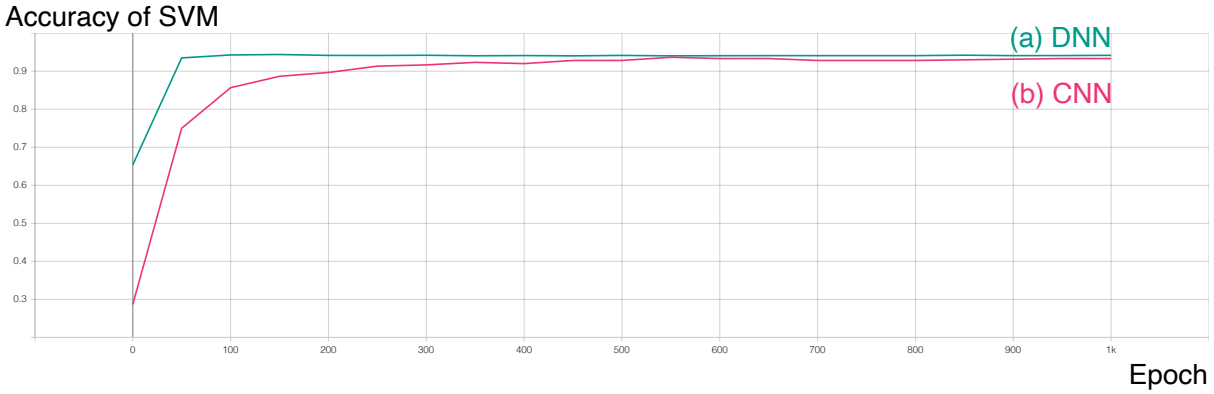

Figure 4.3: Accurayc of SVM in DNN Model and CNN Model using View 1. (a) DNN model, (b) CNN model

4.2 Hyper-Parameter Setting

In order to find the optimum of the two models and have a better performance to evaluate the models, L1-norm and L2-norm as the hyper-parameter settings are applied in this experiment. Besides, the input data is clean MNIST data with one box in order to have less uncertain noise. Examples are shown in Fig. 4.1. The regularizations (r_1, r_2) in CCA loss function (2.28) are set to 0.0001.

4.2.1 L1-norm

Fig. 4.4 shows the effect of four different choices of L1-norm in the DNN model. Despite these variations, the correlation between the two outputs remains relatively consistent due to the characteristics of the DNN model. The accuracy of SVM in the DNN model provides more information, as the rate of increase towards the maximal values differs depending on the L1-norm setting, as seen in Fig. 4.5.

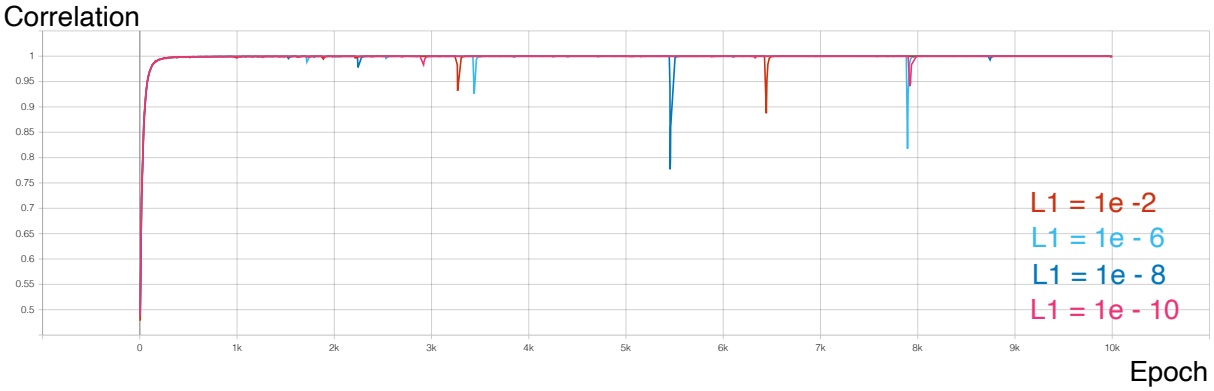

Figure 4.4: Correlation with different L1-norm setting in the DNN Model

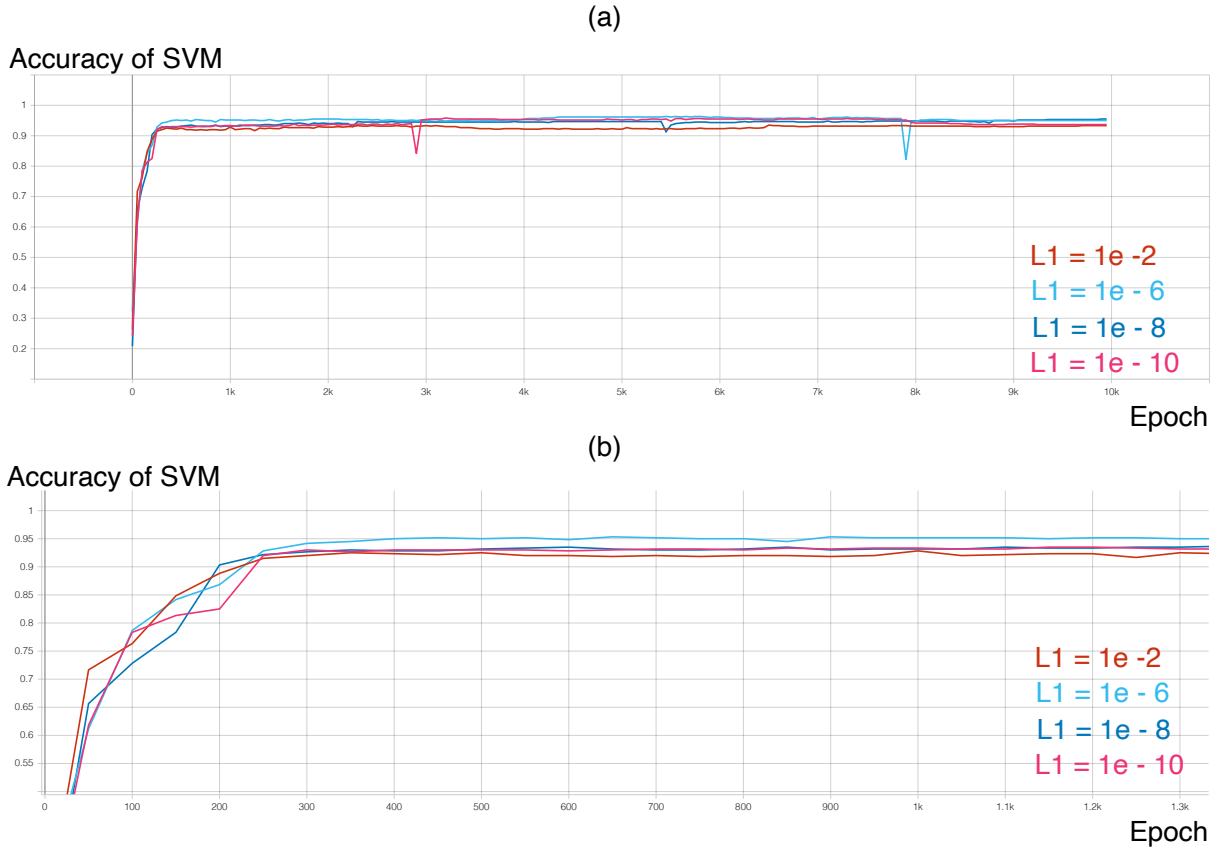

Figure 4.5: Accuracy of SVM in the DNN Model (a)Accuracy of SVM with different L1-norm setting using View 1, (b)Zoomed version of (a)

Four different examples of L1-norm in the CNN model are shown in Fig. 4.6. With different setting, the increase of the correlation between outputs of the CNN model is much different. In the showing case, when $L1 = 1e - 10$ the correlation grows fastest. If we look closer at the change of accuracy of SVM in the CNN model in Fig. 4.7(b), it shows a similar result with accuracy in the DNN model.

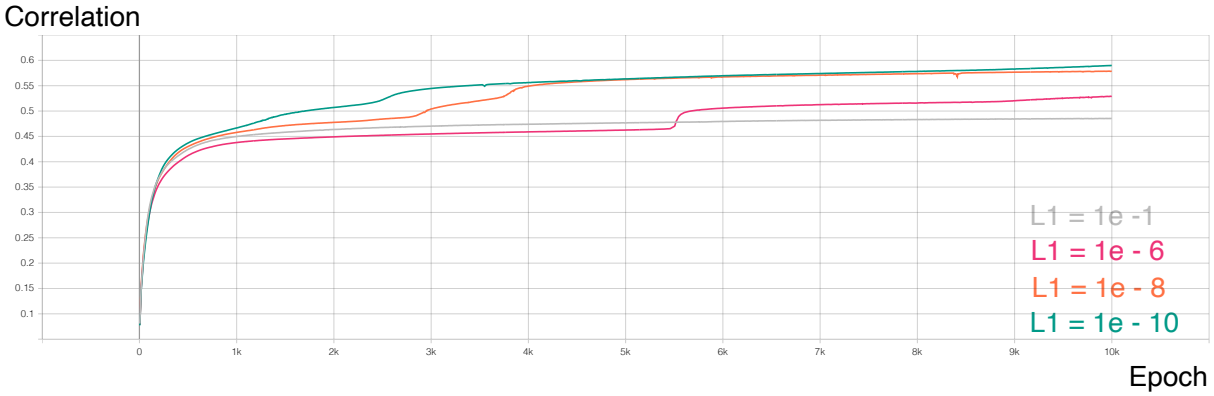

Figure 4.6: Correlation with different L1-norm setting in the CNN Model

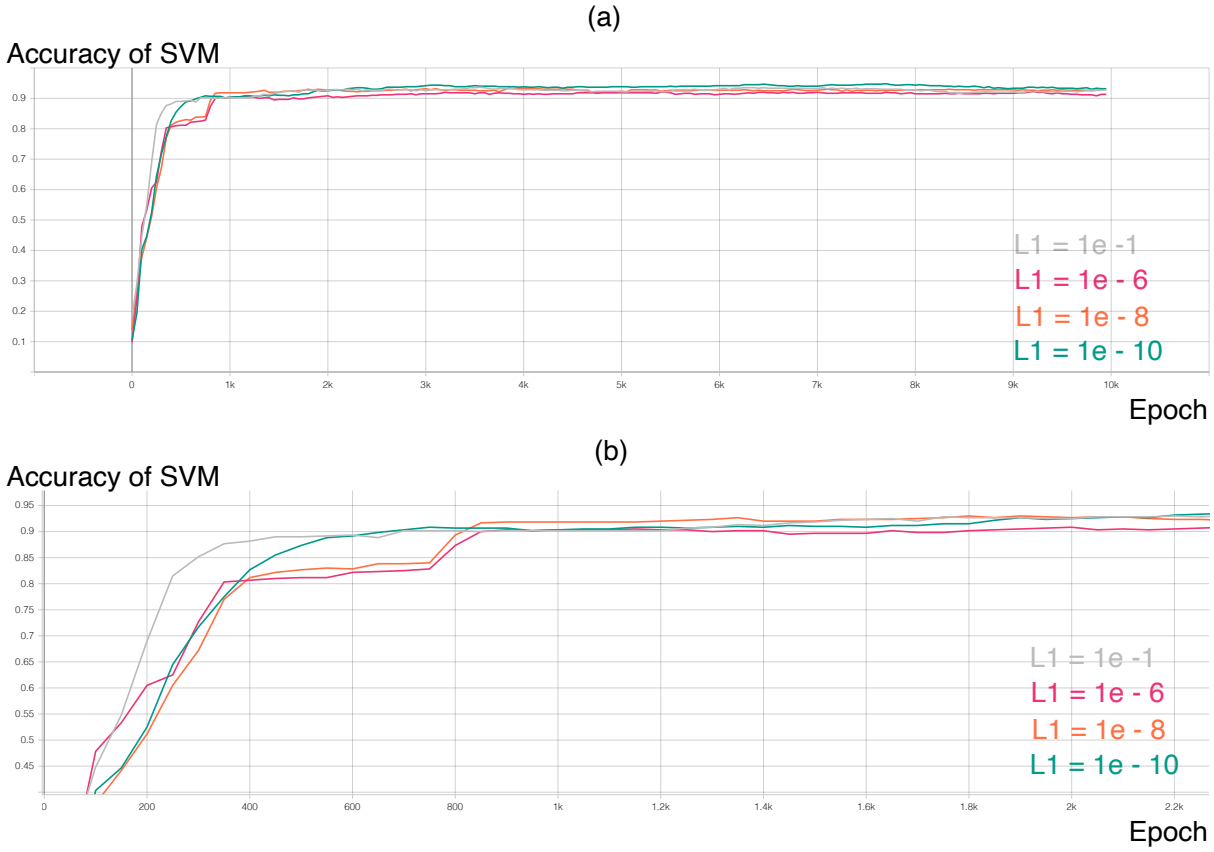

Figure 4.7: Accuracy of SVM in the CNN Model (a)Accuracy of SVM with different L1-norm setting using View 1, (b)Zoomed version of (a)

4.2.2 L2-norm

The performances of different L2-norm settings are similar with performances of different L1-norm settings in the DNN model. It does not display too much difference. Also accuracy in DNN model expresses the similarity with L1-norm experiment that before it reaches the top value the slope of each scalar is different. In the CNN model, there is not much different observed with different settings of L2-norm. Similarly, with different choices the increase in the maximal correlation is different. Additionally, the results of accuracy of SVM in the CNN model shows a similar result with accuracy in the DNN model in Fig. 4.8.

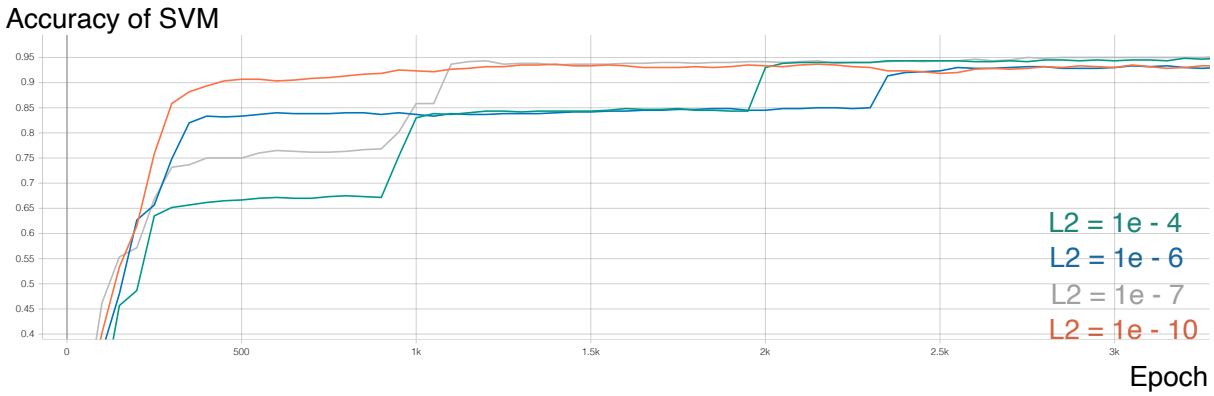

Figure 4.8: Accuracy of L2-norm in CNN Model using View 1

L1-norm and L2-norm regularization are techniques used to prevent overfitting in neural network models by adding a penalty to the loss function during training. However, in this experiment, these regularization did not have a significant effect on model performance. The optimal level of regularization may depend on the specific features and patterns present in the dataset or the architecture of the model. In conclusion, L1-norm and L2-norm regularization do not significantly impact the performance of DNN and CNN model.

4.3 Visualization Results

Due to the fact that the proposed model is an unsupervised learning model, there is no label in real applicaitions to assist in knowing when to stop overfitting or to determine the best performance of the model. Visualization can provide insights into the performance of the model.

As introduced in Saliency Map 3.2.1 and SmoothGrad 3.2.2, it needs a classifier to backpropagate the max value to the input. Thus a simple classifier is applied in the DNN model and the CNN model. In order to minimize the role of the classifier in the end accuracy and to focus on the role of latent representation on the accuracy, the structure of the simple classifier is one fully connected layer and a softmax layer. The maximal value of the output of the classifier is first backpropagated to the output of the DNN or CNN model. The backpropagation then continues to the input data, as illustrated in Fig. 4.9. The Gaussian noise in SmoothGrad is $\mathcal{N}(0, 0.4^2)$.

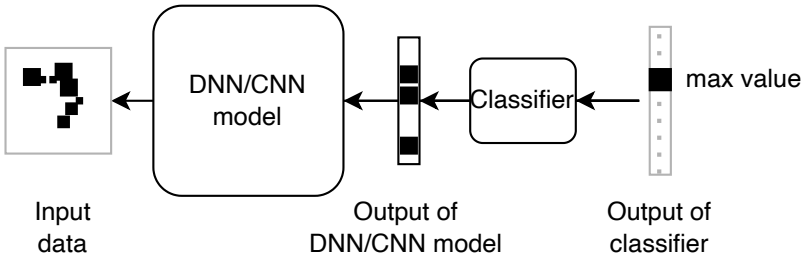

Figure 4.9: Saliency Map and SmoothGrad Process Diagram in the DNN/CNN model

Based on the principle of GradCAM [SCD+17] that can visualize convolutional neural networks, it is only applied in the CNN model. Besides, after the comparison experiments in the previous section, the comparable optimal hyper-parameters setting of $L2 = 1e-6$ is used in this visualization experiment. Moreover, to ensure the quality of experimental results and support the evaluation of subsequent experiments, t-SNE [VdMH08] and the clustering using a k-means clustering [JD88] are applied to visualize correlated latent representations and to have more monitoring effect on the performance of models.

4.3.1 Visualization Evaluation

In this experiment between the DNN model and the CNN model, the input data is clean MNIST data with one box. In the DNN model, the correlation between outputs rapidly increases to 1 around at 200th epochs. The accuracy of the classifier increases to the maximum around at 150th epochs. Afterwards, it keeps to the maximum with more epochs running. While the accuracy of the clustering first increases to maximum at 250th epoch then starts to decrease with more epochs running. Thus, the visualization results at the maximal accuracy of the clustering (250th epochs) and first minimal accuracy (900th epochs) are shown in Fig. 4.10. The first row images are generated by Saliency Map and the results of SmoothGrad are the second row. Before the DNN model processes, the visualized map should look similar with the corresponding original image but much blurry by random noise. At 0 epoch, both visualized maps look like the original images with surrounding noise. When at 250th epochs, the noise is less than before and both visualized maps are more focused on the semantic positions, letter 1. The difference between the two visualization techniques is not that big. At the decreased accuracy time (900th epochs) the two visualized maps are messy by noise. However, it does not necessarily imply that the DNN model is not learning effectively from the input data at 900th epochs. t-SNE results shown in Fig. 4.11 indicates at 900th epochs the output data distribution is still much better than input data distribution which is fit with the fact the accuracy of the clustering is good. From this example image, it explains that the DNN model at 250th epochs learns more from the semantic correlation to generate the outputs which lead to the best performance of the clustering.

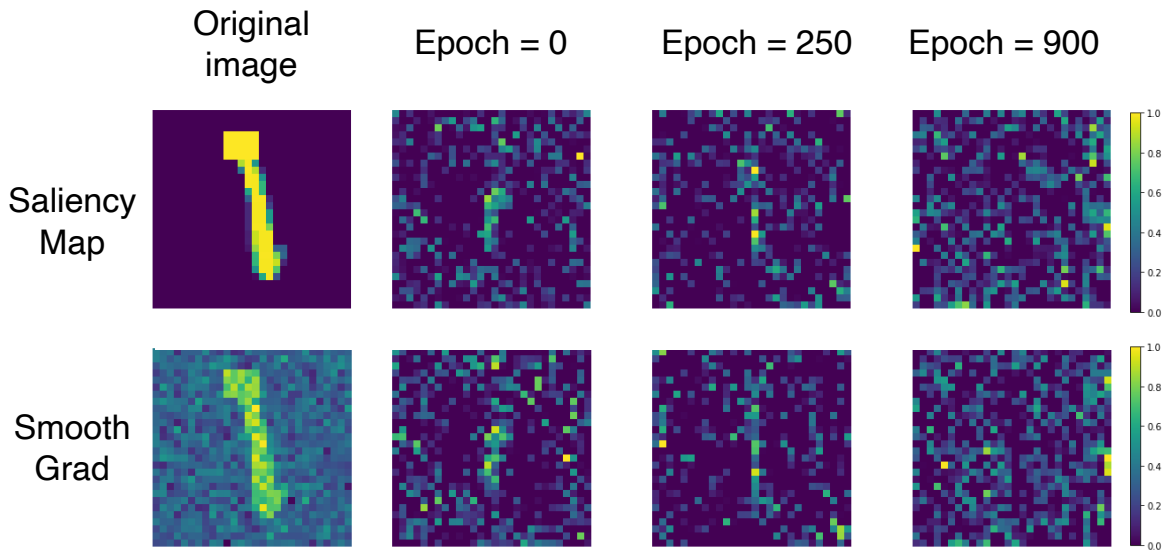

Figure 4.10: Visualization Results in the DNN model

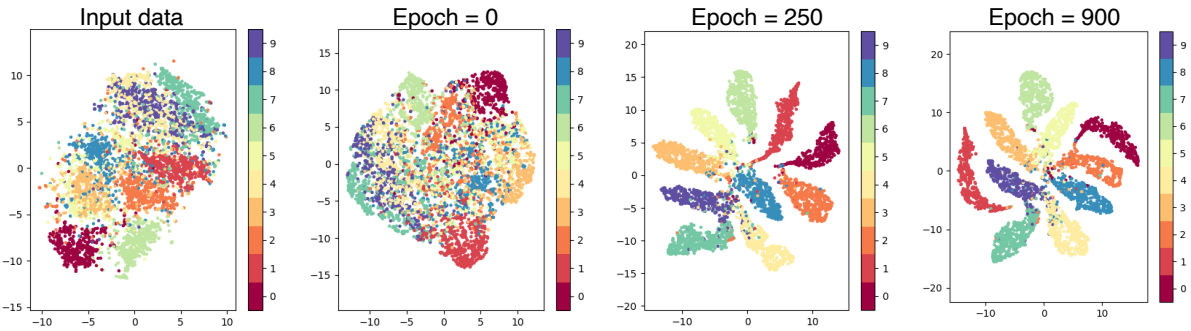

Figure 4.11: t-SNE in the DNN model

In conclusion, the DNN model learns more from the semantic area of the input before the clustering starts to decrease. Saliency Map and SmoothGrad can be used to enhance the interpretability of the DNN model. While SmoothGrad does not outperform Saliency Map in this experiment.

In the CNN model, the correlation between outputs is much lower than in the DNN model. It still keeps around 0.4 with more epochs running. While in the DNN model correlation is almost equal to 1 after 200 epochs. The accuracy of the classifier and the clustering in the CNN model keep increasing. The most difference with the DNN model is that there is no significant decrease in accuracy of the clustering. Then the visualization results in the CNN model are shown in Fig. 4.12. It displays the time of maximal accuracy of the clustering (450th epochs) and the first minimal accuracy of the clustering (950th epochs). The first line is the Saliency Map results. The second one is the SmoothGrad results. GradCAM results are in the third line. It is obvious that in the CNN model Saliency Map and SmoothGrad do not work as well as in the DNN

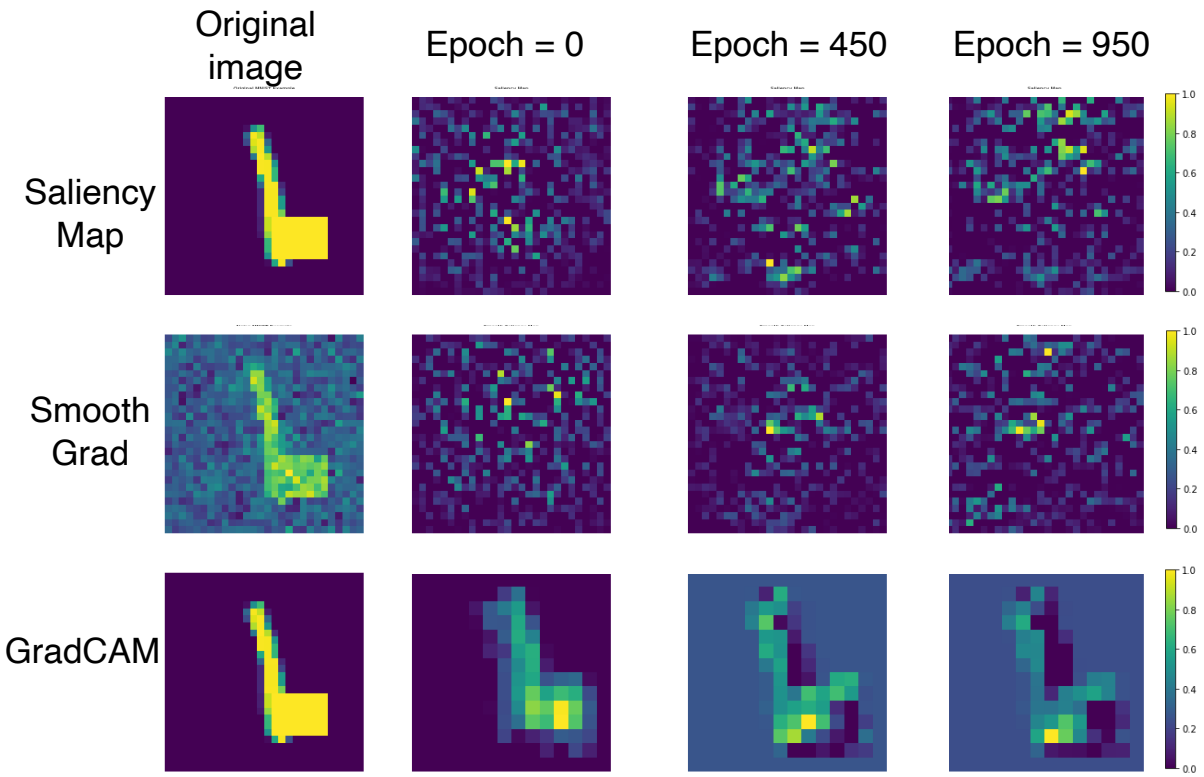

Figure 4.12: Visualization Results in the CNN model

model. While GradCAM has a much clear result than the other two methods. Before the CNN model processes, GradCAM just shows a blurry version of the original image. When the CNN model learns more about the correlation from the input data, GradCAM shows more focus on the semantic position. Whereas, here is an example in which the box is on the digital letter, so even when from the GradCAM at 450th epochs which is the best accuracy of the clustering, it still shows a comparatively high correlation on the box. However, compared with the GradCAM at 950th epochs the semantic area (digit 1) is less strong than its at 450th epochs but the box position is almost as strong as its at 450th epochs. Moreover, the results of t-SNE in Fig. 4.13 supports the resluts of the visualization, which shows that when at the first minimal accuracy of the clustering (950th epochs), the output data distribution is scattered but each cluster cloud is not as divided as at 450th epochs.

In conclusion, the CNN model learns less from the semantic area but starts to focus more on the box area after the clustering starts to decrease. Saliency Map and Smooth-Grad are not effective when applied to the CNN model. In contrast, GradCAM has been shown to provide clear visualizations of the learning processes. Additionally, it introduces next problems that how much influence box has on the correlation between the two views and when the overfitting problem starts to seriously affect the model performance. If the visaulization can detect the overfiiting, it is possible to prevent overfitting in the model

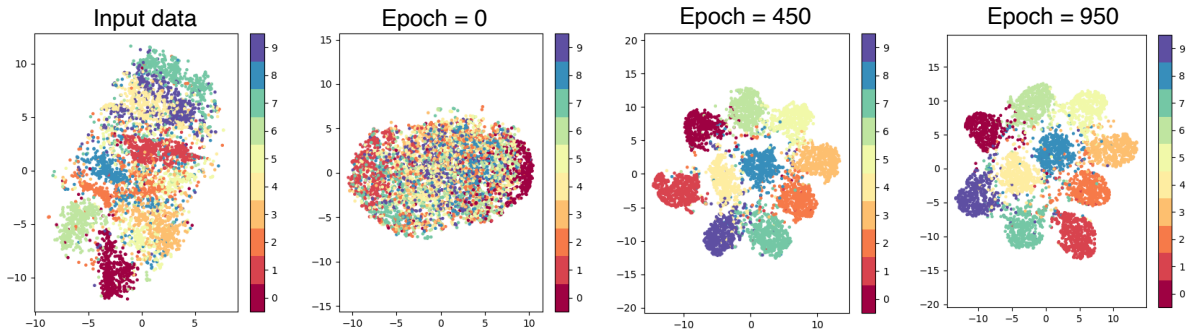

Figure 4.13: t-SNE in the CNN model

and the ouptuts generated by the model with the maximal correlation can represent the original views better.

4.3.2 Overfitting Evaluation

Baesd on the previous experiments, notice that the correlation of the outputs always increases while the accuray of the clustering decreases as the model continues to run. The decrease in the accuracy of the clustering may suggest that the model is overfitting, which is supported by the results of t-SNE. Therefore, to have a clearer and more intuitive display in which epoch the model starts to overfitting or has been strongly influenced by box, a new performance metric is introduced as the box attention, which can be understood as a conjunction matrix on the clustering accuracy of each digit.

An original image $\boldsymbol{I}$ has the corresponding GradCAM map $\boldsymbol{G} \in \mathbb{R}^{m_g \times n_g}$ and the box filter is $\boldsymbol{B} \in \mathbb{R}^{m \times n}$. First scale the $\boldsymbol{G}$ in the same size of $\boldsymbol{I}$ by a linear transformation,

$$\boldsymbol{G}'(i,j) = \frac{m-n}{m_g - n_g}\boldsymbol{G}(i,j) \ ,$$

where $\boldsymbol{G}'$ is the scaled GradCAM map and (i,j) represents the pixel value in the i-th row and j-th column. Then box attention b is defined as the average result of scaled GradCAM map times the box filter,

$$b = \mathrm{E}(\boldsymbol{G}' \cdot \boldsymbol{B}) \ . \tag{4.1}$$

As the definition of box attention, it indicates how much the GradCAM focuses on the box instead of the semantic position of the original image. When it is decreasing, the model learns less from the box position.

Considering that the CNN model has a better performance than the DNN model and GradCAM only works for the CNN model. This experiment only focuses on the CNN model, and the input data uses the clean MNIST data with one box. First, the correlation between the two views is increasing with the model running. Even let the model run 10,000 epochs, the correlation learned by the CNN model is still kept under 0.55 which is reasonable.

Although the accuracy of the classifier gives a keeping good result, the accuracy of the clustering tells that the correlation learned by the CNN model cannot well represent the original input data when the model runs more, as shown in Fig. 4.14. At around 3000th epochs, the accuracy of the clustering has first significantly decreased. From 6000th epochs to 9000th epochs, it decreases again.

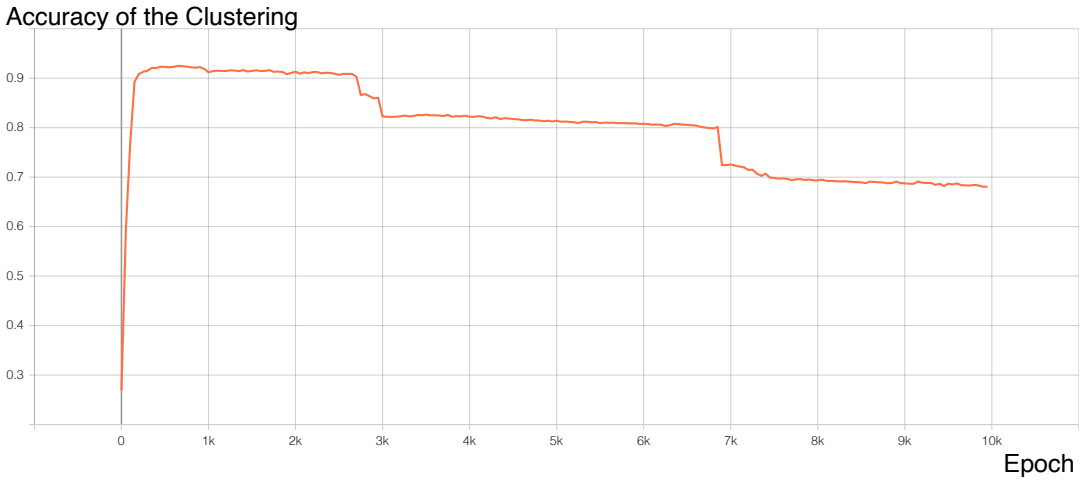

Figure 4.14: Accuracy of the clustering in the CNN model

The maximal accuracy of the clustering is at 600th epochs. At 3000th epochs and 6900th epochs, the accuracy of the clustering has significantly decreased. Thus, the visualized results at these moments and the final epoch are shown in Fig. 4.15. As the result in 4.3, the Saliency Map and SmoothGrad cannot tell the clear results about what the model learned. There are two different observations from the results of the visualization. GradCAM results of a digit 1 show that at the maximal accuracy time it finds the correlation between the two views focusing on both the semantic parts (digit 1) and the box position. At 3000th epochs the learning pixels in the digit 1 image look similar to those at 600th epochs, but the difference between the learning pixels and background pixels is smaller than at 600th epochs. At 6900th epochs and 10,000th epochs, the learning pixels in the digit 1 image only focus on the box position which indicates the model is overfitting. Observation of a digit 5 image are similar with the digit 1. These results support that the model always try to maximize the correlation blindly, especially when it iterates more epochs, the correlation might be learned more from the box area.

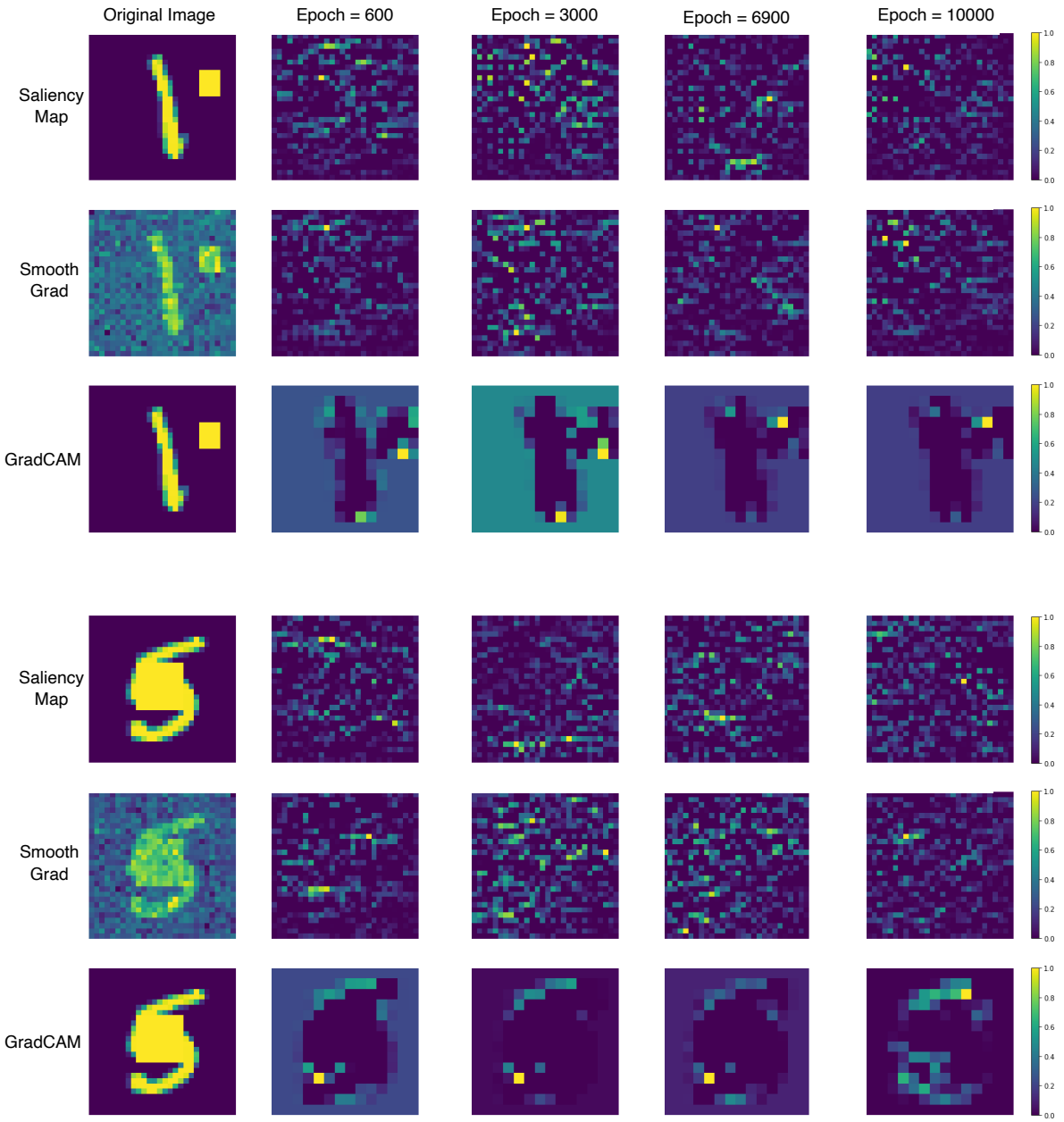

Figure 4.15: Visualization Results in CNN model of digit 1 and 5 images

In contrast, the results of a digit 3 image in Fig. 4.16 show that at 3000th epochs, the learning pixels focus more on the semantic area (digit 3). At 6900th epochs and 10,000th epochs, the focus shifts more towards the box area, but there is still some correlation with the semantic area (digit 3). Another similar results of GradCAM are a digit 7 image shown in Fig. 4.17

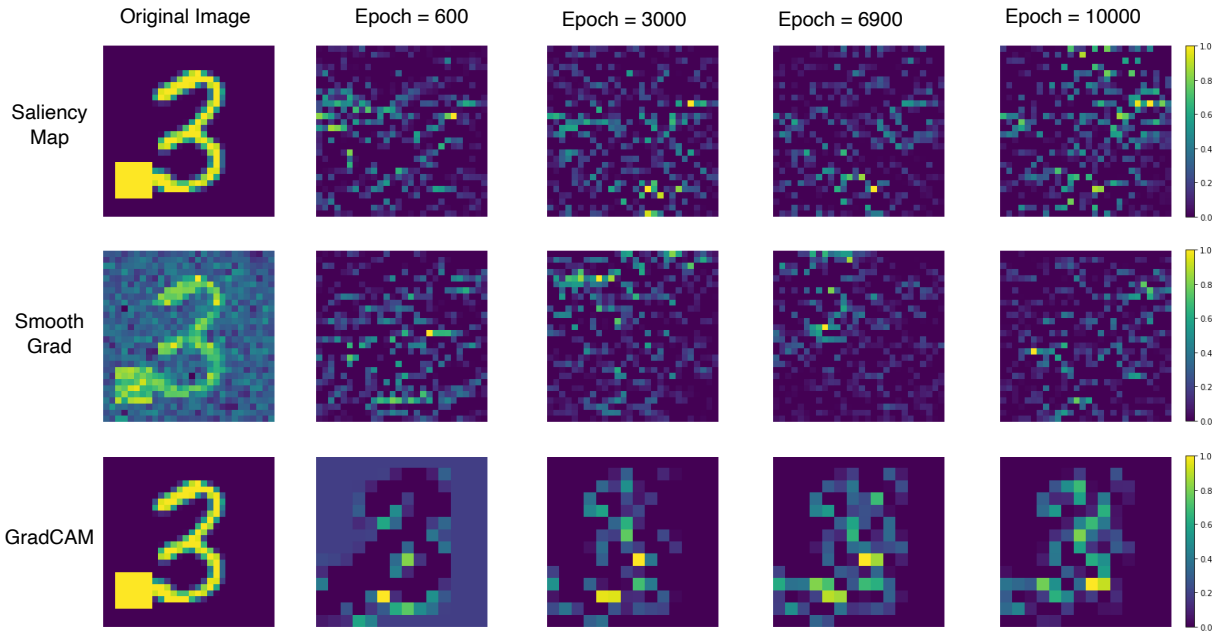

Figure 4.16: Visualization Results in CNN model of a digit 3 image

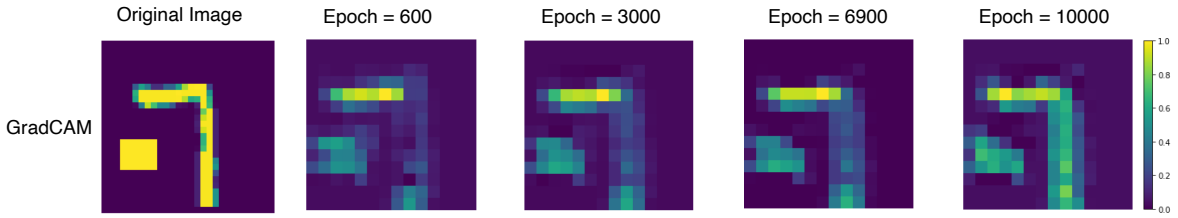

Figure 4.17: Visualization Results in CNN model of a digit 7 image

Additionally, the t-SNE results in Fig. 4.18 and the box attention in Fig.4.19 explain these observations. In t-SNE, all cluster clouds are well separated at 600th epochs, while they are getting closer at 3000th epochs. They are getting mixed at 6000th epochs and it is worse at 10,000th epochs. Especially, for some samples are merged in the middle. Moreover, box attention shows it starts focus more on the box after 3000the epochs, which the accuracy of the clustering starts to significantly decreased. From 6000th epochs to 9000th epochs, box attention decreases again, but the change of its value is not much.

Overall, the effectiveness of Saliency Map and SmoothGrad techniques are limited in the CNN model. In contrast, GradCAM effectively visualizes the learning process of the model. Both box attention and t-SNE can serve as valuable tools in evaluating the performance of the CNN model.

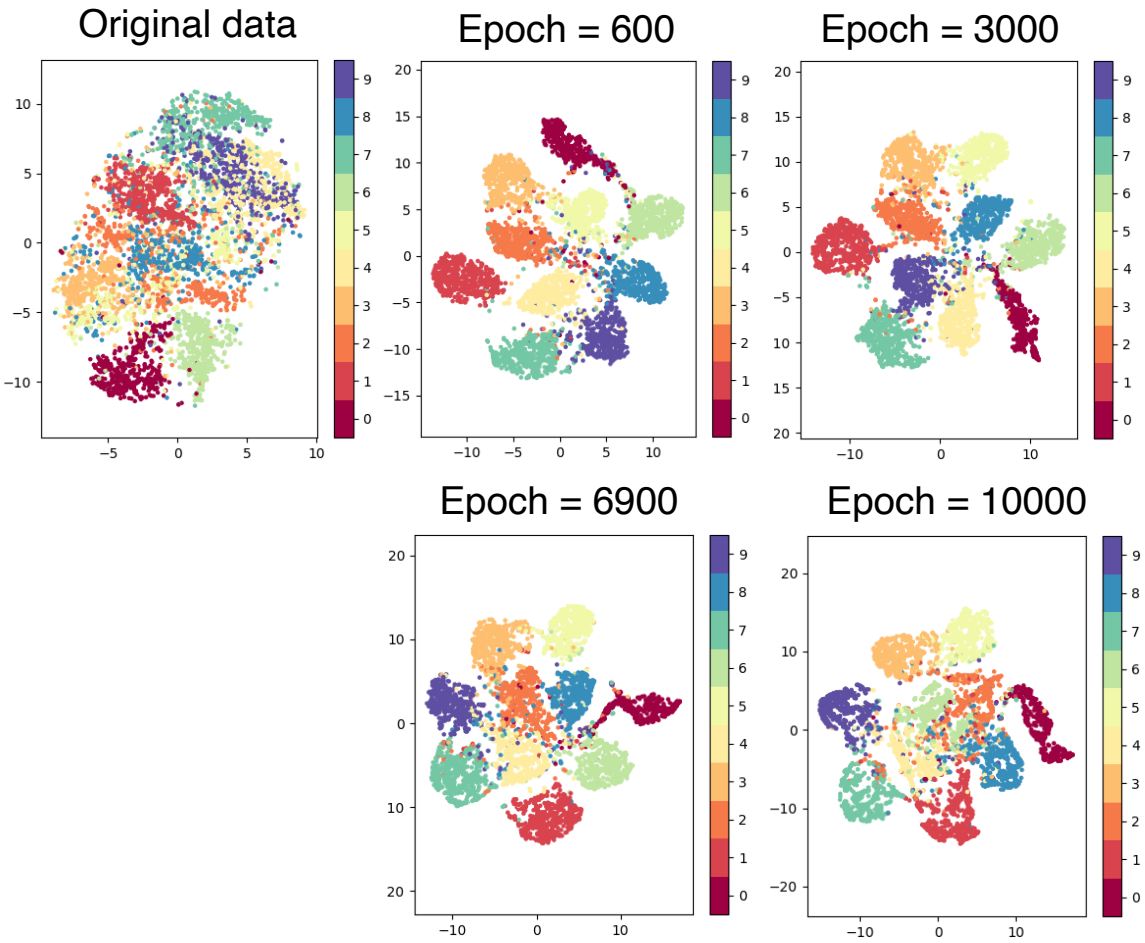

Figure 4.18: t-SNE in CNN model

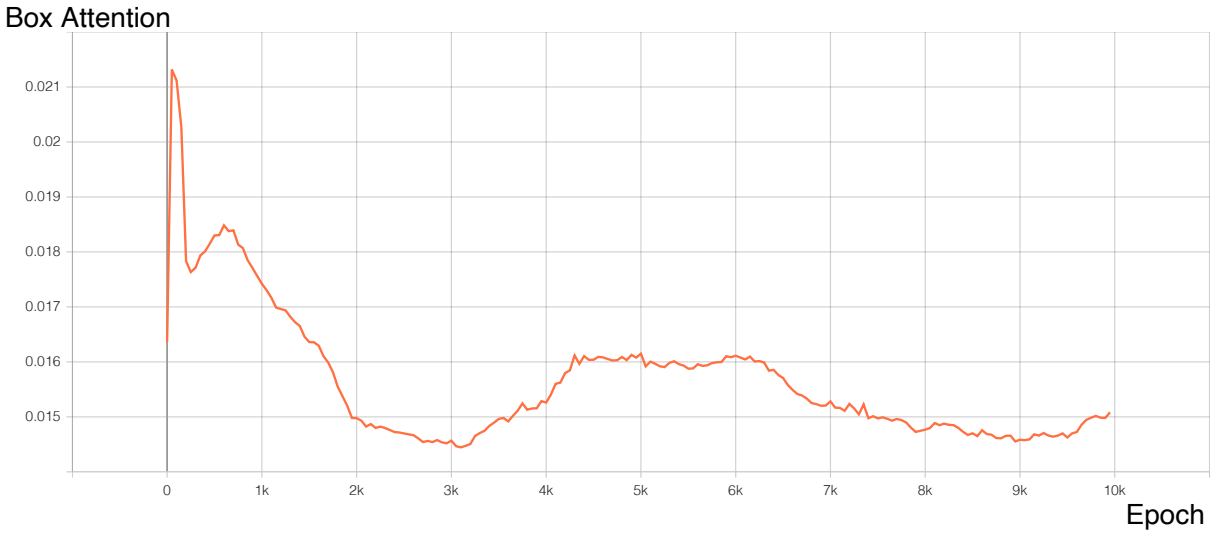

Figure 4.19: Box Attention in CNN model

To further verify the validity of the CNN model the clean MNIST with 2 boxes is also applied in the CNN model to have more comparisons. Consider that the accuracy of the classifier is always kept similar which cannot indicate more information for the CNN model and Saliency Map and SmoothGrad have bad performances, the rest experiment does not display these results anymore.

The correlation learned by the CNN model is still kept around 0.55 even though there should be more structural correlation due to two boxes. But at 6000th epochs, the accuracy of the clustering starts to decrease. Thus, Fig. 4.20 shows GradCAM results at 1300th epochs (maximum accuracy of the clustering), 6000th epochs (when the accuracy starts to decrease), and 9200th epochs (minimum accuracy). When the clustering has the better performance, the GradCAM of the two examples shows the CNN model learned the correlation around the semantic positions (digit 1 and digit 5). When the accuracy starts decreasing, the GradCAM maps more focus on the box area, especially the digit 1 image. In the last column at 9200th epochs, what the CNN model learned from the input data much more focus on the box area, especially the digit 5 image.

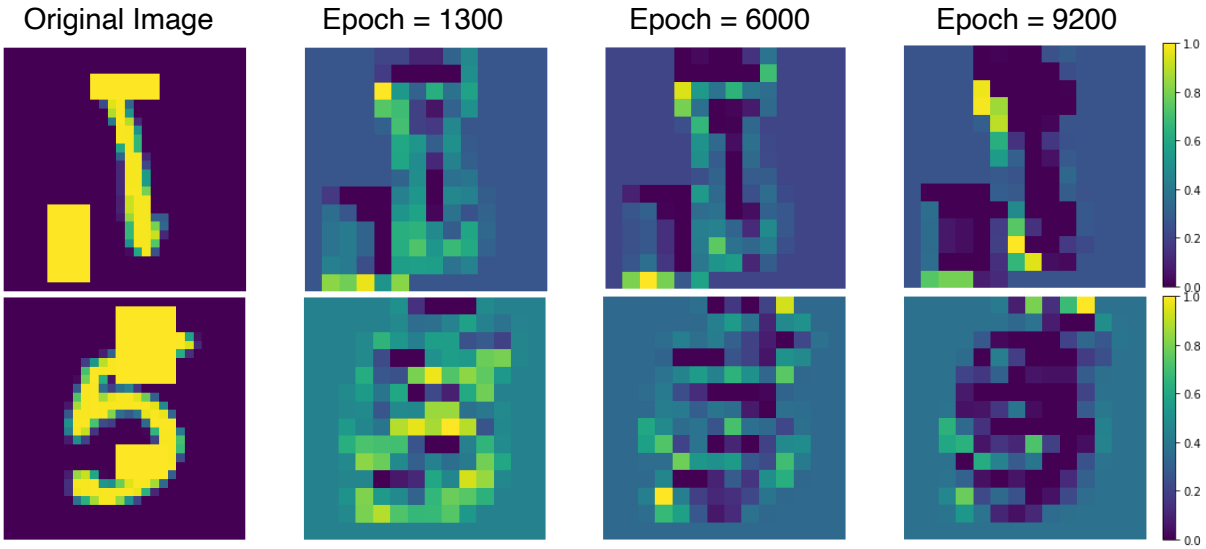

Figure 4.20: Visualization Results in the CNN model with 2 boxes clean MNIST

However, the t-SNE results in Fig. 4.21 and box attention in Fig. 4.22 show a different observation. As the model runs, the distribution of all digit clusters are similar, and box attention decreases, although it remains small.

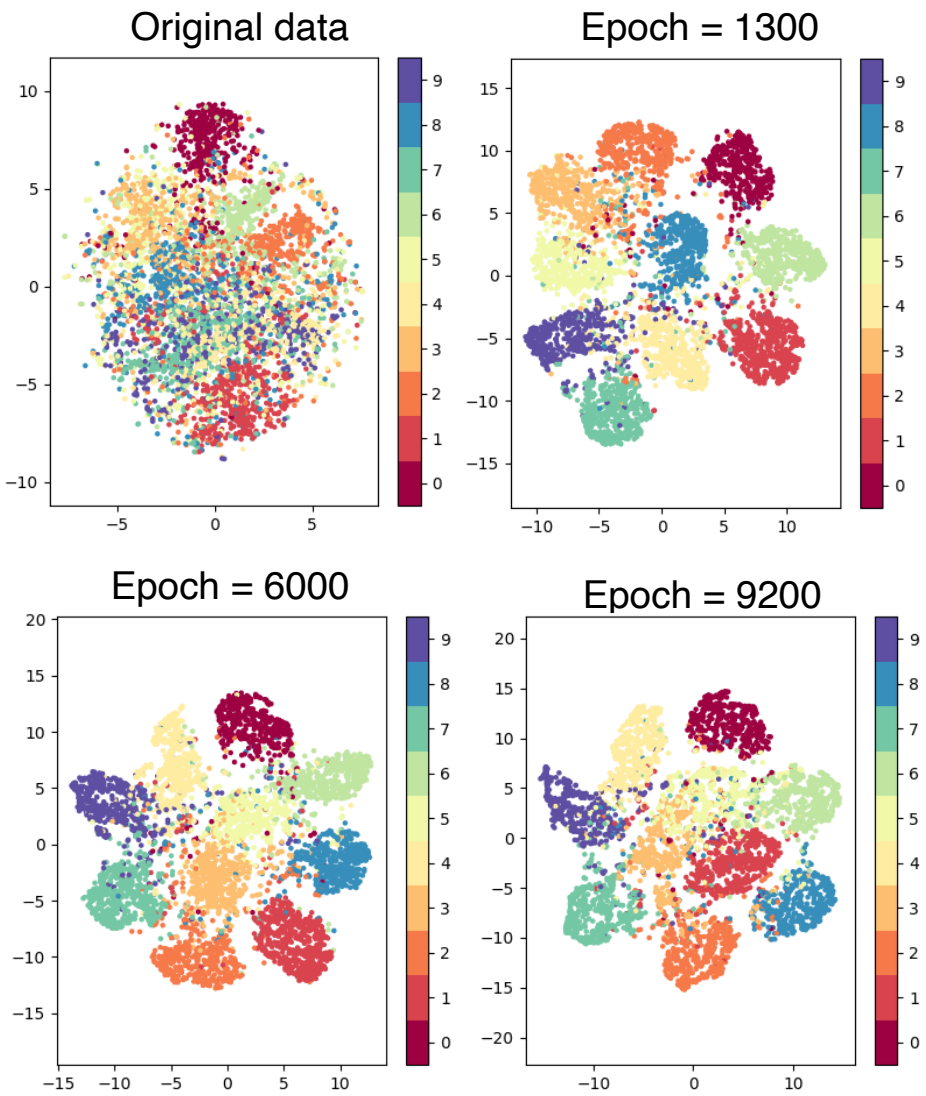

Figure 4.21: t-SNE in the CNN model with 2 boxes clean MNIST

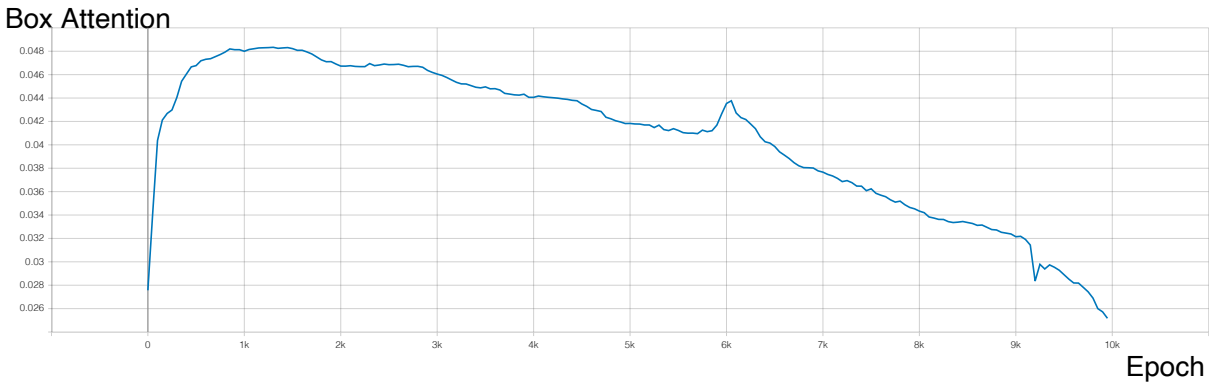

Figure 4.22: Box Attention in the CNN model with 2 boxes clean MNIST

In conclusion, GradCAM can be applied in the CNN model to provide detailed visualizations of the learning process at the pixel level. Compared to the results obtained

using Saliency Map and SmoothGrad in the DNN model, GradCAM has been found to be more effective at capturing the learning processes in the CNN model. Besides, it can provide some insight into the occurrence of overfitting during the learning process and indicate if the model is heavily influenced by the box. However, GradCAM shows the learning process through individual input images, while the correlation learned by the CNN model is between two views of input rather than individual images. As a result, it may be challenging to rely on the results of GradCAM, as its principle involves mapping transformation on a sample-by-sample basis, whereas the learning process in the CNN model occurs at the cluster level.

5 Conclusion and Outlook

5.1 Conclusion

We have seen that, the proposed convolutional neural network-based DCCAE model outperformed the original feedforward neural network-based DCCAE model. The incorporation of a convolutional neural network enabled the DCCAE effectively to exploit the dependent information across pixels and hence was able to learn the shared representations between the two views. Convolutional neural network can not only offer nonlinear transformation but also learn more information from the input data.

Overfitting occurs when the DCCAE attempts to maxmize the correlation between two views by projecting complex nonlinearity without actually learning the underlying shared properties of the data. This can cause the representations generated by the DCCAE that cannot represent the original data well. The proposed model is able to make more accurate representations, which indicates that it learns more from the original data and is less likely to overfitting.

The proposed model used three different techniques to realize a visual representation of the data. Experimental evidence suggests that Saliency Map and SmoothGrad are more effective when applied to simpler network structures, while GradCAM is particularly useful for providing a clear visualization of the learning processes in complex convolutional neural networks. Visualization is an important tool in data science and machine learning because it allows researchers to more easily understand and analyse complex data. By using multiple techniques, the proposed model is able to provide a more comprehensive visualization of the data, which is helpful for further analysis and interpretation.

Nonetheless, there are some limitations of the proposed model during the experiment process. t-SNE can provide a reliable visualization of cluster data through the use of different colors. It is a useful tool for visualizing high-dimensional data by projecting it onto a lower-dimensional space while preserving the relationships between data points. However, it is difficult to determine when overfitting occurs based on the results of t-SNE alone, as the rotation of all the clusters makes it hard to identify the start of overfitting. Using t-SNE in conjunction with other visualization techniques can help to estimate when overfitting occurs.

Secondly, the visualization results from the experiments are not easily interpretable for the input data or showing how the experimental model learns. Saliency Map, SmoothGrad and GradCAM are all techniques that are used to create visualizations of the classification process of a classifier. The classification process establishes a unique mapping between the input data and the labels, which means that each input sample is assigned a specific label. However, in this thesis, these visualization techniques are applied to visualize the learning process that maxmizes the correlation between each view of the input data, rather than focusing on the correlation between individual samples. It explains that in the same experiment, some randomly selected samples showed fluctuations in performance as the model was running, while there were also counter examples of a different outcome. Additionally, an extra classifier is built for Saliency Map and SmoothGrad to help identify the specific indices to be visualized, which increase the uncertainty of the visualized process which is that how the proposed model learns.

Despite some limitations, the proposed model has a improved performance compared to the original DCCAE model. It is able to learn more effectively from the two views and is less prone to overfitting. To increase interpretability, the model applied three visualization techniques: Saliency Map, SmoothGrad, and GradCAM, which allowed the model to provide a more comprehensive and interpretable visualization of the data.

5.2 Outlook

Preventing overfitting in correlation-based mutliview analysis is essential for future research. However, since a correlation-based mutliview analysis model is an unsupervised learning method, it is challenging to prevent overfitting as the model is not trained on labeled data and is expected to discover correlations between multiple views of the data.

One way to prevent overfitting is to use regularization properly, such as applying L1-norm, L2-norm, and regularization in the CCA loss function. Other regularization techniques, such as weight decay [LDS89] and dropout [SHK+14], can also be used. Weight decay adds a penalty term to the loss function of the model, which encourages the model to learn smaller weights and reduces the complexity of the model. Dropout randomly drops out neurons from the network during training, which can help to prevent the model from relying too much on any single neuron.

Another method to prevent overfitting is to explore more visualization techniques. Using a scatter plot to visualize the output of a correlation-based mutliview analysis model, and look for signs of overfitting, such as t-SNE which was applied in the experiment. Moreover, investigating ways to enhance the conjunction matrix can be another approach to prevent overfitting. One possibility could be to incorporate more information into the box attention, such as combining it with additional visualization results.

Overall, there are many ways to prevent overfitting in a correlation-based mutliview analysis model, and the most effective approach will depend on the specific problem and dataset. It is important and challenging to carefully evaluate the performance of the model and try different techniques to identify the approach that works best for a specific problem.

Bibliography

[AABL13] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR, 2013.

[AGGK18] Julius Adebayo, Justin Gilmer, Ian Goodfellow, and Been Kim. Local explanation methods for deep neural networks lack sensitivity to parameter values. *arXiv preprint arXiv:1810.03307*, 2018.

[Aka06] Shotaro Akaho. A kernel method for canonical correlation analysis. *arXiv preprint cs/0609071*, 2006.

[AMH+05] Takahisa Ando, Ken Mashitani, Masahiro Higashino, Hideyuki Kanayama, Haruhiko Murata, Yasuo Funazou, Naohisa Sakamoto, Hiroshi Hazama, Yasuo Ebara, and Koji Koyamada. Multi-view image integration system for glass-less 3d display. In *Stereoscopic Displays and Virtual Reality Systems XII*, volume 5664, pages 158–166. SPIE, 2005.

[And62] Theodore Wilbur Anderson. An introduction to multivariate statistical analysis. Technical report, Wiley New York, 1962.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[Bis95] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[BJ02] Francis R Bach and Michael I Jordan. Kernel independent component analysis. *Journal of machine learning research*, 3(Jul):1–48, 2002.

[Bon64] Raymond E Bonner. On some clustering techniques. *IBM journal of research and development*, 8(1):22–32, 1964.

[BR03] Matthew Barker and William Rayens. Partial least squares for discrimination. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 17(3):166–173, 2003.

[BS05] Steffen Bickel and Tobias Scheffer. Estimation of mixture models using co-em. In *European conference on machine learning*, pages 35–46. Springer, 2005.

[CCHM20] Lei Chen, Jianhui Chen, Hossein Hajimirsadeghi, and Greg Mori. Adapting grad-cam for embedding networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2794–2803, 2020.

[CHC12] Xi Chen, Liu Han, and Jaime Carbonell. Structured sparse canonical correlation analysis. In *Artificial intelligence and statistics*, pages 199–207. PMLR, 2012.

[CNHK11] Xiao Cai, Feiping Nie, Heng Huang, and Farhad Kamangar. Heterogeneous image feature integration via multi-modal spectral clustering. In *CVPR 2011*, pages 1977–1984. IEEE, 2011.

[DBDM03] Tijl De Bie and Bart De Moor. On the regularization of canonical correlation analysis. *Int. Sympos. ICA and BSS*, pages 785–790, 2003.

[DFU11] Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Multi-view learning of word embeddings via cca. *Advances in neural information processing systems*, 24, 2011.

[DSGLM10] Virginia R De Sa, Patrick W Gallagher, Joshua M Lewis, and Vicente L Malave. Multi-view kernel construction. *Machine learning*, 79(1):47–71, 2010.

[FHM+05] Jason Farquhar, David Hardoon, Hongying Meng, John Shawe-Taylor, and Sandor Szedmak. Two view learning: Svm-2k, theory and practice. *Advances in neural information processing systems*, 18, 2005.

[GA11] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *The Journal of Machine Learning Research*, 12:2211–2268, 2011.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[GKP02] Mihaela Gordan, Constantine Kotropoulos, and Ioannis Pitas. Application of support vector machines classifiers to visual speech recognition. In *Proceedings. International Conference on Image Processing*, volume 3, pages III–III. IEEE, 2002.

[GLWS20] Quanxue Gao, Huanhuan Lian, Qianqian Wang, and Gan Sun. Cross-modal subspace clustering via deep canonical correlation analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3938–3945, 2020.

[Goy18] Anil Goyal. *Learning a Multiview Weighted Majority Vote Classifier: Using PAC-Bayesian Theory and Boosting*. PhD thesis, 10 2018.

[HLBKK08] Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL-08: Hlt*, pages 771–779, 2008.

[HMMBST07] David R Hardoon, Janaina Mourao-Miranda, Michael Brammer, and John Shawe-Taylor. Unsupervised analysis of fmri data using kernel canonical correlation. *NeuroImage*, 37(4):1250–1259, 2007.

[Hot92] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer, 1992.

[HR20] Seyed Mohammad Hashemi and Mohammad Rahmati. Cross-domain recommender system using generalized canonical correlation analysis. *Knowledge and Information Systems*, 62(12):4625–4651, 2020.

[HSST04] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[JD88] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[JRHG12] Michal Joachimiak, Dmytro Rusanovskyy, Miska M Hannuksela, and Moncef Gabbouj. Multiview 3d video denoising in sliding 3d dct domain. In *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, pages 1109–1113. IEEE, 2012.

[KD11] Abhishek Kumar and Hal Daumé. A co-training approach for multi-view spectral clustering. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 393–400, 2011.

[KSE05] Einat Kidron, Yoav Y Schechner, and Michael Elad. Pixels that sound. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 88–95. IEEE, 2005.

[LBBH98a] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBBH98b] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. The mnist database of handwritten digits. *Computation and neural systems*, 1(1):77–86, 1998.

[LDS89] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[LLT+15] Meng Liu, Yong Luo, Dacheng Tao, Chao Xu, and Yonggang Wen. Low-rank multi-view learning in matrix completion for multi-label image classification. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[LWB+15] Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Deep multilingual correlation for improved word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 250–256, 2015.

[MKH+95] Niels JS Morch, Ulrik Kjems, Lars Kai Hansen, Claus Svarer, Ian Law, Benny Lautrup, Steve Strother, and Kelly Rehm. Visualization of neural networks using saliency maps. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 2085–2090. IEEE, 1995.

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[MRB01] Thomas Melzer, Michael Reiter, and Horst Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *International Conference on Artificial Neural Networks*, pages 353–360. Springer, 2001.

[MV16] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[Nie02] Allan Aasbjerg Nielsen. Multiset canonical correlations analysis and multispectral, truly multitemporal remote sensing data. *IEEE transactions on image processing*, 11(3):293–305, 2002.

[NKK+11] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *ICML*, 2011.

[NW20] Nam D Nguyen and Daifeng Wang. Multiview learning for understanding functional multiomics. *PLoS computational biology*, 16(4):e1007677, 2020.

[Pea96] Karl Pearson. Vii. mathematical contributions to the theory of evolution.—iii. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, (187):253–318, 1896.

[PLI+15] Parvathy Sudhir Pillai, Tze-Yun Leong, Alzheimer's Disease Neuroimaging Initiative, et al. Fusing heterogeneous data for alzheimer's disease classification. In *MEDINFO 2015: eHealth-enabled Health*, pages 731–735. IOS Press, 2015.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[RK05] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, pages 34–51. Springer, 2005.

[RS18] Nimrod Rappoport and Ron Shamir. Multi-omic and multi-view clustering algorithms: review and cancer benchmark. *Nucleic acids research*, 46(20):10546–10562, 2018.

[RSSTG13] Jan Rupnik, Primoz Skraba, John Shawe-Taylor, and Sabrina Guettes. A comparison of relaxations of multiset cannonical correlation analysis and applications. *arXiv preprint arXiv:1302.0974*, 2013.

[SCD+17] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[Sha48] Claude E Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[SNB05] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*, volume 2005, pages 74–79. Citeseer, 2005.

[SOH21] Ryosuke Sawata, Takahiro Ogawa, and Miki Haseyama. Human-centered favorite music classification using eeg-based individual music preference via deep time-series cca. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1320–1324. IEEE, 2021.

[SR08] Vikas Sindhwani and David S Rosenberg. An rkhs for multi-view learning and manifold co-regularization. In *Proceedings of the 25th international conference on Machine learning*, pages 976–983, 2008.

[SS12] Nitish Srivastava and Russ R Salakhutdinov. Multimodal learning with deep boltzmann machines. *Advances in neural information processing systems*, 25, 2012.

[SSYY20] Xuli Sun, Shiliang Sun, Minzhi Yin, and Hao Yang. Hybrid neural conditional random fields for multi-view sequence labeling. *Knowledge-Based Systems*, 189:105151, 2020.

[STK+17] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[Sun11] Shiliang Sun. Multi-view laplacian support vector machines. In *International Conference on Advanced Data Mining and Applications*, pages 209–222. Springer, 2011.

[Sun13] Shiliang Sun. A survey of multi-view machine learning. *Neural computing and applications*, 23(7):2031–2038, 2013.

[Vap95] Vladimir N Vapnik. Support vector networks. *Machine learning*, 20(3):273–297, 1995.

[VCST02] Alexei Vinokourov, Nello Cristianini, and John Shawe-Taylor. Inferring a semantic representation of text via cross-language correlation analysis. *Advances in neural information processing systems*, 15, 2002.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[VDW77] Arnold L Van Den Wollenberg. Redundancy analysis an alternative for canonical correlation analysis. *Psychometrika*, 42(2):207–219, 1977.

[WALB15a] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International conference on machine learning*, pages 1083–1092. PMLR, 2015.

[WALB15b] Weiran Wang, Raman Arora, Karen Livescu, and Jeff A Bilmes. Unsupervised learning of acoustic features via deep canonical correlation analysis. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4590–4594. IEEE, 2015.

[WALS15] Weiran Wang, Raman Arora, Karen Livescu, and Nathan Srebro. Stochastic optimization for deep cca via nonlinear orthogonal iterations. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 688–695. IEEE, 2015.

[Wil88] James Hardy Wilkinson. *The algebraic eigenvalue problem*. Oxford University Press, Inc., 1988.

[WLY+19] Christopher M Wilson, Kaiqiao Li, Xiaoqing Yu, Pei-Fen Kuan, and Xuefeng Wang. Multiple-kernel learning for genomic data mining and prediction. *BMC bioinformatics*, 20(1):1–7, 2019.

[WPJ+18] Raymond K Walters, Renato Polimanti, Emma C Johnson, Jeanette N McClintick, Mark J Adams, Amy E Adkins, Fazil Aliev, Silviu-Alin Bacanu, Anthony Batzler, Sarah Bertelsen, et al. Transancestral gwas of alcohol dependence reveals common genetic underpinnings with psychiatric disorders. *Nature neuroscience*, 21(12):1656–1669, 2018.

[WZSY12] Martha White, Xinhua Zhang, Dale Schuurmans, and Yao-liang Yu. Convex multi-view subspace learning. *Advances in neural information processing systems*, 25, 2012.

[YHM+21] Xiaoqiang Yan, Shizhe Hu, Yiqiao Mao, Yangdong Ye, and Hui Yu. Deep multi-view learning methods: a review. *Neurocomputing*, 448:106–129, 2021.

[YS19] Jun Yin and Shiliang Sun. Multiview uncorrelated locality preserving projection. *IEEE transactions on neural networks and learning systems*, 31(9):3442–3455, 2019.

[YSYW15] Huanjing Yue, Xiaoyan Sun, Jingyu Yang, and Feng Wu. Image denoising by exploring external and internal correlations. *IEEE Transactions on Image Processing*, 24(6):1967–1982, 2015.

[ZHJ19] Shiwei Zhou, Yu-Hen Hu, and Hongrui Jiang. Multi-view image denoising using convolutional neural network. *Sensors*, 19(11):2597, 2019.

[ZZPZ16] Yanyan Zhang, Jianchun Zhang, Zhisong Pan, and Daoqiang Zhang. Multi-view dimensionality reduction via canonical random correlation analysis. *Frontiers of Computer Science*, 10(5):856–869, 2016.