technische universität
dortmund

Attribute-Based Handwriting Word Recognition with deep Convolutional Neural Networks

Master Thesis

Hyunho Mo
May 31, 2019

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Fabian Wolf, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

CONTENTS

1 INTRODUCTION

Despite digital documents produced by electronic devices are used in many fields, crucial documents, such as official documents with signature and historical documents still rely on paper support and handwriting. Along with the improvement in image processing, this large amount of handwritten documents are processed and preserved in a digital form. Digitization of large collections of documents enables to copy and display its contents via electronic devices. In order to effectively obtain knowledge from these collections, it is necessary to develop methods that extracts information from such a large number of handwriting documents images.

One crucial task is to recognize the word image contents, and convert it into a transcription. This is called word recognition. Within the scope of segmentation-based offline handwriting word recognition aided by a lexicon, the segmented words are described by a set of features. These are used to train a classifier to build a model. In order to perform recognition, the approach of Bag of Features (BoF) extracts the features of an input image. Then, these features are classified based on the trained model into one of the candidate words in the lexicon. Recently, Convolutional Neural Network (CNN) are widely used as integrated framework which replaces above pipeline from pre-processing to classification. CNNs are successfully used for handwritten digit recognition in [LBD+90], and first applied to the MNIST dataset for handwritten digits recognition [LBBH98a]. As such, CNNs are not only originated from handwriting word recognition, but recently used to achieve state-of-the-art results in that birthplace as the form of CNN-RNN hybrid architecture [DKMJ18].

Whereas the CNN architecture is trained through supervised learning algorithm, handwriting word recognition task is considered as a special case of zero-data learning. Therefore, the direct use of a CNN as a conventional multi-class classifier with softmax is not an optimal approach for handwriting word recognition. The reason is that CNNs in such application cannot make prediction for unseen classes of words in a supplied dictionary or lexicon. This problem is also called Out of Vocabulary (OOV) problem. Here, attribute-based classification can be used to address the OOV problem. This approach suggest to use attributes as intermediate representation for both images and transcriptions of word. Images are transformed into probability of existing attribute given an image called attribute score. Transcriptions are transformed into binary

attribute representation reflecting existence of attributes. Then, estimated attribute scores are classified into a transcription of class.

In this thesis, attribute-based classification is used for handwriting word recognition, and it is dubbed as attribute-based word recognition. A special type of deep CNN called PHOCNet [SF18] is employed to estimate attribute scores from word images in this work. The network is originally designed for word spotting, and achieves state-of-the-art results for word spotting on several handwriting datasets. Based on this results, the competence of PHOCNet that predicts attributes from image is exploited to the attribute-based word recognition in this thesis.

With employing the PHOCNet, the recognition task is reduced to classify estimated attributes score by PHOCNet into a class of lexicon. For this multi-class classification task, similar works as [AGFV14] and [PW16] conventionally consider a simple nearest neighbour search. They also propose to use Canonical Correlation Analysis (CCA) for deriving new representations that maximize correlation between two different attribute representations. In this thesis, each of the two methods is cascaded to the PHOCNet as a baseline approach. Then, this thesis also proposes to employ a probabilistic classifier called Direct Attribute Prediction (DAP) [LNH14] for attribute-based word recognition. Additionally, the thesis introduces two novel methods called CCADAP and MLPs *with pseudo samples* which can improve the recognition performance compared to simple nearest neighbour search. CCADAP is a way of using CCA for a temporal recognition of input handwriting documents predicted by DAP. MLPs with pseudo samples proposes to train a conventional softmax MLPs with pseudo samples for zero-data learning.

This thesis is structured as follows. Chapter 2 explains the fundamental concepts of Artificial Neural Network (ANN) and CCA which are used in this thesis. In chapter 3, attribute-based classification is introduced, and two different CNNs estimating attributes from image, PHOCNet and CNN-N-Gram, are described. Then, chapter 4 introduces attribute-based word recognition and its different methods. The experimental results of the methods in chapter 4 are evaluated in chapter 5. Finally, conclusions are drawn from above chapters and discussed in chapter 6.

2

FUNDAMENTALS

2.1 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) is inspired by biological neural networks which constitute animal brains. This computing system is based on a collection of connected elementary units called artificial neurons. The chapter starts from the type of artificial neuron called a perceptron which were inspired by earlier work called McCulloch–Pitts (MCP) neuron, and introduces the motivation and the capability of its multi-layered connection. The variation of perceptrons with learning algorithm, sigmoid neurons, inspire current use ANN in machine learning. Therefore, the chapter also covers sigmoid neurons, learning algorithm and different kinds of cost function. In addition, the chapter also discuss one of the central challenges in machine learning called generalization and its related issues. At the end of the chapter, Convolutional Neural networks(CNNs) which have been widely used in object recognition is also introduced. [Roj96], [Nie15].

2.1.1 *Perceptrons and Multi-Layer Perceptrons (MLPs)*

Perceptrons were developed in the 1950s and 1960s by the scientist Frank Rosenblatt [Ros58], [Nie15]. A perceptron is the single computing unit which take binary inputs and computes a single binary output. He proposed the use of real numbers weights which express the importance of inputs for computing output. If weighted sum of inputs is greater than real number threshold, θ, then the output is one, and otherwise the output is zero. The algebraic expression of computing output is like as below.

$$y = \begin{cases} 1 & \text{if } \sum_{j=1}^{N} w_j x_j > \theta \\ 0 & \text{otherwise} \end{cases} \tag{2.1.1}$$

To simplify described perceptron, the threshold can be moved to the other side of inequality, and the negative value of threshold is called bias.

$$y = \begin{cases} 1 & \text{if } \sum_{j=1}^{N} w_j x_j + b > 0, \text{ where, } b = -\theta \\ 0 & \text{otherwise} \end{cases} \tag{2.1.2}$$

In general, the task of perceptron can be interpreted as the simple model that makes decisions by gathering weighted evidence. Technically, it can be used to compute elementary logical functions. Let assume there are two inputs for perceptron, and it then yields below expression.

$$y = \begin{cases} 1 & \text{if } x_2 > -\frac{w_1}{w_2} x - \frac{b}{w_2} \\ 0 & \text{otherwise} \end{cases} \tag{2.1.3}$$

The conditional statement for output above can be concerned as hyperplane. Computability of logical functions of perceptron can be usually depicted as follows. The

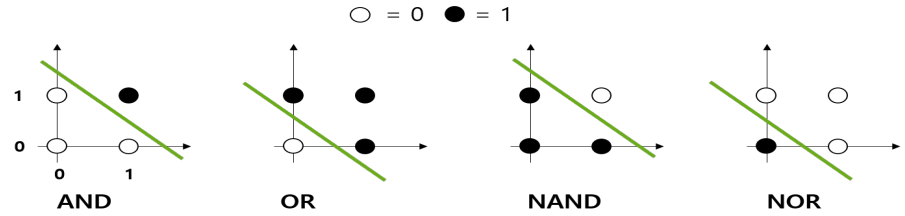


Figure 2.1.1: Computability of logical functions of perceptron.

figure 2.1.1 shows that single perceptron separates plane in two half planes, and it can then compute simple logical functions AND, OR, NAND and NOR. However,

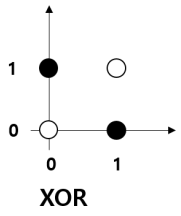


Figure 2.1.2: XOR problem of perceptron.

as shown in the figure 2.1.2, XOR problem cannot be solved by separating space in two subspaces with hyperplane. This limitation of single perceptron and single-layer perceptrons is tackled by adding a layer. Each percetron in first layer of the network

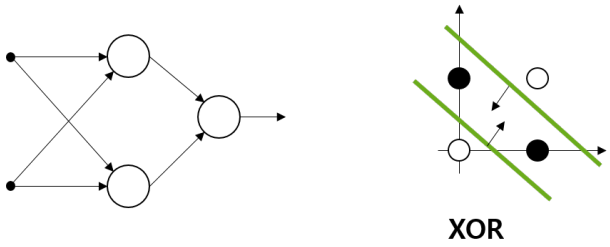


Figure 2.1.3: Solving XOR problem with three neurons in two-layers.

in figure 2.1.3 seperates space with each hyperplane, and perceptron of second layer then make possible to solve XOR problem by computing logical function, 'AND', for outputs of first layer perceptrons. With generalize the task of two-layer perceptrons depicted in figure 2.1.3, many perceptrons in second layer can identify convex sets. A three-layer perceptrons theoretically have capability of identifying arbitrary sets. Any arbitrary set can be partitioned into several convex sets, and these sets can be representable by second layer. And those can then be combined in third layer. Therefore, theoretically, more than three layers is not necessary, and a three-layer perceptrons already has the capability of classifying any complex patterns. However, even Multi-Layer Perceptrons (MLPs) have infinite theoretical possibility, it cannot be fully exploited in practice because of the large number of perceptrons. It will not be possible for person to tune the weights and biases of them manually. For that reason, the learning algorithm for the network parameters is introduced.

2.1.2 *Sigmoid Neurons*

The learning algorithm of the network is based on the trial and error. In the network, small change of weight in any neuron may cause a small change in network output as shown in figure 2.1.4 The evaluation of changed output provide the clues following small changes of the weight. With this simple learning algorithm, network change the weights and biases iteratively to produce better output. However, the above simple learning algorithm is not effectively applicable for perceptrons, because the output of perceptron is whether one or zero, i.e.,binary. Specifically, changing of weights only can cause flip the output, so learning algorithm hard to find proper direction

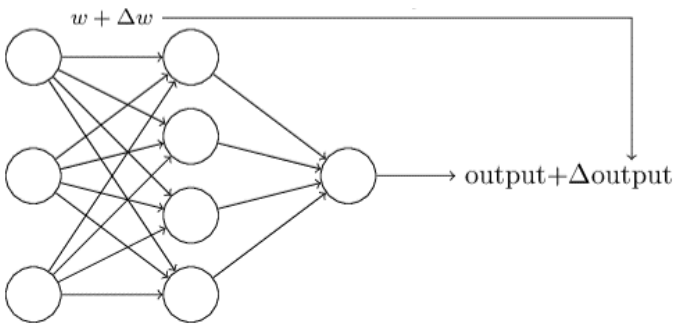


Figure 2.1.4: General idea of learning algorithm for the network [Nie15].

and degree of changing weights. A sigmoid neuron was introduced to address the problem. Each input for the sigmoid neuron is real values between 0 and 1. The output is computed by squashing accumulated weighted input with sigmoid function. It can be algebraically written as bellow.

$$\text{output} = \sigma(w \cdot x + b) \tag{2.1.4}$$

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}} \tag{2.1.5}$$

And, the shape of sigmoid function is plotted below 2.1.5 The use of sigmoid function

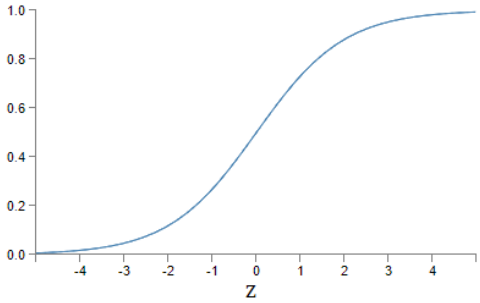


Figure 2.1.5: Sigmoid function [Nie15].

make possible for each neuron to have output as real number larger than 0 and smaller than 1. And the smoothness of sigmoid activation function does important role in network learning. With considering the smoothness of function as partial derivative,

the small change of output with respect to small change of parameters can then be written as (2.1.6) [Nie15].

$$\Delta output \approx \sum_j \frac{\partial output}{\partial w_j} \Delta w_j + \frac{\partial output}{\partial b} \Delta b \tag{2.1.6}$$

The binary output of perceptron can be described as activation with not differentiable signum function. But, in the case of sigmoid neuron, $\Delta output$ comes out as linear function of Δw_j and $\Delta\, b$.

Rectified Linear Unit (ReLU)

The section only discuss about sigmoid neurons which used sigmoid as the activation function. One of well-known variation on the sigmoid neuron is the rectified linear neuron or rectified linear unit [Nie15]. The output of a rectified linear unit with input x, weight vector w, and bias b is given by [Nie15]

$$max(0, w \cdot x + b) \tag{2.1.7}$$

Graphically, the rectifying function max(0,z) looks like figure 2.1.6. Depending on

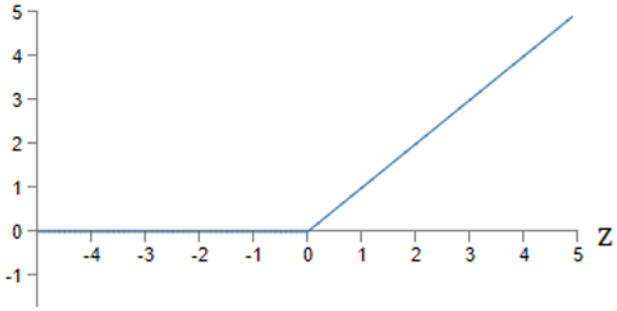


Figure 2.1.6: The rectifying function max(0,z) [Nie15].

the application, the use of above rectifying function as activation function, ReLU, outperform sigmoid networks. It will be discussed in later section.

2.1.3 *Loss Functions*

From the outline of network learning shown in figure 2.1.4, more specific goal of learning is to find weights and biases which minimize difference between network

output and target value for all training inputs. To derive algebraic expression and to apply mathematical optimization, the difference need to be mapped onto a real number. This function is called a loss function or cost function.

Quadratic Cost

It is also known as the mean squared error. The function is described in equation (2.1.8).

$$C(\theta) = \frac{1}{2n} \sum_{i=1}^{n} \|\hat{y}^{(i)} - y^{(i)}\|^2 \tag{2.1.8}$$

θ denotes the collections of all parameters in network, n is the total number of training samples. And $\hat{y}^{(i)}$ is network output for i-th training sample, and $y^{(i)}$ is the target value also called label of i-th training sample. The quadratic cost function is non-negative because it is defined as summation of squared error. Therefore, learning algorithm is simply find parameters which minimize above cost function.

Softmax log-Loss

In the use of ANN for multi-class classifier, the label of training sample can be one-hot encoded vector which indicates target class as one. In the probabilistic point of view, the goal of ANN is to estimate output class probability for each class in test. To get the representation of probability as output, fully connected layer with softmax function is widely used for output neurons' activation. Let z_j^L as computed output for j-th neuron in output layer L, and $\hat{p}_j$ is the probability of j-th class, then softmax function can be written as follows.

$$\hat{p}_j = \frac{e^{z_j^L}}{\sum_{k=1}^{K} e^{z_k^L}} \tag{2.1.9}$$

Each output of the softmax layer can be concerned as probability of each class, because summation of all outputs is one.

$$\sum_{j=1}^{K} \hat{p}_j = \frac{\sum_{j=1}^{K} e^{z_j^L}}{\sum_{k=1}^{K} e^{z_k^L}} = 1 \tag{2.1.10}$$

From above representation, loss function can be derived by computing cross-entropy.

$$E = -\frac{1}{N}\sum_{i=1}^{N} l^{(i)} \cdot \log \hat{p}^{(i)} \tag{2.1.11}$$

The $l^{(i)}$ vector of above equation is one-hot encoded vector which contain one at the index of class label of i-th sample.

Binary Logistic Loss

An ANN can be used as multi-label classifier as well as multi-class classifier. Especially, when the network is used to predict binary n-out-of-k representations, the Binary Logistic Loss function is mainly used for learning in the network. The loss function can be derived from a probabilistic point of view. Let θ be the collections of all parameters in network, and $y^{(i)}$ represents the target output given input $x^{(i)}$. The goal of learning which stated in the beginning of this subsection is then can be interpreted as to find parameters, $\hat{\theta}$, which maximize the likelihood of correct prediction, label $y^{(i)}$, given input for total n number of samples [SF17].

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(y^{(i)} \mid x^{(i)}, \theta) \tag{2.1.12}$$

To derive loss function from above objective equation, let assume conditional independence for M number of each output given input. Then each likelihood can be written as below product form.

$$p(y_1, ..., y_M \mid x^{(i)}, \theta) = \prod_{m=1}^{M} p(y_m \mid x^{(i)}, \theta) \tag{2.1.13}$$

The binary label y_m only has the value one or zero, and each prediction can be denoted as $\hat{y_m}$ and $1 - \hat{y_m}$ respectively.

$$p(y_m = 1 \mid x^{(i)}, \theta) = \hat{y}_m \tag{2.1.14}$$

$$p(y_m = 0 \mid x^{(i)}, \theta) = 1 - \hat{y}_m \tag{2.1.15}$$

From (2.1.14), each output probability, $\hat{y}_m$, follows Bernoulli distribution. Therefore, the product of each likelihood (2.1.13) can be rewritten as below.

$$\prod_{m=1}^{M} p(y_m \mid x^{(i)}, \theta) \sim \prod_{m=1}^{M} \hat{y}_m^{y_m} (1-\hat{y}_m)^{(1-y_m)}, \quad y_m \in \{0,1\} \tag{2.1.16}$$

Finally, the Binary Logistic Loss function is negative-log of (2.1.16).

$$l(\hat{y}, y) = -\frac{1}{M} \sum_{m=1}^{M} [y_m \log \hat{y}_m + (1-y_m) \log(1-\hat{y}_m)] \tag{2.1.17}$$

2.1.4 *Learning with Gradient Descent*

With combining the statement about learning in seciont 2.1.2 and 2.1.3, the goal of network train is to find network parameters which make the cost, also called loss, as small as possible. Because the loss function of ANN depends on very large number of parameters, computing derivatives for all parameters with calculus to find extremum of loss function is not reasonable. For this reason, Gradient Descent (GD) is adapted as learning algorithm for ANN. In this subsection, the weights are only considered as parameters to derive analytic representation, and will generalize to bias. The change of cost function with respect to the change of N number of weights analytically can be written as

$$\Delta C \approx \frac{\partial C}{\partial w_1} \Delta w_1 +, ..., + \frac{\partial C}{\partial w_N} \Delta w_N. \tag{2.1.18}$$

The direction of change, ΔC, should be negative because the goal is to find minimum of C. Let define a vectors of changes in weight as Δw and gradient of C which is a vector of partial derivatives as ∇C.

$$\Delta w = [\Delta w_1, ..., \Delta w_N]^T \tag{2.1.19}$$

$$\nabla C = [\frac{\partial C}{\partial w_1}, ..., \frac{\partial C}{\partial w_N}]^T \tag{2.1.20}$$

Then (2.1.18) can be written as

$$\Delta C \approx \nabla C \cdot \Delta w. \tag{2.1.21}$$

Now, let assume the change rule of weight, Δw, with positive learning rate η as below.

$$\Delta w = -\eta \nabla C \tag{2.1.22}$$

Then (2.1.21) guarantees negative direction of changing C, ΔC, because

$$\Delta C \approx -\eta \nabla C \cdot \nabla C = -\eta \|\nabla C\|^2. \tag{2.1.23}$$

With the proof of $\Delta C \leqslant 0$ in (2.1.23), the assumed change rule of weight in (2.1.22) is applicable. Therefore, the rule of changing weight can be written as below with notation w' as updated weight.

$$w \to w' = w - \eta \nabla C \tag{2.1.24}$$

In the iteration of gradient descent, update rule of each parameter can then be written as below with denoting k and l as each weight and bias respectively.

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{2.1.25}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l} \tag{2.1.26}$$

Stochastic Gradient Descent (SGD)

As shown in the loss functions, (2.1.8) and (2.1.11), they are an average over costs for individual training samples. When N is the number of samples, It can be analytically written as

$$C = \frac{1}{N} \sum_{i=1}^{N} C_{x_i}. \tag{2.1.27}$$

Therefore, when use a large amount of training samples, above (2.1.27) full batch gradient descent requires huge computation, and slow down the training. To tackle those problems, Stochastic Gradient Descent (SGD) randomly take small number of training samples, and compute gradient with them to estimate the overall gradient.

The subset of full batch, X_{mini}, which contains randomly choosen M samples is called minibatch.

$$\{x_1^{(rnd)}, ..., x_M^{(rnd)}\} \in X_{mini} \subset X \tag{2.1.28}$$

$$\frac{\sum_{j=1}^{M} \nabla C_{x_j^{(rnd)}}}{M} \approx \frac{\sum_{i=1}^{N} \nabla C_{x_i}}{N} = \nabla C \tag{2.1.29}$$

With above estimation, the update rule of each parameter in SGD can be written as

$$w_k \to w_k' = w_k - \eta \frac{1}{M} \sum_{j=1}^{M} \frac{\partial C_{x_j^{(rnd)}}}{\partial w_k} \tag{2.1.30}$$

$$b_l \to b_l' = b_l - \eta \frac{1}{M} \sum_{j=1}^{M} \frac{\partial C_{x_j^{(rnd)}}}{\partial b_l}. \tag{2.1.31}$$

SGD takes unfolded subset, X_{mini}, in iteration until covers entire training sample set, X, and it is called to complete an epoch of training.

2.1.5 *Backpropagation*

In previous section 2.1.4 the way of updating network parameters with using gradient is only discussed. However, an efficient computing of gradient also matter for learning in ANN. A fast gradient computing algorithm known as backpropagation is widely used for learning in neural networks. The backpropagation algorithm was originally introduced in the 1970s, but its importance wasn't fully appreciated until a famous 1986 paper [DR88] by David Rumelhart, Geoffrey Hinton, and Ronald Williams [Nie15]. That paper describes several neural networks where backpropagation works far faster than earlier approaches to learning, making it possible to use neural nets to solve problems which had previously been insoluble [Nie15]. The idea behind backpropagation is that the error propagates back. it can be derived from simple two layer example below, and can generalize to other MLPs which have more than one hidden layers. In figure 2.1.7, x_i denotes inputs, and y_j and z_k are values after L-1 and L layer respectively. And let assume both layer use sigmoid activation (2.1.4). Then

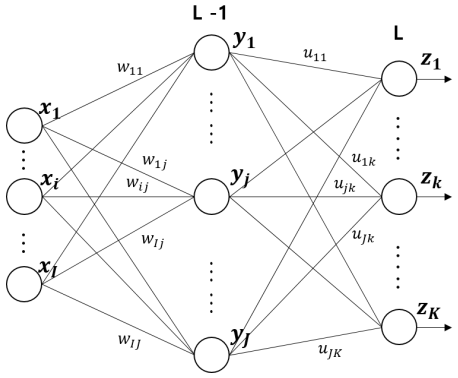


Figure 2.1.7: Example of two layer network to derive back propagation algorithm.

output of j-th neuron in L-1 layer and that of k-th neuron in L layer can be written as follow.

$$y_j = \sigma\left(\sum_{i=1}^{I} w_{ij}x_i\right) = \sigma(w_j^T \cdot x), \quad \text{where } w_j^T = [w_{1j}, ..., w_{IJ}]^T \tag{2.1.32}$$

$$z_k = \sigma\left(\sum_{j=1}^{J} u_{jk}y_j\right) = \sigma(u_k^T \cdot y), \quad \text{where } u_k^T = [u_{1k}, ..., u_{JK}]^T \tag{2.1.33}$$

Let substitute y_j in (2.1.33) to (2.1.32).

$$z_k = \sigma\left(\sum_{j=1}^{J} u_{jk} \cdot \sigma\left(\sum_{i=1}^{I} w_{ij}x_i\right)\right) \tag{2.1.34}$$

And then, with the target output, z_k^*, the cost for inputs can be written as

$$C_x = \sum_{k=1}^{K} (z_k - z_k^*)^2. \tag{2.1.35}$$

The cost, C_x, can be written with respect to parameters by substituting z_k of (2.1.35) to (2.1.33) and (2.1.34).

$$C_x = \sum_{k=1}^{K} \left[\sigma\left(\sum_{j=1}^{J} u_{jk} \cdot \sigma\left(\sum_{i=1}^{I} w_{ij} x_i \right) \right) - z_k^* \right]^2 \tag{2.1.36}$$

$$C_x = \sum_{k=1}^{K} [\sigma(u_k^T \cdot y) - z_k^*]^2 \tag{2.1.37}$$

From update rule of weights derived in (2.1.25), the partial derivatives need to be computed.

$$\frac{C_x}{u_{jk}} = 2[\sigma(u_k^T \cdot y) - z_k^*]\sigma'(u_k^T \cdot y) y_j \tag{2.1.38}$$

Derivative of sigmoid is directly determinable from function values.

$$\sigma'(z) = \sigma(z)(1 - \sigma(z)) \tag{2.1.39}$$

With using (2.1.39), the equation (2.1.39) can be rewritten as

$$\frac{C_x}{u_{jk}} = 2[\sigma(u_k^T \cdot y) - z_k^*]\sigma(u_k^T \cdot y)(1 - \sigma(u_k^T \cdot y)) y_j \tag{2.1.40}$$

$$= \underbrace{2[z_k - z_k^*] z_k (1 - z_k)}_{\delta_k} y_j. \tag{2.1.41}$$

Let define δ_k as error signal for k-th output in layer L, and then inspect how the error signal propagated in preceding layer with deriving partial derivatives of C_x with respect to w.

$$\frac{C_x}{w_{ij}} = 2[\sigma(u_k^T \cdot y) - z_k^*]\sigma'(u_k^T y) u_{jk} \sigma'(w_j^T) x_i \tag{2.1.42}$$

$$= x_i y_j (1 - y_j) \sum_{k=1}^{K} 2[z_k - z_k^*] z_k (1 - z_k) u_{jk} \tag{2.1.43}$$

$$= \underbrace{x_i \, y_j (1 - y_j) \sum_{k=1}^{K} \delta_k u_{jk}}_{\delta_j} = x_i \delta_j \tag{2.1.44}$$

From above relation between δ_k and δ_j, back- propagating error can be generalized to MLPs more than two layers. Let o_j be the output of j-th neuron in any layer, and assume ANN have $L > 2$ layers. Then error signal for all layers can be generally defined as below.

$$\delta_j = \begin{cases} o_j(1-o_j)(o_j - z_j^*) & \text{if } j \in \text{output layer } L \\ o_j(1-o_j)\sum\limits_{k\in m+1} \delta_k w_{jk} & \text{if } j \in \text{layer } m < L \end{cases} \tag{2.1.45}$$

Based on above generalization, update rule of weights between i-th neuron of current layer and j-th neuron of following layer (2.1.25) can be written with regard to back-propagating error.

$$w_{ij} \to w'_{ij} = w_{ij} - \eta\, o_i\, \delta_j \tag{2.1.46}$$

As shown in (2.1.45), the error in output layer is determined by the output of its neurons and target value (i.e. label). On the other hand, (2.1.45) also shows that the error of all non-output layer is defined with error of all neurons of subsequent layer and weights of associated connections. In stand point of errors, they are firstly determined at output neurons. And that computed errors in output layer are then used to calculate errors of the preceding layer, and the errors of current layer are again used to compute errors its preceding layer. Thus, the error is propagated backwards from output layer to first layer, and this error back-propagating algorithm is called backpropagation.

2.1.6 *Generalization and Regularization*

Until previous section 2.1.5, the learning algorithm only concerned to minimize training error (i.e. optimizing loss function). However, the reason why train machine learning model is to apply it for previously unseen inputs, and the learning algorithm discussed so far may not always guarantee optimized output for test samples. In such a case, Generalization and Regularization should be additionally considered to get reliable results in application.

Generalization

The ability of machine learning model to perform well on previously unobserved inputs is called generalization [GBC16]. Specifically, in addition to minimize training

error, another requirement for machine learning model such as ANN is to minimize the gap between training error and test error. The test error is also called generalization error, and it is defined as the expected value of the error on a new inputs [GBC16]. With regard to above two different goals, the model can be evaluated by underfitting and overfitting. Underfitting occurs when the model cannot sufficiently minimize training error [GBC16]. Overfitting occurs when the learning from training cannot sufficiently generalize to test. The relation between generalization and under/overfitting can be easily described by model capacity. In general, machine learning can be considered as to variate model capacity [GBC16]. Trained model with insufficient capacity are unable to solve complex tasks even in given labeled training samples. On the other hands, too high capacity with over-train can solve given specific complex task, but it may not proper model to solve the present task such as classifying previously unobserved patterns. As regards the model of ANN, underfitting can be rather easily avoided with increasing the number of neurons, and regularization techniques are widely used to prevent overfitting.

Regularization

"Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error" [GBC16]. According to the explanatory definition above, many approaches can be categorized as regularization technique. Among them, the subsection covers Data Augmentation, Weight Decay and Dropout.

(i) Data Augmentation

Training and generalization error vary as the size of the training set varies. Expected generalization error can never increase as the number of training examples increases [GBC16]. The larger number of labeled training samples make possible for learning model to reduce gap between training and generalization errors. However, in most cases in machine learning, the labeled training samples are scarce, and the number of those are limited. Data Augmentation is one of regularization techniques which increases the amount of training data using previously given original training data. Depending on specification of problems, various data augmentation techniques can be applied. One of the most intuitive examples is image augmentation for visual object classification tasks. Below figure, 2.1.8, shows augmentation of training word image labeled as "move". Let assume word images given in figure 2.1.8 are used for training word image classifier. Then the augmented image, (b) of figure 2.1.8, help classifier

(a) Original (b) Augmented

Figure 2.1.8: Example for image augmentation with affine transform.

train more various visual pattern of images for class "move". Thus, the additional learning based on augmented samples is obviously advantageous to classify unseen images of that class in test.

(ii) Weight Decay

Weight decay is also known as L2 regularization. The idea of the technique is to add regularization term to the loss function. Let unregularized and original function discussed in section 2.1.3 as C_0, then the regularized loss function can be written as below.

$$C = C_0 + \frac{\lambda}{2}\|w\|_2^2 = C_0 + \frac{\lambda}{2}w^T w \tag{2.1.47}$$

In regularization term, the squared magnitude of all weights, same as squared L2-norm, is scaled by a factor $\lambda/2$, where $\lambda > 0$ is known as the regularization parameter and $1/2$ is used for cancelling 2 appeared by gradient with respect to parameters w. The intuitive interpretation of L2 regularization is to make the ANN used to learn small weights by penalizing peaky weight vectors. In other words, it can be interpreted as a way of trade-off between finding diffuse weights and minimizing the original loss function. The comprise between original loss function and regularization term depends on the value of regularization parameter λ. The preference of small weights can be algebraically specified by generalized form of (2.1.25).

$$w \to w - \eta\frac{\partial C}{\partial w} \tag{2.1.48}$$

The derivative of regularized loss function (2.1.47) is as follow.

$$\frac{\partial C}{\partial w} = \frac{\partial C_0}{\partial w} + \lambda w \tag{2.1.49}$$

With substituting partial derivative term in (2.1.48) to (2.1.49), the update rule of weight with regularization can be written as below.

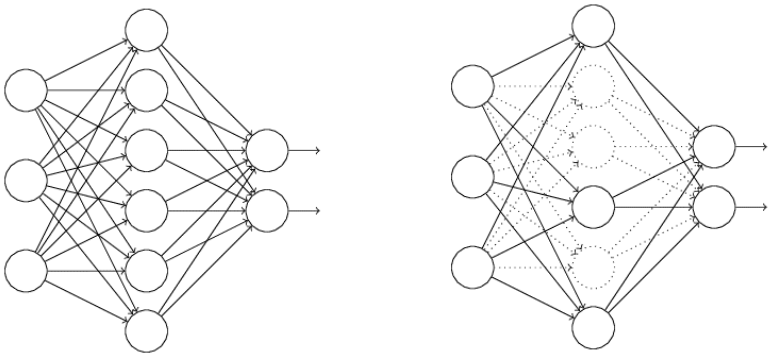
$$w \to w - \eta\left(\frac{\partial C_0}{\partial w} + \lambda w\right) \tag{2.1.50}$$

$$= (1 - \eta\lambda)w - \eta\frac{\partial C_0}{\partial w} \tag{2.1.51}$$

The rescaling factor $(1 - \eta\lambda)$ makes the weights smaller, and it is then referred to as weight decay. This suppressing weights help to avoid too high model capacity related to description of generalization in section 2.1.6. Hence, relatively simple models with weight decay tend to reduce overfitting [Nie15].

(iii) Dropout

Previously discussed regularization techniques are common for other machine learning as well as ANN. Different to them, dropout is a technique for reducing overfitting in neural networks by modifying the network itself. In each iteration of training, specified ratio of randomly selected hidden neurons in the network are temporarily deleted. In particular, dropout technique is same as to train several different networks which have same number but different combination of neurons in hidden layer. And then, the output is decided by using some kind of averaging or voting scheme for outputs of large number of different networks. Below figure 2.1.9 shows a iteration of dropout for one hidden layer MLPs with ratio 0.5.



(a) no dropout (b) dropout 0.5

Figure 2.1.9: Example for dropout [Nie15].

In terms of classification task, the dropout also means that forced to learn more robust features that are useful in conjunction with many different random subsets of the other neurons [KSH]. In summary, dropout procedure with large iteration is same as to average the effects of a very large number of different networks. Consequently, the dropout technique which reflects different ways of overfit from different networks will be to reduce oveffiting[Nie15].

2.1.7 *Convolutional Neural Networks*

A shallow network with one hidden layer discussed in section 2.1.1 shows theoretically ideal capability in complex pattern classification. However, in practice, a deeper network is advantageous in learning to solve complex visual pattern recognition problem. Such network can use intermediate layers to build up multiple level of abstraction. For instance, the neurons in the first hidden layer can be used to recognize edges of given visual images, and the neurons of second hidden layer can learn to recognize region or contour built up from edges. Even in the case of more than two hidden layers, the following hidden layers until output layer can be used to learn more abstracted complex shapes. Hence, Deeper networks promise enhanced capabilities in visual pattern recognition. However, those networks have much larger amount of parameters than shallow networks. Training of such large amount of parameters was not feasible for many years because of several limitations such as computational power, vanishing gradient and overfitting problem. After several years has passed, the advent and wide use of General-Purpose computing on Graphics Processing Units (GPGPU) enabled massive parallel processing, and it made possible to train huge amount of parameters. Different to the physical limitation, the vanishing gradient problem is inherited from the error back propagation algorithm and small upper bound of the gradient value of a sigmoid or hyperbolic tangent activation function. The problem was moderately addressed by using Rectified Linear Units (ReLU) non-linear activation function. Even though parallel computation with GPGPU has been used for training, deeper network still had large amount of parameters to be trained if all the layers are fully connected like as shallow networks discussed before. The special architecture network called Convolution Neural Network (CNN) make possible to shrink the number of independent parameters to be trained and avoid overfitting problem. CNN is one of the most widely used types of deep network[Nie15]. The origins of convolutional neural networks go back to the 1970s[Nie15]. But the seminal paper establishing the modern subject of convolutional networks was a 1998 paper [LBBH98b] [Nie15]. In the paper, CNN is applied to MNIST dataset for handwritten

digits recognition. And CNNs use three basic ideas: local receptive fields, shared weights and pooling[Nie15].

(i) Local receptive fields

The first architectural characteristics of CNN is to use local receptive fields for connecting a small region of input image in previous layer to each neuron in following hidden layer. In convolutional network terminology, a small window is often referred to as the kernel. The output, kernel responses of input image, is referred to as the feature map. And overlapped region in the input image by the kernel is called the local receptive field for the hidden neuron. In CNN, as shown in its name, the operation of the local receptive field is convolution. Let two-dimensional input image as I and kernel as K, then the algebraic expression of convolution of spatial position x,y in the feature map, $S(x, y)$, can be written as (2.1.52).

$$S(x, y) = (K * I)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I(x - s, y - t)K(s, t) \tag{2.1.52}$$

In above, the kernel size is $m \times n$ with $m = 2a + 1$ and $n = 2b + 1$. And the kernel is assumed to be flipped relative to input. For example, as for visual pattern recognition, the neurons of input layer is depicted as 28×28 with pixel intensities of image instead of vertical line of neurons as usual in MLPs. The local receptive field with 5×5 kernel slides across the entire input image with starting from its top-left corner. Then the local receptive field slides over by one pixel to the right, to connect to the second neuron from the left corner in first hidden layer. And so on,the first hidden layer will be build up with 24×24 neurons. The example is concretely illustrated in figure 2.1.10. The idea of using local receptive field is also referred to as sparse interactions to distinguish it from fully connection between traditional neural network layers.

(ii) Shared weights

As described in (2.1.52), each unit of $m \times n$ kernel has each different weight with respect to its spatial position s, t. In other words, each hidden neuron has a bias and $m \times n$ weights connected to its local receptive field. The idea of shared weights is to use the same weights and bias for each of the hidden neurons. For that, let suppose σ is the neural activation function. b is the shared value for the bias. $w_{s,t}$ is a $m \times n$

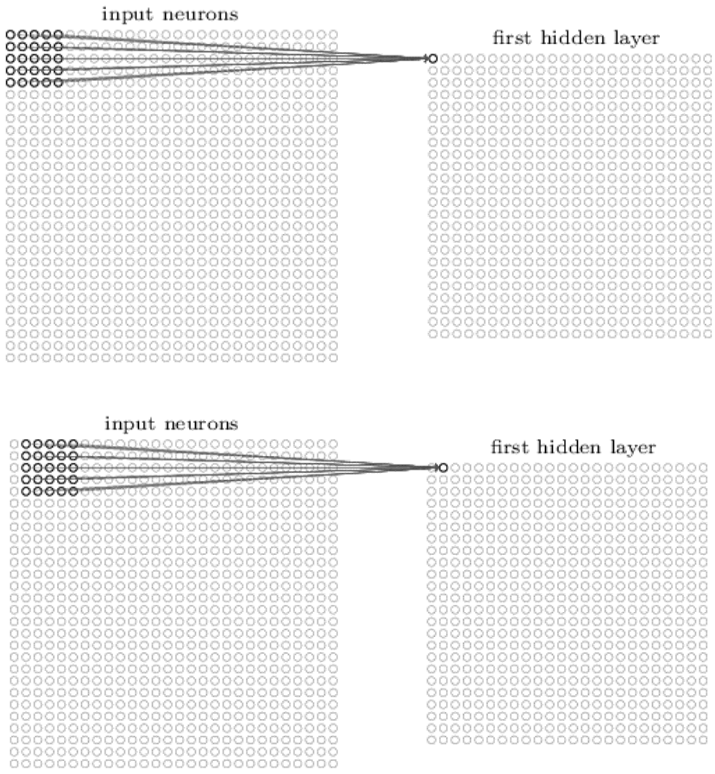


Figure 2.1.10: Illustration of sliding local receptive field over by one pixel from top-left corner (top) to the right (bottom) [Nie15].

array of shared weights. And, similar to (2.1.4), the output of each neuron of spatial position x, y can then be written as below with convolution operation (2.1.52).

$$\sigma\left(\sum_s \sum_t I_{x-s,y-t} \cdot w_{s,t} + b\right) \tag{2.1.53}$$

The response of the local receptive fields with shared weights is same notion for linear filter response of image filtering process in typical computer vision. So, it means that all the neurons in the hidden layer detect exactly the same visual feature for each feature map, just at different spatial position in the image. The illustrations in figure 2.1.10 can detect just a single kind of localized feature with one feature map. However, a number of localized features are needed to visual pattern recognition, so a convolutional layer typically consists of several different feature maps as shown

in figure 2.1.11. The advantages of sharing weights and biases not only capturing

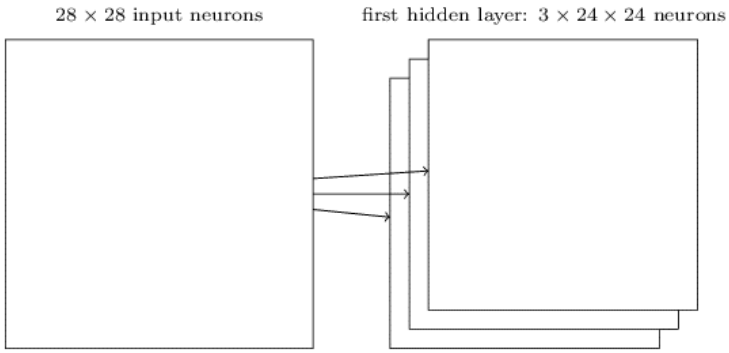


Figure 2.1.11: Illustration of 3 different feature maps with each different 5×5 shared weights [Nie15].

localized feature but greatly reducing the number of parameters. In the case of example in figure 2.1.11, with $28 \times 28 = 784$ input neurons, it assumed 5×5 shared weights and a single shared bias for each feature map. Each feature map requires 26 parameters, and the convolutional layer is then defined by a total of $26 \times 3 = 78$ parameters. On the other hand, let assume a fully connected layer with relatively modest 5 hidden neurons instead of convolutional layer for comparison. It requires a total of 784×5 weights, plus an extra 5 biases, overall $3,925$ parameters. Hence, even in this example, the fully connected layer would have more than 50 times as many parameters as the convolutional layer. For sure, the example cannot exactly represent a direct comparison between the number of parameters, because the convolutional and fully connect layer are different in essential ways. It is clear that the sharing weights with convolutional layer would reduce the number of parameters compared to fully-connected layer which results in the same performance especially for large networks. The reduced number of parameters, in turn, make possible the deep networks to be trained with using convolutional layers.

(iii) Pooling

A convolutional network typically contains a set of three layers in a row. the first layer, convolutional layer, performs feature mapping with local receptive fields. In the following layer called ReLU layer, each linear activation is run through the rectified linear activation function. Lastly, in the pooling layer, a pooling function is applied to modify the output of the nonlinear activation. A pooling function replaces the

output of the net at a certain location with a summary statistic of the nearby outputs [GBC16]. The max pooling operation, one of the most widely used pooling function, reports the maximum output within a rectangular neighbourhood and prepares a condensed feature map. In a concrete example, a pooling unit simply outputs the maximum activation in the 2×2 input region, as illustrated in the following figure. Note that since the example shown in figure 2.1.12 have 24×24 neurons output from

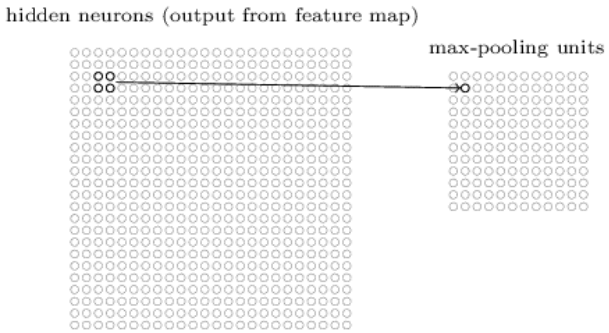


Figure 2.1.12: Illustration of max-pooling with a region of 2×2 neurons [Nie15].

the convolutional layer, after 2×2 max-pooling it has 12×12 neurons for each feature. Finally, the convolutional layer mentioned above usually involves more than a single

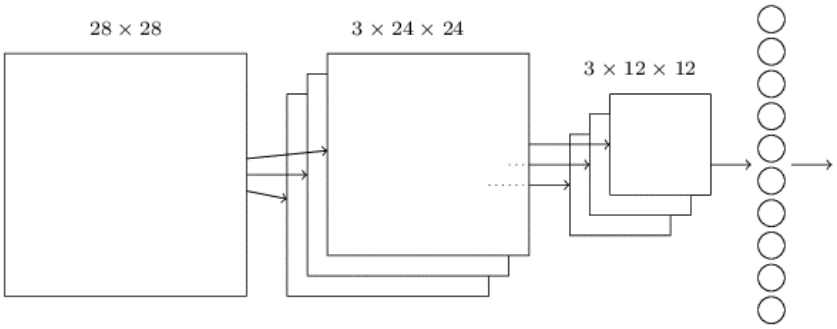


Figure 2.1.13: Illustration of convolutional, pooling and fully connected layer [Nie15].

feature map, and max-pooling is applied to each feature map separately. So, the intuitive interpretation is that once a feature has been detected by linear activation (i.e. convolution), the convolutional networks efficiently take care of its rough location relative to other features instead of the exact positional information. This pooling

offers a crucial benefit which reduces the number of parameters needed in later layers. For example, in figure 2.1.13, total $3 \times 12 \times 12 = 432$ neurons are fully-connected to 10 neurons in the last layer. If the network does not involve pooling layer, then a larger number of parameters are required for $3 \times 24 \times 24 = 1,728$ neurons to fully-connect to 10 neurons.

2.2 CANONICAL CORRELATION ANALYSIS (CCA)

In a scientific process, the object or the data can be described by two sets of variables corresponding two different views. The analysis of the relations between those two representations resulted from two views may improve the understanding of the underlying process. For example in the field of visual object recognition with machine learning, one variables can be comprised by feature extraction of humans' understanding. The another variable can be resulted from the machine learning model. Identifying the relations between two feature representations which are extracted from same visual object can improve the understanding of machine learning model and its output. Additionally, the relation could provide new information about the functioning of the system. Canonical Correlation Analysis (CCA) is the method that determines and analyses the relation described above. CCA was first introduced by H.Hotelling [Hot36] as a method for finding relationships between variables, and first applied in an economics study [Wau42]. Recently, the applicability of CCA has been demonstrated in modern fields of science such as neuroscience [LCJ15], machine learning [XBSY13] and bioinformatics [JRFR13]. The section begins with an introduction to the principles of CCA. After introducing principles, the following subsection formulates and solves CCA problem with mathematics. Additionally, one extension of CCA to under-determined setting, regularised CCA, is discussed [UMK+17].

2.2.1 *The Basic Principles of CCA*

CCA is a method for finding linear relations between two multidimensional datasets. The principle of CCA is to find position vectors which have minimally enclosed angle between their unit norm vector images in the new coordinate system after linear transformation with two different datasets respectively. In other words, CCA is to find two positions in the two data spaces respectively that have images on a unit ball such that the angle between them is minimised [UMK+17]. The following notation can be used to derive more specific illustration of principle. Two different views of the data

comprise multiple variables on a set of observations. Because the thesis only covers the topic of visual object recognition, let the set of variables are the attributes (i.e. feature) representation of the visual objects and be denoted by A for the further use of these notations. And then, the datasets of different views x and y are denoted by the matrices $A_x \in \mathbb{R}^{n \times p}$ and $A_y \in \mathbb{R}^{n \times q}$ respectively where n represents the number of observations and p and q are defined by the dimensions of two different representation of features. Throughout the thesis, all the description about CCA assume that the variables are standardised to zero mean and unit variance. As mentioned above, the goal of CCA is to analyse the linear relation between the variables of A_x and A_y based on linear transformation. For the aim, let the position vectors and their images are denoted by $w_x \in \mathbb{R}^p$ and $w_y \in \mathbb{R}^q$ and $z_x \in \mathbb{R}^n$ and $z_y \in \mathbb{R}^n$ respectively. The positions are also referred to as canonical weight vectors, and the images are typically termed as canonical components. With the notations above, the linear transformations can be algebraically expressed as below.

$$A_x w_x = z_x \quad , \quad A_y w_y = z_y \tag{2.2.1}$$

The data matrices A_x and A_y represent linear transformations of the positions w_x and w_y onto the images z_x and z_y [UMK+17]. In other words, the canonical components, z_x and z_y, are the vectors in the linear subspace spanned by column vectors of A_x and A_y respectively. There are two constraints of these mapping of CCA. The first constraint is that canonical components are unit norm vectors. And the second one is that the enclosing angle between them, $\theta \in [0, \pi/2]$, should be minimised [GZ95]. For the enclosing angle, the minimised angle is same as maximised cosine value. Therefore, two constraints can be formulated as below.

$$\max \frac{< z_x, z_y >}{\|z_x\|_2 \|z_y\|_2} = \max < z_x, z_y > = \max \operatorname{corr}(z_x, z_y) \tag{2.2.2}$$

$$\|z_x\|_2 = 1, \ \|z_y\|_2 = 1 \tag{2.2.3}$$

The minimised enclosing angle constraint is also referred to as the canonical correlation, because it is same for maximizing correlation between unit norm canonical components. Hence in summary with above notations, the principle of CCA is to find weight vectors, w_x and w_y, that are mapped onto an n-dimensional unit ball after the linear transformations A_x and A_y with the constrains of their images, $\max < z_x, z_y >$ [UMK+17].

2.2.2 *Mathematical Definitions of CCA*

This subsection formulates CCA problem based on the principles and notations introduced above, and solve that through eigenvalue problem. Let the smallest angle is denoted by θ_1 resulted from the images z_x and z_y. And then, it determines the first canonical correlation, $cos\theta_1$, with unit norm constraint of images.

$$\begin{aligned} &objective \ : cos\theta_1 = \max < z_x, z_y > \\ &constraints \ : \|z_x\|_2 = 1, \ \|z_y\|_2 = 1 \end{aligned} \tag{2.2.4}$$

Let the maximum above is obtained by first pair of canonical components, z_x^1 and z_y^1. The second smallest angle, θ_2, is found by z_x^2 and z_y^2 that are orthogonal to each preceding canonical components respectively. The procedure is continued until the number of pairs reaches pre-defined number $d \leqslant \min(p, q)$. This number of canonical correlations, d, corresponds to the dimensionality of CCA, and it should be less than or equal to smaller dimension of two variables because it determines the number of patterns which can be extracted from given two datasets. Hence (2.2.4) can be recursively generalised to below expression.

$$objective \ : cos\theta_d = \max < z_x^d, z_y^d > \tag{2.2.5}$$

$$constraints \begin{cases} \|z_x^d\|_2 = 1, \ \|z_y^d\|_2 = 1 & \text{(unit norm)} \\ < z_x^d, z_x^r >= 0, < z_y^d, z_y^r >= 0, \ \forall d \neq r & \text{(orthogonal)} \end{cases} \tag{2.2.6}$$

Theoretically, CCA is solved iteratively by finding d number of canonical components pair. However, in practice, the large dimensionality of CCA for extracting enough pattern from data requires intensive computing. For this reason, instead of solving it iteratively, H.Hotelling proposed solving a standard eigenvalue problem as eigenvalue-based methods for CCA [Hot36]. Alternatively, a generalised eigenvalue problem approach is widely adapted for computing libraries of CCA [BJ02]. In the technique of solving CCA with eigen-value based method, the formulation focuses on finding the positions w_x and w_y at first, and obtaining the components z_x and z_y based on their

linear relation (2.2.1). The objective and constraints, (2.2.4), can then be expanded to (2.2.7) with respect to weight, w, and data, A.

$$\begin{aligned} &< z_x, z_y >=< A_x w_x, A_y w_y >= w_x^T A_x^T A_y w_y \\ &\|z_x\|_2 = z_x^T z_x = w_x^T A_x^T A_x w_x = 1 \\ &\|z_y\|_2 = z_y^T z_y = w_y^T A_y^T A_y w_y = 1 \end{aligned} \tag{2.2.7}$$

To make convenient to formulate CCA as eigenvalue problem, the multiplication of data matrices can be rewritten in terms of sample covariance matrices C_{xy} and C_{yx} and the empirical variance matrices C_{xx} and C_{yy}. Let each variables is represented by column vector of data matrix, a.

$$A_x = \begin{pmatrix} | & & | \\ a_{x,1} & \cdots & a_{x,p} \\ | & & | \end{pmatrix}, \quad A_y = \begin{pmatrix} | & & | \\ a_{y,1} & \cdots & a_{y,q} \\ | & & | \end{pmatrix} \tag{2.2.8}$$

Then, the sample covariance matrix C_{xy} is defined by $\frac{1}{n-1} A_x^T A_y$ as derived in (2.2.9).

$$\begin{aligned} C_{xy} &= \begin{pmatrix} cov(a_{x,1}, a_{y,1}) & \cdots & cov(a_{x,1}, a_{y,q}) \\ \vdots & \ddots & \vdots \\ cov(a_{x,p}, a_{y,1}) & \cdots & cov(a_{x,p}, a_{y,q}) \end{pmatrix} \\ &= \frac{1}{n-1} \begin{pmatrix} a_{x,1}^T a_{y,1} & \cdots & a_{x,1}^T a_{y,q} \\ \vdots & \ddots & \vdots \\ a_{x,p}^T a_{y,1} & \cdots & a_{x,p}^T a_{y,q} \end{pmatrix} \\ &= \frac{1}{n-1} A_x^T A_y \end{aligned} \tag{2.2.9}$$

And the empirical variance matrices C_{xx} and C_{yy} are given by $\frac{1}{n-1}A_x^T A_x$ and $\frac{1}{n-1}A_y^T A_y$ respectively based on (2.2.10).

$$\begin{aligned} C_{xx} &= \begin{pmatrix} cov(a_{x,1}, a_{x,1}) & \cdots & cov(a_{x,1}, a_{x,p}) \\ \vdots & \ddots & \vdots \\ cov(a_{x,p}, a_{x,1}) & \cdots & cov(a_{x,p}, a_{x,p}) \end{pmatrix} \\ &= \frac{1}{n-1}\begin{pmatrix} a_{x,1}^T a_{x,1} & \cdots & a_{x,1}^T a_{x,p} \\ \vdots & \ddots & \vdots \\ a_{x,p}^T a_{x,1} & \cdots & a_{x,p}^T a_{x,p} \end{pmatrix} \\ &= \frac{1}{n-1}A_x^T A_x \end{aligned} \tag{2.2.10}$$

Now, previously formulated CCA problem (2.2.7) can be rewritten by (2.2.11) in terms of C_{xy}, C_{xx} and C_{yy}.

$$\begin{aligned} &< z_x, z_y >=< A_x w_x, A_y w_y >= w_x^T A_x^T A_y w_y = w_x^T C_{xy} w_y \\ &\|z_x\|_2 = z_x^T z_x = w_x^T A_x^T A_x w_x = w_x^T C_{xx} w_x = 1 \\ &\|z_y\|_2 = z_y^T z_y = w_y^T A_y^T A_y w_y = w_y^T C_{yy} w_y = 1 \end{aligned} \tag{2.2.11}$$

In (2.2.11), the factor $\frac{1}{n-1}$ can be neglected, because the correlation between z_x and z_y does not change with their scaling, because the factor is always cancelled.

$$corr(z_x, z_y) = \frac{< z_x, z_y >}{\|z_x\|_2 \|z_x\|_2} = \frac{\frac{1}{n-1}}{\sqrt{\frac{1}{n-1}}\sqrt{\frac{1}{n-1}}} \cdot \frac{w_x^T C_{xy} w_y}{\sqrt{w_x^T C_{xx} w_x}\sqrt{w_y^T C_{yy} w_y}} \tag{2.2.12}$$

Substituting (2.2.11) into the defined problem (2.2.4), the objective and constraints can be summarized as shown below.

$$\begin{aligned} &objective \ : \ max\,(w_x^T C_{xy} w_y) \\ &constraints \ : \ w_x^T C_{xx} w_x - 1 = 0, \quad w_y^T C_{yy} w_y - 1 = 0 \end{aligned} \tag{2.2.13}$$

Without loss of generality for unit norm representation, the constraints are expressed in squared form . Finally, the Lagrange function (2.2.14) can be obtained, and solved using the Lagrange multiplier technique.

$$L = w_x^T C_{xy} w_y - \frac{\rho_1}{2}(w_x^T C_{xx} w_x - 1) - \frac{\rho_2}{2}(w_y^T C_{yy} w_y - 1) \tag{2.2.14}$$

Two Lagrange multipliers are denoted by ρ_1 and ρ_2 respectively, but $\rho_1 = \rho_2 = \rho$ since $w_x^T C_{xx} w_x = 1$ and $w_y^T C_{yy} w_y = 1$. The goal of solving CCA problem is to find position vectors, w_x and w_y. For the purpose, the algebraic expression (2.2.14) is differentiated with respect to w_x and w_y.

$$\frac{\partial L}{\partial w_x} = C_{xy} w_y - \rho C_{xx} w_x = 0, \tag{2.2.15}$$

$$\frac{\partial L}{\partial w_y} = C_{yx} w_x - \rho C_{yy} w_y = 0 \tag{2.2.16}$$

The problem comprises two equation for two unknown. The equation (2.2.14) can be written by (2.2.17).

$$w_x = \frac{C_{xx}^{-1} C_{xy}}{\rho} w_y \tag{2.2.17}$$

With substituting w_x of (2.2.17) into (2.2.16), the equation is expressed merely about w_y.

$$C_{yx} C_{xx}^{-1} C_{xy} w_y = \rho^2 C_{yy} w_y \tag{2.2.18}$$

If the sample covariance matrix C_{yy} of above is non-singular, then the problem is same as to solve below standard eigenvalue problem.

$$C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy} w_y = \rho^2 w_y \tag{2.2.19}$$

The square roots of eigenvalues of the matrix $C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy}$ correspond to the canonical correlations, and those are found by solving below characteristic equation.

$$|C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy} - \rho^2 I| = 0 \tag{2.2.20}$$

After finding eigenvalues by solving above characteristic equation (2.2.20), the position vector, w_b, can be obtained by computing eigenvectors which satisfy the equation.

$$(C_{yy}^{-1}C_{yx}C_{xx}^{-1}C_{xy} - \rho^2 I)w_y = 0 \tag{2.2.21}$$

Since the relation between two position vectors, w_a and w_b are already known, the another weight vectors w_a can be simply computed based on algebraic expression (2.2.17). On the other hand, the Lagrange function can also be formulated and solved through the generalised eigenvalue problem. In here, instead of finding corresponding pairs of position vectors separately, they are formulated and founded at once from the simultaneous representation of the equations, (2.2.15) and (2.2.16).

$$\begin{aligned} C_{xy}w_y &= \rho C_{xx}w_x \\ C_{yx}w_x &= \rho C_{yy}w_y \end{aligned} \tag{2.2.22}$$

And above simultaneous equations can be represented as a generalised eigenvalue problem of the form $\beta L w = \alpha R w$ where the pair (β, α) is an eigenvalue of the pair (L, R) [Saa11] [UMK+17].

$$\begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \rho \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} \tag{2.2.23}$$

Specifically, the generalised eigenvalue problem can be expressed in the below form.

$$\begin{aligned} &Lw = \rho Rw \\ &\text{where} \quad L = \begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix}, R = \begin{pmatrix} C_{xx} & 0 \\ 0 & C_{yy} \end{pmatrix}, w = \begin{pmatrix} w_x \\ w_y \end{pmatrix} \end{aligned} \tag{2.2.24}$$

Let p is lower than q, then the generalised eigenvalues comprises in pairs.

$$\rho = \{0, \rho_1, -\rho_1, \rho_2, -\rho_2, ..., \rho_{p-1}, -\rho_{p-1}, \rho_p, -\rho_p\} \tag{2.2.25}$$

The generalised eigenvectors which is position vectors w_x and w_y are computed from the corresponding positive generalised eigenvalues.

In summary, CCA is to find two position vectors w_x and w_y having images z_x and z_y on a unit ball such that the angle between images is minimised after linear transformations A_x and A_y respectively. In practice, it is computed by applying

eigenvalue-based methods, and the eigenvalues are canonical correlations and the eigenvectors are the position vectors that want to compute by solving CCA.

2.2.3 *Regularised CCA*

The objective of CCA discussed in section 2.2.1 and 2.2.2 is to extract linear relations in overdetermined datasets. In other words, standard CCA assumes that the number of observations, n, exceeds the number of variables, p and q, in either dataset. This assumption of overdetermined setting enables to guarantee the non-singularity of the covariance matrices, C_{xx} and C_{yy} in (2.2.17), (2.2.18) and (2.2.19) of section 2.2.2. However, when the sample size is insufficient or many of the variables are co-linear, then CCA is ill-posed in underdetermined setting. It can be algebraically expressed by the notation used in section 2.2.1.

$$n < min(p, q) \tag{2.2.26}$$

If the number of observations n less than the number of variables, p and q, then the eigenvalue problem cannot be solved without regularisation, because (2.2.26) hurts the invertibility of the covariance matrices. The regularisation technique for CCA in underdetermined setting (2.2.26) was proposed in the work of [Vin76]. The idea is to add arbitrary constants, $\lambda_1 > 0$ and $\lambda_2 > 0$, to the diagonal elements of covariance matrices, $C_{xx} + \lambda_1 I$ and $C_{yy} + \lambda_2 I$, respectively for improving the invertibility of C_{xx} and C_{yy}. With considering regularisation, the optimisation problem of CCA (2.2.13) can be rewritten by below.

$$\begin{aligned} &objective \ : \ max\,(w_x^T C_{xy} w_y) \\ &constraints \ : \ w_x^T (C_{xx} + \lambda_1 I) w_x - 1 = 0, \quad w_y^T (C_{yy} + \lambda_2 I) w_y - 1 = 0 \end{aligned} \tag{2.2.27}$$

After applying Lagrange multiplier technique similar to the mathematical steps in section 2.2.2, the CCA problem reduces to below standard eigenvalue problem.

$$(C_{yy} + \lambda_2 I)^{-1} C_{yx} (C_{xx} + \lambda_1 I)^{-1} C_{xy} w_y = \rho^2 w_y \tag{2.2.28}$$

Similarly, the eigenvalues are obtained by solving below characteristic equation.

$$|(C_{yy} + \lambda_2 I)^{-1} C_{yx} (C_{xx} + \lambda_1 I)^{-1} C_{xy} - \rho^2 I| = 0 \tag{2.2.29}$$

Alternatively, the position w_x and w_y also can be found by solving the generalised eigenvalue problem like as (2.2.23).

$$\begin{pmatrix} 0 & C_{xy} \\ C_{yx} & 0 \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} = \rho \begin{pmatrix} C_{xx} + \lambda_1 I & 0 \\ 0 & C_{yy} + \lambda_2 I \end{pmatrix} \begin{pmatrix} w_x \\ w_y \end{pmatrix} \tag{2.2.30}$$

This regularization technique enables to avoid singular problem of covariance matrices, but it inevitably introduces non-negative regularisation parameters $\lambda_1 > 0$ and $\lambda_2 > 0$. And choosing the parameters likely affects to the optimality of the solution of CCA, such that cannot resolve ill-posed condition with too small parameters or the covariance matrices are dominated by λI with too large parameters. A cross-validation procedure can be used for automatically selecting the optimal regularisation parameters in CCA. This regularisation technique is firstly proposed in [Lar31], and it is a well-established non-parametric model selection procedure to evaluate the validation of statistical predictions. In general, the procedure starts from partitioning the observations into the subsamples. And then, the training of CCA, estimating a statistic, is first occurred on one subsample. The other hold-out subsamples are utilized for the validation purpose. The cross-validation approach specifically developed for CCA has been further extended in [SWZ08] [UMK+17]. When the sample size is too small and not enough to split the data into several folds, then a leave-one-out cross-validation procedure is an option. However, in most cases the sample size is large enough to partition the observations into k-folds, and each fold is then used for estimating parameters in turn until the iteration of folds covers all the samples. This iterative approach is known as k-fold crossvalidation, and usually the procedure is repeated until an optimal set of parameters are searched for. Algorithm 1 outlines k-fold cross-validation to determine the optimal regularisation parameters, λ_1 and λ_2, in CCA [UMK+17].

Regularised CCA is introduced to resolve singularity of covariance matrices in under-determined conditions such as more variables than observations and excessive colinearity of variables. The regularisation procedure is to find proper non-negative scalar parameters such that result in optimal solution of CCA problem by improving invertibility of the covariance matrices under eigenvalue-based methods.

Algorithm 1 : Repeated k-fold cross-validation for regularised CCA [UMK+17].

Input: Data matrices A_x and A_y, number of repetitions R, number of folds F
Output: Regularisation parameter values λ_1 and λ_2 maximising the correlation on test data

Pre-defined ranges for values of λ_1; λ_2;
Initialise $r = 1$;

repeat
  Randomly partition the observations into F folds;
  for all values of λ_1 **do**
    for all values of λ_2 **do**
      for $i = 1, 2..., F$ **do**
        Training set: $F - i$ folds, test set: i fold;
        Standardise the variables in the training and test sets;
        For the training data, solve $|(C_{yy} + \lambda_2 I)^{-1} C_{yx} (C_{xx} + \lambda_1 I)^{-1} C_{xy} - \rho^2 I| = 0$;
        Find w_y corresponding to the greatest eigenvalue satisfying $((C_{yy} + \lambda_2 I)^{-1} C_{yx} (C_{xx} + \lambda_1 I)^{-1} C_{xy} - \rho^2 I) w_y = 0$;
        Find w_x using $w_x^1 = \frac{(C_{xx} + \lambda_1 I)^{-1} C_{xy} w_y^1}{\rho^1}$;
        Transform the training positions w_x and w_y using the test observations $A_{x,test} w_x = z_x$ and $A_{y,test} w_y = z_y$;
        Compute $\cos(z_x, z_y) = \frac{<z_x, z_y>}{\|z_x\|_2 \|z_y\|_2}$;
      end for
      Store the mean of the F values for $\cos(z_x, z_y)$ obtained at λ_1 and λ_2;
    end for
  end for
until $r = R$;

Compute the mean of the R values for $\cos(z_x, z_y)$ obtained at λ_1 and λ_2;
Return the combination λ_1 and λ_2 that maximises $\cos(z_x, z_y)$

3

RELATED WORK

The handwriting word recognition, the work presented in this thesis, conventionally employs machine learning techniques with assuming the standard supervised learning setting. This means manually labeled training samples for each class are required for learning task, and the number of them highly affects classification accuracy. Although the requirement, preparing large number of labeled samples per class, is obvious, it is hard to meet such a requirement becuase the handwriting samples are often very scarce and it is very exhaustive to label the handwriting images manually in the case of multi-class classification with large number of classes. For example, a typical desk dictionary may define about 100,000 english vocabulary items [ACW14]. And let assume the recognition system requires at least 10 samples per class to cover the differences in writing styles, then overall 1,000,000 samples are needed to achieve good classification accuracy for unspecified test set. Obviously, one million different samples are not plausible amount to be manually labeled and to be sufficiently trained by system. Therefore, in general, it is plausible for recognition system to be given only a description of the classes without labeled training samples. The machine learning tasks in this environment is called zero-data learning [LEB08]. Since the handwriting word recognition can be seen as a special case of zero-data classification, the approaches in zero-data learning have begun to be used to classify handwriting word image in recent years

In computer vision, one of the reliable and efficient approach for the object recognition including word recognition is the Bag of Features (BoF) methods with machine

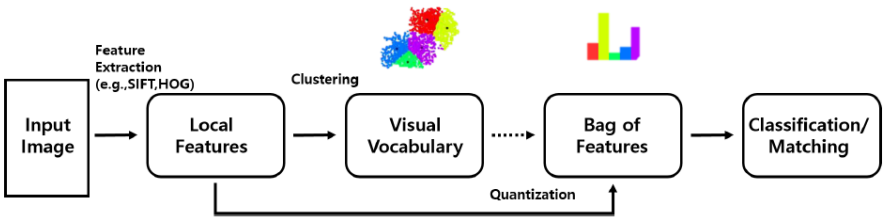


Figure 3.0.1: Processing pipeline of Bag of Features (BoF)-Approach.

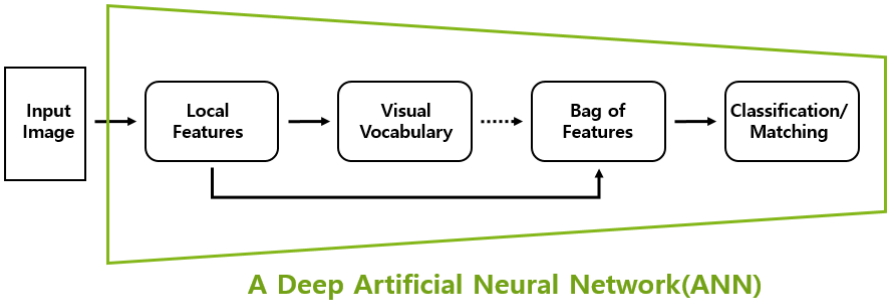


Figure 3.0.2: Computability of logical functions of perceptron.

learning techniques. The method comprises processing pipeline as shown figure 3.0.1. In general, the low-level features of the images such as SIFT or HoG were extracted. And then, visual words and its histogram are computed. At the last step, the machine learning techniques such as K-Nearest Neighbors algorithm (k-NN) or Support Vector Machine (SVM) are typically adapted for classification tasks. Recently, the ANN, discussed in section 2.1, is widely used as an alternative of above step-by-step approach as shown in figure 3.0.2. The use of ANN offers the advantages of integrated framework and synthetical optimization.

The chapter mainly introduces attribute-based classification [LNH14] to tackle aforementioned zero-data problem in handwriting word recognition. It begin with the outline of attribute-based classification, and discusses about semantic attributes as well as introduces a special attribute representation of words called Pyramidal Histogram of Characters (PHOC) [AGFV14]. After that, the second part of the chapter introduces two different deep CNNs [SF16] [PW16] which are designed for extracting attribute scores from word images. At the end of the chapter, an another kind of attribute based classifier called Direct Attribute Prediction (DAP) and its probabilistic realization [LNH14] are explained.

3.1 ATTRIBUTE-BASED CLASSIFICATION

Attribute-based classification is a way of zero-data learning in the scope of recognizing classes of patterns or objects. Zero-data learning can be defined as a machine learning process when no labeled training data are available for several classes (i.e. zero-data classes) but the descriptions of those classes are given. The idea of realization of such learning is to effectively utilize the descriptions of the classes in order to compensate

Figure 3.0.3: Non-discriminative attribute representation of words.

the lack of labeled data. This idea is originated from the capability of human learning which can identify new classes only with its description. For example, from the phrase "an animal included in horse family with distinctive black and white striped coat.", people may reliably identify zebra if they can distinguish different colors and shapes on the images even if they have never seen an image of zebra with label before. From the above example, a set of semantically meaningful high-level properties such as black and white stripes derived from its description called attributes. In summary, attribute-based classification is to identify and to classify objects based on semantic features called attributes that are shared by classes to distinguish them based on the presence or absence of attribute

Attributes of words

As clarified above, an attribute is a distinctive semantic property of an object. So, one of the most intuitive attributes of word might be the presence of characters. However, these simplest attributes are not an word discriminative representation, because different sequence of same set of characters possibly comes out as different word. For example, the words, "tame" and "mate", would have same attributes representation in this case. Therefore, suitable attributes of words should encode not only presence of character but its spatial position to distinguish the words which contain same set of characters. The attribute representation satisfied above requirements was introduced by Almazan et al. in [AGFV14], and it was dubbed as Pyramidal Histogram of Characters (PHOC) by them. As shown in its terminology, the attributes are represented as binary histogram based on estimating whether certain fraction of word contains certain characters or not. The term pyramidal in the above context represents level-wise approach in the way of splitting word. Specifically, the spatial split of word is determined by level, i.e., the word is split into equal n parts in the level n. For example, contained characters in the first half of word determines one histogram in level 2, and characters of second half of word determines another histogram. Therefore, with adapting enough high level, the concatenation of each histogram throughout entire levels can result in word discriminative representation. In addition, the linguistic

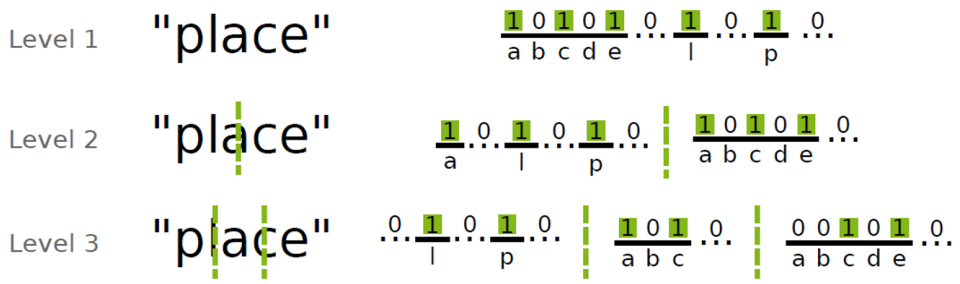


Figure 3.1.1: Example of three level PHOC representation [SF16].

point of view, two or three consecutive character may help distinguish different words. These one, two and three consecutive character attributes are called unigram, bigram and trigram respectively. However, the previous works, [SF17] and [PW16], already shown that the use of bigram and trigram does not lead to meaningful improvement in word spotting or word recognition tasks, even though it increases the dimensionality of PHOC representation.

3.2 DEEP CNNS FOR ATTRIBUTE PREDICTION

While the transformation from transcription of a word into attribute is trivial, that from an image to an estimated representation of attributes may be learned with modern machine learning techniques. In document image analysis, such as word spotting and word recognition, AttributeSVMs were adapted to estimate a binary attribute representation from a word image in the work of Almazán et al. [AGFV14]. A more recent work, [SF18] and [PW16], uses a deep CNN as integrated framework for estimating attribute instead of deciding vector encoding method and classification algorithm separately, and the work of Sudholt et el. [SF18] showed that learning attribute representations with PHOCNet results in state-of-the-art performances for segmentation-based word spotting in Handwritten Documents [PZG+16]. Another specially designed deep CNN estimating attributes was proposed by Poznanski et al. [PW16], and they dubbed it as CNN-N-Gram. The network is used for estimating PHOC attributes from handwritten word image in the process of segmentation-based word recognition.

Based on the outperforming results of the PHOCNet in segmentation-based word-spotting, the PHOCNet is used for segmentation-based word recognition in this thesis.

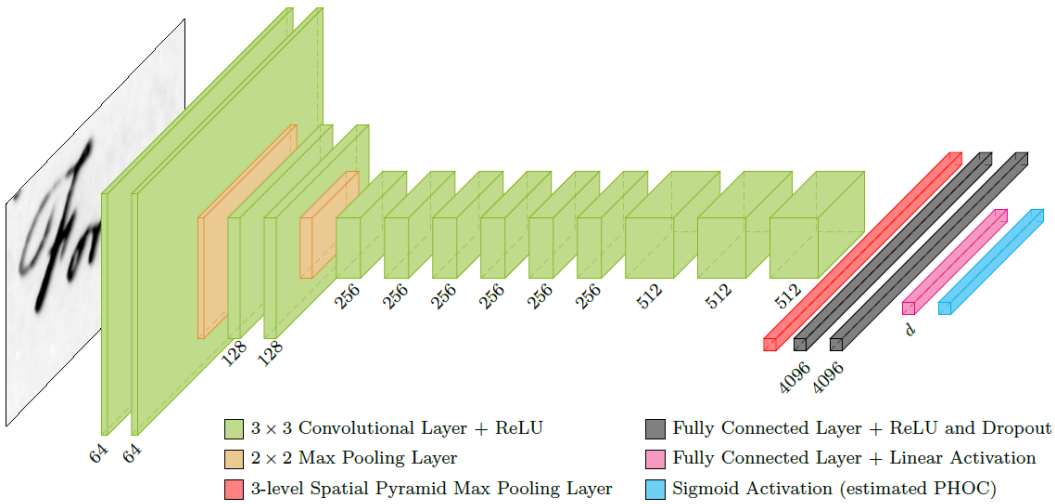


Figure 3.2.1: The architecture of the SPP-PHOCNet [SF16].

And the experimental results of the recognition tasks in this thesis are compared with those of [PW16].

3.2.1 *PHOCNet*

A PHOCNet is inspired by VGG16 [SZ14], but specially designed for estimating PHOC representation from the segmented word images that typically have arbitrary size. Because its work can be assorted as multi-label classification task, PHOCNet places sigmoid activation functions in its last layer with Binary Cross Entropy Loss as cost function for training the network. In the test, the PHOCNet computes each posterior of attribute being 1 given an image as each of its sigmoid outputs respectively. The overall architecture of the SPP-PHOCNet is depicted in figure 3.2.1.

The PHOCNet is a specially designed CNN architecture suitable for estimating attributes from document images, and its architecture is inspired by VGG16 which is one of the proposed networks in [SZ14]. The network showed state-of-the-art result in large-scale natural image recognition tasks, and therefore PHOCNet inherits most of design choices of VGG16 in its convolutional part which is responsible for feature extraction from the image. These networks replicate convolutional layers with increasing the number of filters and include max-pooling layer between convolutional layers, so that the networks learn large number of more abstract features from reduced

spatial representations in higher convolutional layer. In addition, PHOCNet uses only 3×3 filters for imposing a regularization in [SZ14].

While the previous design choices are suitable for both natural images and word images, following two aspects have to be considered when designing an attribute CNN architecture for document analysis. At first, in the case of segmentation-based word spotting and recognition, the input images of the network rather largely vary in their size. Besides, traditional strategies to generate fixed size images such as resizing and cropping possibly causes semantic distortion. Secondly, typical CNN used for natural image classification like proposed in [SZ14], is assorted as multi-class classifier. It works to predict one out of a number of target classes for a given image with softmax activation just before the output layer of network. However, this scenario is not feasible for Attribute CNN, because estimating attributes vector representation can be concerned as multi-label classification task. With considering above two aspects, the PHOCNet introduces following two differences to the VGG16.

Spatial Pyramid Pooling (SPP) and Temporal Pyramid Pooling (TPP) layer

One of the basic ideas of CNN, convolution with local receptive field, enables to extract feature from input images that vary in size. And varying size of input images then results in different number of parameters of feature maps at the end of the convolutional part. However, fully-connected layers assign fixed number of neurons that cannot take into account varying number of parameters from preceding convolutional layers. To address this problem, the PHOCNet employs a Spatial Pyramid Pooling (SPP) layer between the last convolution layer and first fully-connected layer. In order to achieve fixed number of parameters regardless of the size of feature maps, a SPP layer divides each feature map into fixed number of spatial bins, and then applies pooling operations for each bins. The term 'Pyramid' means that divided spatial bins are respectively subdivided in the following level. In detail, each cell is divided both along the horizontal and vertical axis. And max pooling operation is then applied to each cell. Let the number of feature maps and levels are denoted as k and l respectively. The number of output from the first SPP layer is same as the number of feature maps, k. On the second level, each k number of spatial bin of the first level is divided by both axes and therefore max pooling is applied for $4 \cdot k$ bins. With one more step of iteration, the third level of SPP layer produces $16 \cdot k$. After all, the pooling output are stacked throughout all the levels, and the SPP layer then always provides $\sum_l 4^{l-1} \cdot k$ inputs to the following fully-connected layer.

However, according to the argument in the [SF18], a fine grained spatial partition along the horizontal axis is much more important than that along the vertical axis

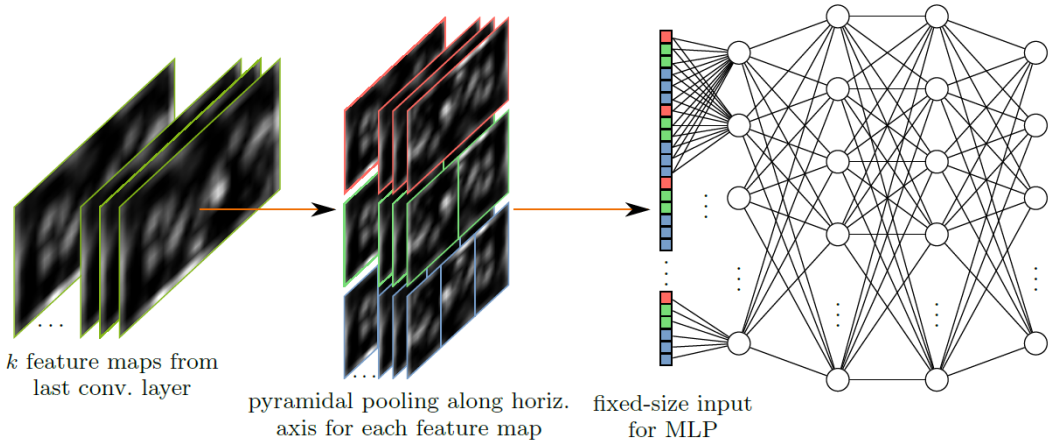


Figure 3.2.2: The visualization of the TPP Layer [SF18].

when dealing with word images. Based on the claim, the author proposed a modified version of the SPP layer which divides image representations into spatial bins only along the horizontal axis rather than both axes. Proposed layer is dubbed as Temporal Pyramid Pooling (TPP) layer, because the concatenation of different spatial splitting along the axis of writing direction for all the levels represents the temporal progress of writing. The idea of TPP layer is visualized in figure 3.2.2. As shown in the figure, the number of horizontal splitting for each feature map is same as level l . Hence, the TPP layer yields only $\sum_l l \cdot k$ which is much smaller number of outputs than SPP layer without loss of performance. Indeed, both SPP layer and TPP layer enable to fed into fixed size input for MLP-part despite arbitrary number of parameters of feature maps from last convolutional layer.

When Sudholt et al. uses TPP layer in [SF18], the proposed attribute CNN architecture is named TPP-PHOCNet to distinguish it from the firstly proposed PHOCNet employing SPP layer in [SF16]. Because only the modified version, TPP-PHOCNet, is used in this thesis work, the term 'PHOCNet' in the rest of the chapters indicates TPP-PHOCNet of [SF18] to avoid confusion.

Sigmoid Activation (estimated PHOC)

Different to heavily used traditional CNNs used for predicting the most probable class for a given natural image with softmax (2.1.9), the goal of the attribute CNN for document analysis application is to estimate attribute vector for a given word image.

In other words, in the attribute-based classification tasks, CNNs are used as multi-label classifier rather than multi-class classifier. With using the same notation in (2.1.12) except specifying output representation y as attribute a, the aim of the attribute CNNs can be written as below from a probabilistic point of view.

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} p(a^{(i)} \mid x^{(i)}, \theta) \tag{3.2.1}$$

As explained in section 3.1, the PHOC considers the presence of attributes with the form of binary histogram. If M number of attributes $a_1, ..., a_M$ for a given sample are assumed to be dependent variables each other, then they follow multivariate Bernoulli distribution.

$$p(a_1, ..., a_M \mid x^{(i)}, \theta) \tag{3.2.2}$$

The approach make the network have to estimate all of the following posteriors.

$$\begin{aligned}
&p(a_1 = 0, a_2 = 0, ..., a_{M-1} = 0, a_M = 0 \mid x^{(i)}, \theta) \\
&p(a_1 = 1, a_2 = 0, ..., a_{M-1} = 0, a_M = 0 \mid x^{(i)}, \theta) \\
&\qquad\qquad\qquad \vdots \\
&p(a_1 = 1, a_2 = 1, ..., a_{M-1} = 1, a_M = 0 \mid x^{(i)}, \theta) \\
&p(a_1 = 1, a_2 = 1, ..., a_{M-1} = 1, a_M = 1 \mid x^{(i)}, \theta)
\end{aligned}$$

Even without considering bi-gram and tri-gram , the PHOC representation with 5 levels requires the length of $36 \cdot (1 + 2 + 3 + 4 + 5) = 540 = M$ for 26 letters of English alphabet and 10 letters of digits. It means that the network should compute at least 2^{540} different terms and therefore the output layer size also is 2^{540}. However, $2^{540} \approx 3.599 \cdot 10^{162}$ is practically infeasible number to implement and to compute. As usual, the conditional independence of attributes is assumed based on empirical evidence to make the problem tractable. Then, (3.2.2) can be re-written as below

$$p(a_1, ..., a_M \mid x^{(i)}, \theta) = \prod_{m=1}^{M} p(a_m \mid x^{(i)}, \theta). \tag{3.2.3}$$

Here, the problem is reduced to compute $p(a_m \mid x^{(i)}, \theta)$. A multi-label classifier such as PHOCNet traditionally employs sigmoid activation which maps the linear input

to (0,1). Its output represents pseudo probability of presence of the attribute, and therefore the attribute can be interpreted as Bernoulli distributed variables.

Regarding the learning of the PHOCNet, it employs a binary logistic-loss for multi-label classification. The output of m-th sigmoid function, estimation of being '1' of m-th attribute given sample, is denoted by $p(a_m = 1 \mid x^{(i)}, \theta)$. And $\hat{a}_m$ denotes the label of that attribute. By taking into account above notations, the equality of the output of the network and its label can be expressed as

$$p(a_m = 1 \mid x^{(i)}, \theta) = \hat{a}_m, \quad p(a_m = 0 \mid x^{(i)}, \theta) = 1 - \hat{a}_m \tag{3.2.4}$$

$$\prod_{m=1}^{M} p(a_m \mid x^{(i)}, \theta) = \prod_{m=1}^{M} \hat{a}_m^{a_m} (1 - \hat{a}_m)^{(1-a_m)}, \quad a_m \in \{0, 1\}. \tag{3.2.5}$$

The negative log-likelihood of (3.2.5) is given by

$$L(\hat{a}^{(i)}, a^{(i)}) = -\frac{1}{M} \sum_{m=1}^{M} [a_m \log \hat{a}_m + (1 - a_m) \log (1 - \hat{a}_m)]. \tag{3.2.6}$$

Finally, the loss function can be computed by summing it all up for N samples in a batch

$$L(\hat{a}, a) = -\frac{1}{N \cdot M} \sum_{i=1}^{N} \sum_{m=1}^{M} [a_m^{(i)} \log \hat{a}_m^{(i)} + (1 - a_m^{(i)}) \log (1 - \hat{a}_m^{(i)})]. \tag{3.2.7}$$

3.2.2 *CNN-N-Gram for Handwriting Word Recognition*

Attribute-based classification was introduced by Lampert et al. [LNH14] especially for zero-shot recognition of natural image. Handwriting word recognition can be considered as a special case of fine-grained zero-shot classification, because the task includes the case of classifying a word image into classes which are not necessarily covered by training data. Therefore, as clarified in [AGFV14], word recognition with embedded attributes proposed by Almazán et al. is closely related to attribute-based classification. In their work, the input image is transformed into the vector of attribute scores with choosing Fisher Vecotrs (FV) and linear SVMs as encoding method and classification algorithm respectively. On the other side of that, the ground truth label of an input image, i.e., the transcription of the word image is also transformed into the PHOC representation in a deterministic way. With regard to the recognition task,

ahead of processing the input image, each word in the closed lexicon that includes all words appearing in training and test are encoded into the binary vector of its ground truth attributes, i.e., PHOC. Here, the attribute score represents estimated presence of attribute given input image, and it has the same vector length to the PHOC representation with each element valued (0,1). Indeed, the recognition task, obtaining a transcription of the word image, becomes to find the nearest neighbour of the input among lexicon words. Additionally, Almazán et al. propose a Common Subspace Regression(CSR) in order to bring the imperfect attribute score of the word image to their ground truth binary value in a common subspace. The use of CSR for this purpose derives same formulation to CCA which learns a common subspace maximizing correlation of them. After applying CSR same as CCA, the recognition task is again to search the nearest neighbour on such a learned common subspace.

The recognition approach of [PW16] is closely related to that of [AGFV14]. A major difference is the way of estimating attribute vector from word image. In [PW16], a deep CNN called CNN-N-Gram is used to replace AttributeSVMs used in [AGFV14]. The task of the network is to estimate n-gram frequency profile which is same as PHOC given word image. Even [PW16] and [SF18] use identical attribute representation, the tasks of the network in their works are different. The PHOCNet is designed for handwriting word retrieval and the CNN-N-Gram is designed for handwriting word recognition. Because both networks are commonly inspired by VGG [SZ14] with respect to their architectures and have similar output representation, they have similar layout.

However, there are also architectural differences in detail. The convolutional part of CNN-N-Gram is inspired by VGG [SZ14]. The number of convolutional layers and filters in CNN-N-Gram is slightly different to the that of PHOCNet, and the CNN-N-Gram employs less conventional maxout activation rather than ReLU (2.1.2) used in the PHOCNet.

According to the claim of [PW16], the novelty of their network is separating the fully-connected layers into multiple parallel branches. They split the output of the last convolutional layer into 19 branches where each branch has two hidden layers and sigmoid activation so that they make each branch specialize in certain spatial section respectively. They assert that the usage of splitting parallel fully-connected layers not only allows the convolutional layer to learn visual feature of the characters regardless of their spatial position in the word, but also enables the fully-connected layers to learn spatial information.

There are similarities and differences to the PHOCNet with respect to the training procedure. The layout of each branch is similar to the fully-connected layers of the PHOCNet, and therefore [PW16] uses the Binary Cross Entropy Loss as cost function.

However, proposed branching architecture requires huge number of neurons and parameters compared to conventional non-splitting fully-connected layer used in the PHOCNet. Those large number of parameters need huge computational resources for learning. To address exploding computational cost in the training of their branched network, they propose a gradual way of learning which is similar to incremental training employed in [JSVZ14]. The training is based on the iteration of partial convergence of learning parameters. At first, the network is trained with only one branch of fully-connected layers. When the parameters are partially converged after initial training, one of remaining branches is added with an initial learning rate. Then, the remaining branches are being added iteratively.

While the PHOCNet is able to accept the input images of arbitrary size by the use of a SPP-layer, the CNN-N-Gram requires input images of fixed size. To generate fixed size input images, they resize the input image 100×32 without preserving the aspect ratio.

After training the network, [PW16] apply CCA to learn a common subspace and projections which maximize the canonical correlation between estimated attribute of word images and binary attribute of their ground truth label. At test, the recognition is to find the best output class which has a minimum cosine distance from an input image among lexicon words in the common subspace learned with CCA.

3.3 PROBABILISTIC CLASSIFIER WITH ATTRIBUTES

Proposed solution of handwriting word recognition in [AGFV14] can be seen as a special case of attribute-based classification introduced by [LNH14]. In [LNH14], they utilize their own notation and graphical representation to demonstrate that attribute-based classification is a proper solution to the zero-shot learning task. They denotes input image as $x \in X$, and the K number of training classes are given by $\{y_1, ..., y_K\} \in Y$. Assuming a fixed length of M of the binary attribute representation, the k-th class in Y induces a binary vector: $a^{y_k} = (a_1^{y_k}, ..., a_M^{y_k})$. Similarly, they denote L number of test classes as $\{z_1, ..., z_L\} \in Z$ where all test classes are zero-shot classes not appeared during training. However, $Z \cap Y = \varnothing$ is not necessary and adopted as a representative situation for the sake of clarify. For handwriting word recognition, Y and Z are joint sets in general, and only a few zero data classes appeared during the test. They also clarify that $Z \subseteq Y$ is sufficient. Under these zero-shot learning setting, the task of attribute-based classification is to learn a non-trivial classifier which classify an input image to the previously unseen classes: $X \rightarrow Z$ by transferring information between the classes in Y and Z through attributes.

To implement the idea, they offer a method that decouple the images from the layer of classes with using an in-between attribute layer and term it as Direct Attribute Prediction (DAP) to distinguish it from another method called Indirect Attribute Prediction (IAP). The graphical representation of DAP is depicted in figure 3.3.1 While the input images and label classes of training input are always observed, in-between attributes are not observed but embedded. In detail, DAP method starts by extracting attribute representation $a^z = \{a_1^z, ..., a_M^z\}$ from the classes of training sample $z \in Z$ in a deterministic way. During training, the parameters of attribute embedding method β_m are learned with employing any supervised machine learning algorithms. For instance, Almazan et al. [AGFV14] uses FVs and SVMs for the encoding and learning algorithm, and this thesis work employs a specific CNN called PHOCNet explained in section 3.2.1. At test, the attribute score of query image can directly be estimated and then used for predicting output classes even for zero-data classes.

The graphical representation of DAP, figure 3.3.1, is realized by probabilistic model. From a probabilistic point of view, the classification task is to find the best class which maximizes the posterior of test class given a query image:

$$\underset{i=1,...,L}{\operatorname{argmax}}\, p(z_l \mid x_i) \tag{3.3.1}$$

The novelty of DAP is to use in-between layer of attribute. Taking that into account, they first form a model for the $image - attribute$ layer and then $attribute - class$ layer. Then they combine them with marginalization. Initially, the probabilistic representation of $image - attribute$ layer is given by

$$p(a_1, ..., a_M \mid x_i) \tag{3.3.2}$$

The method assumes that the trained classifier estimates each attribute given an image respectively, $p(a_m \mid x_i)$. To reflect the output of the trained classifier to the model, they assume conditional independence. Hence the complete $image - attribute$ layer is represented by

$$p(a_1, ..., a_M \mid x_i) = \prod_{m=1}^{M} p(a_m \mid x_i) \tag{3.3.3}$$

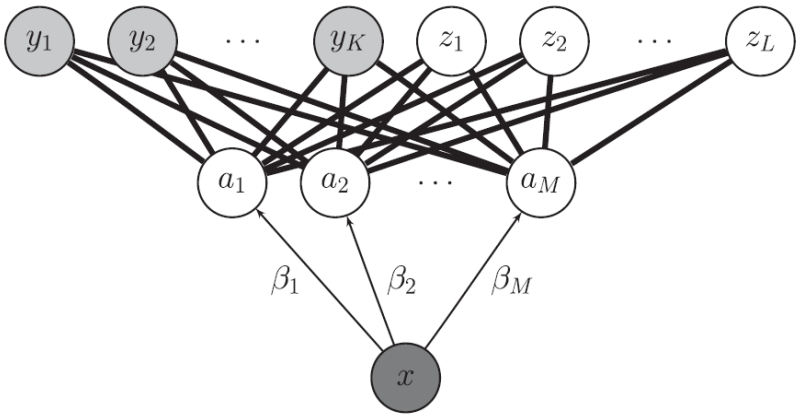


Figure 3.3.1: Illustration of graphical representation of DAP [LNH14].

On the $\mathtt{attribute}-\mathtt{class}$ layer, the classification task is to predict the test class from an input attribute pattern. They apply Bayes' rule to derive model so that it implies the descriptions of test class.

$$p(\mathtt{class} \mid \mathtt{pattern}) = \frac{p(\mathtt{class}, \mathtt{pattern})}{p(\mathtt{pattern})} = \frac{p(\mathtt{pattern} \mid \mathtt{class})p(\mathtt{class})}{p(\mathtt{pattern})}$$

Because every test class induces its attribute vector in a deterministic way, the colored likelihood term, $p(\mathtt{pattern} \mid \mathtt{class})$, make possible to derive the probabilistic representation of the model with considering the descriptions of test class. In specific, while the model of $\mathtt{attribute}-\mathtt{class}$ layer should provide posterior each test class given attribute pattern, $p(z_l \mid a_1, ..., a_M)$, the given information is that the only one attribute pattern aligned to a test class make likelihood be one. Otherwise, the likelihood and resulting posterior are always zero:

$$p(a_1 = a_1^{z_l}, ..., a_M = a_M^{z_l} \mid z = z_l) = p(a = a^{z_l} \mid z = z_l) = 1 \tag{3.3.4}$$

$$p(a_1 = a_1^{z_k}, ..., a_M = a_M^{z_k} \mid z = z_l) = p(a = a^{z_k} \mid z = z_l) = 0 \tag{3.3.5}$$

where $k = 1, ..., l-1, l+1, ...L$. To formulate above case statement, they use Iverson's bracket notation [Knu92] given by

$$p(a \mid z_l) = [[a = a^{z_l}]], \quad [[a = a^{z_l}]] = \begin{cases} 1, & \text{if} \quad a = a^{z_l} \\ 0, & \text{otherwise} \end{cases} \tag{3.3.6}$$

By applying Bayes' rule and Iverson's bracket notation, The complete $image - attribute$ is defined as

$$p(z_l \mid a) = \frac{p(a \mid z_l)\, p(z_l)}{p(a)} = \frac{[[a = a^{z_l}]]\, p(z_l)}{p(a^{z_l})} \tag{3.3.7}$$

In (3.3.7), only $p(a = a^{z_l})$ is considered as denominator because the posterior is always zero in other cases based on the case statement of (3.3.6).

Two derived models can be combined through marginalizing for all attribute patterns:

$$p(z_l \mid x_i) = \sum p(z_l \mid a) p(a \mid x_i), \text{ for } a \in \{0, 1\}^M \tag{3.3.8}$$

By substituting (3.3.3) and (3.3.7) into (3.3.8), each term of summation can be reformulated as

$$p(z_l \mid a) p(a \mid x_i) = \frac{p(z_l)}{p(a^{z_l})} [[a = a^{z_l}]] p(a \mid x_i) = \begin{cases} \frac{p(z_l)}{p(a^{z_l})} p(a^{z_l} \mid x_i), \text{ if } a = a^{z_l} \\ 0, \text{ otherwise} \end{cases} \tag{3.3.9}$$

The $[[a = a^{z_l}]]$ term in the right-hand side of (3.3.7) makes the model neglect all other cases except $a = a^{z_l}$. Hence, with previous steps of probabilistic realization, the posterior of a test class given an image can be calculated by

$$p(z_l \mid x_i) = \sum_{a \in \{0,1\}^M} p(z_l \mid a) p(a \mid x_i) = \frac{p(a^{z_l} \mid x_i) p(z_l)}{p(a^{z_l})} \tag{3.3.10}$$

The remaining task of probabilistic realization of DAP is to assign the best output class from all test classes $z_1, ..., z_L$ to a input image x_i based on calculated posteriors

by (3.3.10), and it is also known as the maximum a posteriori (MAP) estimation. The complete probabilistic model of DAP is derived by substituting (3.3.10) into (3.3.1)

$$\begin{aligned}\hat{z}_{MAP}(x_i) &= \underset{l=1,\dots,L}{\operatorname{argmax}}\, p(z_l \mid x_i) = \underset{l=1,\dots,L}{\operatorname{argmax}} \frac{p(a^{z_l} \mid x_i)p(z_l)}{p(a^{z_l})} \\ &= \underset{l=1,\dots,L}{\operatorname{argmax}} \frac{p(a_1^{z_l},\dots,a_M^{z_l} \mid x_i)p(z_l)}{p(a^{z_l})}\end{aligned} \tag{3.3.11}$$

Applying the conditional independence of attributes given image already assumed at (3.3.3) the estimation of the best output class is given by

$$\hat{z}_{MAP}(x_i) = \underset{l=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i)p(z_l)}{p(a^{z_l})} \tag{3.3.12}$$

4

ATTRIBUTE-BASED WORD RECOGNITION

In word recognition, the goal is to obtain a transcription of the word image, usually aided by a lexicon. In this thesis, attribute-based word recognition is an approach that achieves above goal by using an intermediate layer of attributes which allows to derive a common representation for word images and transcriptions. This approach offers two major benefits. First, it can be a solution to the problem of recognizing out of vocabulary (OOV) which means input words of test never appeared during training. It is because the use of attributes enables the recognition system to transfer information between word classes in a lexicon. The other is that it facilitates comparing images and strings based on a fixed length common representation.

An overview of an attribute-based word recognition is depicted in figure 4.0.1. It starts with determining an attribute representation of the words. In this thesis, the PHOC is chosen as attribute embedding on the basis of what is described in section 3.1. Then, as specified in section 3.2, the PHOCNet is employed to estimate attribute

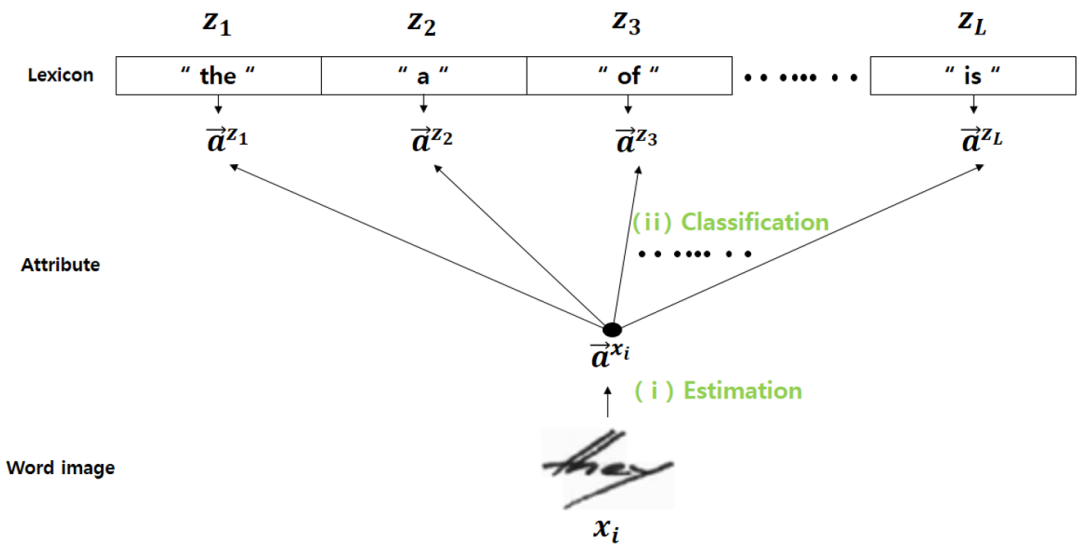
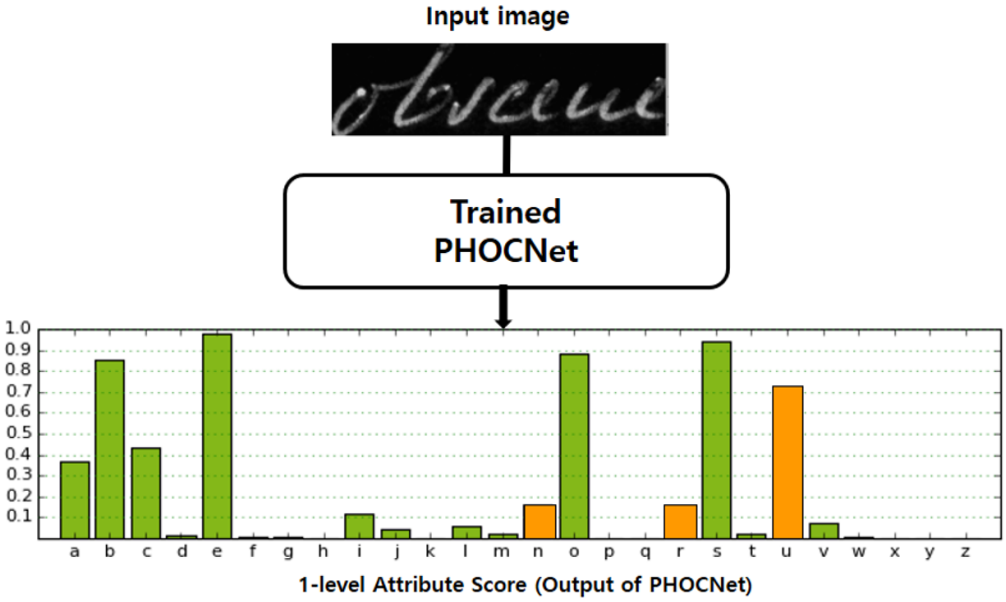


Figure 4.0.1: Overview of an Attribute-Based word recognition.

scores $\vec{a}^{x_i}$ from an input image x_i. At the same time, each word in the lexicon z_l is transformed into the PHOC of transcription $\vec{a}^{z_l}$ in a deterministic way that does not require learning. Attribute scores $\vec{a}^x$ and PHOCs of transcription $\vec{a}^z$ are always fixed length vector. Therefore, an arrow over those vectors is omitted in the rest of the thesis: a^x and a^z. Consequently, the recognition task is reduced to the problem of classifying attribute scores of an input image a^{x_i} into one of the classes in a lexicon transformed to PHOCs $a^{z_1}, ..., a^{z_L}$. For the former, the algorithm used at the lower half of the figure 4.0.1, can be considered as a multi-label classifier. In the latter, the algorithm used at the upper half of the figure, is included in a multi-class classifier. Therefore, the overall performance of recognition is determined by the cascade of both classifiers. In particular, this thesis mainly focus on the multi-class classifiers for the following reasons.

Recognition results are influenced by the design of multi-class classifier as well as by the way of estimating attribute scores, because the algorithm estimating attribute score is trained for minimizing loss per attribute rather than difference in whole transcription of word. In [AGFV14], attribute-based prediction algorithm requires one classifier per attribute. Besides, the PHOCNet used in this work also assumes conditional independence of each attribute given image in order to be a multi-label classifier with sigmoid activation. Then, the loss function of PHOCNet derived in (3.2.6) is simply to sum up the loss per attribute with negative log. This means the PHOCNet only learns and estimates the presence of each character independently in a certain spatial position from word images. However, a precise estimation of independent character existence not always guarantee the best word recognition result regardless of the way of multi-class classification.

One example, illustrated in figure 4.0.2, is taken from one experiment in order to show that the use of different multi-class classifiers can lead to different recognition results under same experimental configuration. The input of the example is a segmented word image with its ground truth label of 'obscene'. The values of 1-level attribute scores histogram are the first 26 outputs of the PHOCNet trained under same setups as [SF18]. Each of them represents the posterior probability of the presence of the English alphabet letters from a to z given word image. As discussed in section 3.2.1, employing the PHOCNet yields state-of-the-art results for word spotting in handwritten documents. It means the PHOCNet detects visual pattern of handwritten characters very well. Additionally, in the example, the PHOCNet predicts that the characters contained in the ground-truth label, such as 'b','e','o' and 's', would be present in the input image. However, this successful detection of visual pattern is not always profitable for handwriting word recognition because of visual ambiguity. In the example, whereas the input word does not contain character 'u' but includes the 'n' as



Multi-class Classifier	Reocgnition Result	Ground-truth Label
Method 1	'obsecure'	'obscene'
Method 2	'obscene'	

Figure 4.0.2: Example of recognition results from using different multi-class classifiers.

its sixth character, the PHOCNet yields a higher probability of 0.69 for the character 'u' than the character 'n' with a probability of 0.18. It can be a instance of the claim that the output of the PHOCNet is reliable, because the visual appearance of the character is more close to 'u' than 'n' even in the recognition of a human. However, especially in the case of Method 1 which represents the most naive approach, this strong belief of existing character 'u' overwhelms existence of ground-truth 'n' and misleads to incorrect lexicon word in recognition. On the other hand, one other superior method to Method 1 introduced in this chapter can classify given attribute scores to correct class of lexicon same as ground-truth label. The reason is that this Method 2 involves learning algorithm so that the multi-class classifier of given attribute scores enables compensate visual ambiguity or uncertainty of handwriting.

In this chapter, four different multi-class classifiers are introduced to improve the performance of attribute based word recognition. At first, the most naive approach that find nearest neighbour with cosine distance is introduced. Additionally, the cascade of classifiers in terms of probability, already introduced in section 3.3 as DAP, is used for handwriting word recognition. Third, a method termed CCADAP is proposed to avoid overfitting problem of CCA by using a temporal recognition of the word image. In the end, a conventional softmax MLPs is used as multi-class classifier, and a pseudo samples approach is introduced to train MLPs classifier for zero-data classes.

4.1 DIRECT ATTRIBUTE MATCHING (DAM)

The PHOCNet can successfully estimate attribute scores of an input image, and the transcription of each word in lexicon can be encoded as PHOC representation. After getting those attribute-based representations, the matching procedure in recognition can be simply done by suitable distance measure and nearest neighbour search between them, because attribute scores of an input image and PHOC of lexicon words lie on similar space of same dimensionality. Each of the score is spanned between 0 and 1 because the value is the output of a sigmoid function, and PHOC has binary representations. In the thesis, this conventional matching method for common representation is termed as Direct Attribute Matching (DAM).

DAM classifier is realized using the following notations. Let the number of attributes and classes be denoted by M and L respectively. An input image at test time and a word in the lexicon are denoted by x_i and z_l. Then, the vector of attribute scores is defined by

$$a^{x_i} = (a_1^{x_i}, ..., a_M^{x_i}) = (p(a_1 = 1 \mid x_i), ..., p(a_M = 1 \mid x_i)), \quad a^{x_i} \in (0,1)^M. \tag{4.1.1}$$

The vector a^{x_i} is the ordered set of M PHOCNet outputs where each of them is a posterior probability of a attribute existence given an image: $p(a_m = 1 \mid x_i)$. Likewise, the M dimensional PHOC vector of a lexicon word is defined by

$$a^{z_l} = (a_1^{z_l}, ..., a_M^{z_l}), \quad where\ a^{z_l} \in \{0,1\}^M. \tag{4.1.2}$$

The vector a^{z_l} is the ordered set of M different binary numbers determined by logical encoding in PHOCs. Thus the cosine distance between a^{x_i} and a^{z_l} is given by

$$d_{cos}(a^{x_i}, a^{z_l}) = 1 - \frac{< a^{x_i}, a^{z_l} >}{\|a^{x_i}\|_2 \|a^{z_l}\|_2} \tag{4.1.3}$$

where $<,>$ denotes inner product between two vectors and $\|\cdot\|_2$ represents L2-norm of a vector. The rest of the thesis also follows these notations. Since both vectors have same length and lie on fixed low dimension, it is very fast to compute distance between them. Finally, the recognition is to find the nearest neighbour based on computed distances:

$$l^* = \underset{l=1,\dots,L}{\operatorname{argmin}}\ d_{cos}(a^{x_i}, a^{z_l}) = \underset{l=1,\dots,L}{\operatorname{argmin}} \left(1 - \frac{< a^{x_i}, a^{z_l} >}{\|a^{x_i}\|_2 \|a^{z_l}\|_2}\right) \tag{4.1.4}$$

where l^* indicates the index of the best output class. In addition, the Bray-Curtis (BC) dissimilarity also can be considered as a distance metric that measures dissimilarity between two vectors. BC dissimilarity is defined as

$$d_{BC}(a^{x_i}, a^{z_l}) = \frac{\sum |a^{x_i} - a^{z_l}|}{\sum (a^{x_i} + a^{z_l})}. \tag{4.1.5}$$

However, only cosine distance is considered as metric for DAM in the thesis, because all other similar works that are the benchmark for evaluating performances use cosine distance. Hence, in the rest of the thesis, cosine DAM is a baseline method which helps evaluate performance by comparing experimental results with other works [AGFV14], [PW16], and methods introduced in following sections 4.2, 4.3.2 and 4.4.

4.2 DIRECT ATTRIBUTE PREDICTION (DAP) MODEL

As discussed in section 3.3, a probabilistic realization of DAP computes the posterior of a lexicon word given an image:

$$p(z_l \mid x_i) = \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)}{p(a^{z_l})}. \tag{4.2.1}$$

Thus, the overall posterior probability is determined by defining $p(a_m^{z_l} \mid x_i)$, $p(z_l)$ and $p(a^{z_l})$. $\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i)$ refers to the product of attribute scores estimated by PHOCNet. To compute it, 'one minus PHOCNet output' is required to estimate the

probability of complement state, because the output of PHOCNet, i.e., attribute score, only represents the probability of the attribute being present given an image:

$$p(a_m = a_m^{z_l} \mid x_i) = \begin{cases} p(a_m = 1 \mid x_i), \text{ if } a_m^{z_l} = 1 \\ p(a_m = 0 \mid x_i) = 1 - p(a_m = 1 \mid x_i), \text{ if } a_m^{z_l} = 0 \end{cases} \tag{4.2.2}$$

where $p(a_m = 1 \mid x_i)$ is the output of the PHOCNet. The factor implies that the DAP method exploits much more information from an image than cosine DAM in the process of recognition, since DAP utilizes the score of false attributes which are not being present in the class z_l. The example depicted in the table 4.2.1 explicitly specifies the advantage of the DAP method. The example considers only three attributes: M=3. For l-th class, the first attribute is present, but the remaining two attributes are not. Estimation performance of the PHOCNet for all attributes is assumed to be identical so that it gives 0.9 if present, otherwise 0.1. Then, based on (4.2.2), the first factor is given by

$$\begin{aligned} \prod_{m=1}^{3} p(a_m^{z_l} \mid x_i) &= p(a_1 = a_1^{z_l} \mid x_i) p(a_2 = a_2^{z_l} \mid x_i) p(a_3 = a_3^{z_l} \mid x_i) \\ &= p(a_1 = 1 \mid x_i) p(a_2 = 0 \mid x_i) p(a_3 = 0 \mid x_i) \\ &= p(a_1 = 1 \mid x_i)(1 - p(a_2 = 1 \mid x_i))(1 - p(a_3 = 1 \mid x_i)) \\ &= 0.9 * 0.9 * 0.9. \end{aligned} \tag{4.2.3}$$

z_l	$a_1^{z_l}$	$a_2^{z_l}$	$a_3^{z_l}$
PHOCs	1	0	0
PHOCNet Outputs	$p(a_1 = 1 \mid x_i)$	$p(a_2 = 1 \mid x_i)$	$p(a_3 = 1 \mid x_i)$
Attribute Scores	0.9	0.1	0.1

Table 4.2.1: Example of attribute representations and their values for the purpose of comparing DAP and DAM.

To compare, cosine distance is calculated by substituting the values of the table 4.2.1 into (4.1.3):

$$\begin{aligned} d_{cos}(a^{x_i}, a^{z_l}) &= 1 - \frac{< a^{x_i}, a^{z_l} >}{\|a^{x_i}\|_2 \|a^{z_l}\|_2} \\ &= 1 - \frac{p(a_1 \mid x_i) a_1^{z_l} + p(a_2 \mid x_i) a_2^{z_l} + p(a_3 \mid x_i) a_3^{z_l}}{\|a^{x_i}\|_2 \|a^{z_l}\|_2} \\ &= 1 - \frac{p(a_1 \mid x_i) \cdot 1 + \cancel{p(a_2 \mid x_i) \cdot 0} + \cancel{p(a_3 \mid x_i) \cdot 0}}{\|a^{x_i}\|_2 \|a^{z_l}\|_2} \\ &= 1 - \frac{p(a_1 \mid x_i)}{\|a^{x_i}\|_2 \|a^{z_l}\|_2}. \end{aligned} \tag{4.2.4}$$

In DAP, all outputs of the PHOCNet are used for classification, whereas cosine distance does not reflect scores of false attributes. Specifically, DAP reflects non-presence of false attribute given image by subtracting aligned PHOCNet output from 1. However, cosine distance only take into account attribute score of true attribute, because the scores of false attributes are cancelled by multiplying zero in the linear combination of the inner product term. As discussed in 3.1 and 3.2.1, the PHOC representation with 5 levels requires the length of $36 \cdot (1+2+3+4+5) = 540 = M$ for 26 letters of English alphabet and 10 letters of digits. However, according to the [BS12], the average length of English words is 5.1. Therefore, even if the average length of an English word is supposed to be 6 with all different characters, the number of true attributes of English word is only 36 out of 540 in average. This means that estimation of almost 500 attribute scores is redundant for every class while DAP always uses all 540 attribute scores. In addition, in this work, to use more attribute scores including false attributes is more advantageous for the classification. Because the PHOCNet is trained not only for true attribute but also for false attribute of training samples. Taking the table shown 4.2.1 as an example, cosine distance between a^{x_i} and a^{z_l}, (4.1.4), does not exploit $p(a_2 \mid x_i)$ and $p(a_3 \mid x_i)$ by multiplying zeros. However, ignoring those estimation is a huge loss, because the PHOCNet is trained and can estimate scores of attribute not appeared in word image as 0.1 for both a_2 and a_3.

The second component of the DAP model (4.2.1), $p(z_l)$, represents a prior probability of lexicon class. In other words, it indicates the probability of $z = z_l$ as classification result in a future test. One possible assumption also proposed by [LNH14] is a uniform distribution:

$$p(z_l) = p(z = z_l) = \frac{1}{L}, \quad \text{for all } l = 1, ..., L \tag{4.2.5}$$

where L is the number of words in the lexicon. It considers the absence of specific knowledge about the classes of upcoming test image. However, frequencies of words in a large number of training samples induces a reliable belief of class of upcoming word for document image analysis applications. Thus, it deserves to contemplate an empirical distribution of training classes $p(z_l)_{train}$ which indicates the number of training samples for each classes over total number of training samples. Then, the prior probability $p(z_l)_{test}$ is assumed to follow $p(z_l)_{train}$. The reason is that $p(z_l)_{train}$ induces a term frequency (TF) of training documents, and therefore it can be used as a common feature unless train and test documents belong to totally different categories:

$$p(z_l)_{test} \sim p(z_l)_{train} = \frac{\#\,\text{training samples for}\, z_l}{\#\,\text{training samples}} = \frac{\#\,\text{certain word}}{\#\,\text{words in document}} = \text{TF} \tag{4.2.6}$$

To apply empirical distribution for the DAP, a smoothing technique is required to avoid zero posteriors which are given by

$$p(z_k)_{train} = \frac{\#\,\text{training samples for}\, z_k}{\#\,\text{training samples}} = \frac{0}{\#\,\text{training samples}} = 0 \tag{4.2.7}$$

$$p(z_k \mid x_i) = \frac{\prod_{m=1}^{M} p(a_m^{z_k} \mid x_i) p(z_k)}{p(a^{z_l})} = \frac{\prod_{m=1}^{M} p(a_m^{z_k} \mid x_i) \cdot 0}{p(a^{z_l})} = 0 \tag{4.2.8}$$

where z_k represents zero-data class. As formulated in (4.2.7), the probability of z_k from training is always zero, because there is no training sample for that class. The assumption proposed in (4.2.6) makes test prior of z_k to zero. Then, the output class of DAP will never be z_k, because the posterior of a zero-data class always goes to zero by multiplying $p(z_k) = 0$. In order for (4.2.8) to be zero-data learning, a pseudo count is added to every class. Considering the pseudo count also called smoothing parameter, the empirical distribution is smoothed as follows:

$$p(z_l)_{train} = \frac{\#\,\text{training samples for}\, z_l + \alpha}{\#\,\text{training samples} + \alpha \cdot L} \tag{4.2.9}$$

where α is a smoothing parameter. This technique is called $Laplace\ smoothing$ and often referred to as $add-one\ smoothing$ where $\alpha = 1$. After laplace smoothing, (4.2.7) is rewritten by

$$p(z_k)_{train} = \frac{\#\,\text{training samples for}\, z_k + \alpha}{\#\,\text{training samples} + \alpha \cdot L} = \frac{\alpha}{\#\,\text{training samples} + \alpha \cdot L} \neq 0. \tag{4.2.10}$$

Hence, in the rest of the thesis, smoothed empirical distribution of training classes is considered as test class prior to address zero probability problem.

The last term of DAP computation is the denominator $p(a^{z_l})$. Since $M = 540$ induces too large number 2^{540} outcomes for joint probability $p(a)$, attributes are assumed to be statistically independent to each other, i.e., $p(a)$ is assumed to follow a factorial distribution:

$$p(a) = p(a_1, ..., a_M) = \prod_{m=1}^{M} p(a_m). \tag{4.2.11}$$

As defined in 3.1, an attribute has only two possible states, $true$ and $false$, written as 1 and 0 respectively. Therefore, estimating probability of independent attribute can be considered as a Bernoulli trial:

$$p(a_m = a_m^{z_l}) = \begin{cases} p(a_m = 1), & \text{if } a_m^{z_l} = 1 \\ p(a_m = 0) = 1 - p(a_m = 1), & \text{if } a_m^{z_l} = 0. \end{cases} \tag{4.2.12}$$

In the process of Bernoulli trials, it is reasonable to assume unbiased attribute which means the probability of presence is the same at every test time as $1/2$:

$$p(a_m^{z_l}) = p(a_m = a_m^{z_l}) = \frac{1}{2} \quad \text{for all } m = 1, ..., M. \tag{4.2.13}$$

This assumption simplifies the classification task of DAP, because the denominator of (4.2.1) always being constant $(1/2)^M$ for all classes. Similar to class prior $p(z_l)$, empirical approach can be considered for $p(a_m)$ by estimating M different distributions from training samples. The probability of an attribute being present is assumed to be

biased. Let the number of training samples be N, then $p(a_m)_{train}$ is determined by counting the number of occurrences of a_m in N independent experiments:

$$p(a_m^{test} = 1) \sim p(a_m^{train} = 1) = \frac{\#\,samples\,of\,(a_m^{train} = 1)}{N}. \tag{4.2.14}$$

The above experiments is iterated M times to derive the probability of all attributes.

4.3 CCADAP

In this section, the application of CCA similar work [AGFV14] is discussed first, and then problems of the CCA application in this work are discussed. Lastly, CCADAP is proposed to address those application problems of CCA.

4.3.1 *CCA*

As discussed in 2.2, the motivation of CCA is to analyse the relation between two feature representations resulting from two different views. With respect to attribute-based word recognition, the attribute of word appears in two different representations. One representation is attribute score estimated by the PHOCNet. The other representation is the binary PHOC. Then, CCA is a method to learn relation between the attribute score and PHOC of word. Taking one step further, practically, the goal of CCA in this work is to make estimated imperfect scores get closer to perfect binary values by learning a common subspace of them. To achieve this goal, a ridge regression is possibly a more intuitive solution than CCA, because it learns to minimize the distance between the space spanned by attribute scores and binary PHOCs directly rather than learning common subspace and projection like as CCA. However, CCA has advantage of exploiting the correlation between scores, the correlation between PHOCs and the correlation between those two, whereas a ridge regression does not take advantage of all such correlations.

Similar to the notations used in section 2.2, let N be a number of training samples, and let M be the dimension of attribute representations. Then, the datasets of estimated score $A_x \in (0,1)^{N\times M}$ and label PHOC $A_y \in \{0,1\}^{N\times M}$ are defined by

$$A_x = \begin{pmatrix} | & & | \\ a_{x,1} & \cdots & a_{x,M} \\ | & & | \end{pmatrix}, \quad A_y = \begin{pmatrix} | & & | \\ a_{y,1} & \cdots & a_{y,M} \\ | & & | \end{pmatrix} \tag{4.3.1}$$

In order to find a weight W which minimizes linear distance between those two, a ridge regression problem is defined by

$$W^* = \underset{W}{\operatorname{argmin}} \frac{1}{2}\|A_x W - A_y\|_2^2 + \frac{1}{2}\Omega(W) \tag{4.3.2}$$

where $\Omega(W)$ is a regularization term. Then, based on [HK70], the closed form solution of W from (4.3.2) is given by

$$W = (A_x^T A_x + \rho I)^{-1} A_x^T A_y \tag{4.3.3}$$

The term $A_x^T A_x$ implies that the solution of ridge regression exploits correlation between attribute scores. As assumed in 2.2, the attributes are standardized to zero mean and unit variance. For the standardized variables, the covariance is equal to its correlation and is defined by the inner product of variables:

$$corr(a_x, a_y) = cov\left(\frac{a_x - \overline{a}_x}{s_x}, \frac{a_y - \overline{a}_y}{s_y}\right) = cov(a_x, a_y) \tag{4.3.4}$$

$$cov(a_x, a_y) = \frac{< a_x, a_y >}{N-1} - \overline{a}_x \overline{a}_y = \frac{< a_x, a_y >}{N-1} \tag{4.3.5}$$

where $\overline{a} = 0$ and $s = 1$ denotes mean and standard deviation respectively. Therefore, $A_x^T A_x$ refers to a covariance matrix equal to a correlation matrix.

$$\begin{aligned} C_{xx} &= \begin{pmatrix} cov(a_{x,1}, a_{x,1}) & \cdots & cov(a_{x,1}, a_{x,M}) \\ \vdots & \ddots & \vdots \\ cov(a_{x,M}, a_{x,1}) & \cdots & cov(a_{x,M}, a_{x,M}) \end{pmatrix} = \begin{pmatrix} corr(a_{x,1}, a_{x,1}) & \cdots & corr(a_{x,1}, a_{x,M}) \\ \vdots & \ddots & \vdots \\ corr(a_{x,M}, a_{x,1}) & \cdots & corr(a_{x,M}, a_{x,M}) \end{pmatrix} \\ &= \frac{1}{N-1} \begin{pmatrix} a_{x,1}^T a_{x,1} & \cdots & a_{x,1}^T a_{x,M} \\ \vdots & \ddots & \vdots \\ a_{x,M}^T a_{x,1} & \cdots & a_{x,M}^T a_{x,M} \end{pmatrix} \\ &= \frac{1}{N-1} A_x^T A_x \end{aligned} \tag{4.3.6}$$

Through the same interpretation as (4.3.2), C_{xy}, C_{yx} and C_{yy} are also defined as

$$C_{XY} = \frac{1}{N-1} A_x^T A_y, \qquad C_{YX} = \frac{1}{N-1} A_y^T A_x, \qquad C_{YY} = \frac{1}{N-1} A_y^T A_y \tag{4.3.7}$$

Hence, with respect to the ridge regression, the correlations between scores $A_x^T A_x$ as well as between score and binary label $A_x^T A_y$ are considered in the process of finding solution W that minimizes the objective function (4.3.2).

In CCA, instead of analysing linear relation between scores a_x and PHOC labels a_y directly, its objective is to minimize enclosing angle between images z_x and z_y in the common subspace where z_x and z_y are projected from a_x and a_y respectively. Specifically, as explained in section 2.2, the images in the new coordinate system, z_x and z_y, are algebraically expressed as (2.2.1). CCA pursues above objective by finding canonical weights $w_x \in \mathbb{R}^M$ and $w_y \in \mathbb{R}^M$ which holds constraints of z_x and z_y described in (2.2.2) and (2.2.3). By the formulation of CCA problem described in section 2.2, the problem is reduced to find eigenvalues of the matrix $C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy}$ by solving the characteristic equation written in (2.2.20). Then, the weight vector of PHOC labels w_y can be obtained by computing eigenvectors of (2.2.21), and the weight vector of attribute scores w_x is induced by the relation between w_x and w_y defined at (2.2.17). Thus, the learning weights of CCA, w_x and w_y, are determined by the matrix $C_{yy}^{-1} C_{yx} C_{xx}^{-1} C_{xy}$. It implies that CCA exploits the correlation between scores C_{xx}, between score and PHOC label C_{yx}, C_{xy} and between PHOC labels C_{yy}. To compare CCA with a ridge regression, the solution of a ridge regression (4.3.3) can be expressed by covariance matrices:

$$W = (A_x^T A_x + \rho I)^{-1} A_x^T A_y = (C_{xx} + \rho I)^{-1} C_{xy} \tag{4.3.8}$$

C_{yy} is not considered in the regression but is considered in CCA. The covariance matrix of PHOC labels C_{yy} is given by

$$C_{yy} = \begin{pmatrix} cov(a_{y,1}, a_{y,1}) & \cdots & cov(a_{y,1}, a_{y,M}) \\ \vdots & \ddots & \vdots \\ cov(a_{y,M}, a_{y,1}) & \cdots & cov(a_{y,M}, a_{y,M}) \end{pmatrix} = \begin{pmatrix} corr(a_{y,1}, a_{y,1}) & \cdots & corr(a_{y,1}, a_{y,M}) \\ \vdots & \ddots & \vdots \\ corr(a_{y,M}, a_{y,1}) & \cdots & corr(a_{y,M}, a_{y,M}) \end{pmatrix} \tag{4.3.9}$$

where $a_{y,m} \in \{0,1\}^N$ for all M attributes. In other words, CCA additionally takes into account correlation between binary PHOCs representing ground-truth attribute labels. Therefore, CCA deserves to be considered as a better learning algorithm than the regression in this application. The reason is that CCA analyses and reflects more relations between attribute representations than the case of regression.

Using a machine learning model such as CCA to learn relations between attribute representations may result in better recognition performances than without using that. However, the attribute-based word recognition model used in this thesis already

employs the machine learning model called PHOCNet to estimate attribute score. In the case of using these multiple machine learning models, duplicated use of training samples from different learning method can cause overfitting problems. The most intuitive choice of training data for CCA is to use the training samples of handwriting word recognition benchmark dataset which is also used to train PHOCNet. Let the data matrix of estimated scores from such training samples be A_x^{train}. And let the data matrix of encoded binary PHOCs from their ground-truth labels be A_y^{train}. Then, with reference to the formulas in (2.2.4) and (2.2.7), the objective of CCA for this case is given by

$$w_x^{train}, w_y^{train} = \underset{w_x, w_y}{\operatorname{argmax}} < A_x^{train} w_x, A_y^{train} w_y > \tag{4.3.10}$$

where learned weights by CCA are denoted by w_x^{train} and w_y^{train}. In the test, $A_x^{test} \in (0,1)^{Q \times M}$ represents attribute scores estimated from input images, and $A_y^{lexicon} \in \{0,1\}^{L \times M}$ contains binary PHOCs encoded from transcription of lexicon words. Let $A[i]$ refers to i-th row of matrix A. Then, as specified in section 4.1, employing DAM classifier without CCA is written by

$$\vec{o}^{\,*} = (o_1^*, ..., o_Q^*) = \underset{l=1,...,L}{\operatorname{argmin}} \, d_{cos}\big(A_x^{test}[q] \, , \, A_y^{lexicon}[l] \big) \quad \text{for } q = 1, ..., Q \tag{4.3.11}$$

where the vector element o_q indicates the index of output class resulted from q-th input word. Q is test sample size which indicates the number of input image that goes into the recognition system at one time during test, and therefore the smallest possible number of Q is one. According to these notations, the classification task with CCA can be described by

$$\vec{o}^{\,*} = (o_1^*, ..., o_Q^*) = \underset{l=1,...,L}{\operatorname{argmin}} \, d_{cos}\big(A_x^{test} w_x^{train}[q] \, , \, A_y^{lexicon} w_y^{train}[l] \big). \tag{4.3.12}$$

Please note that $'\text{for } q = 1, ..., Q'$ is omitted in here and the rest of thesis. In detail, test samples and lexicon words are projected on to the new coordinate system which maximizes correlation by w_x^{train} and w_y^{train}. These projection weights are learned from training samples and their ground-truth labels where training samples of CCA are also used to train PHOCNet. Using the same dataset for training PHOCNet and CCA in this way causes overfitting for CCA because of a bias. To begin with, the PHOCNet is trained by given training sample in handwriting word recognition benchmark dataset. After the training, trained PHOCNet estimates attribute scores

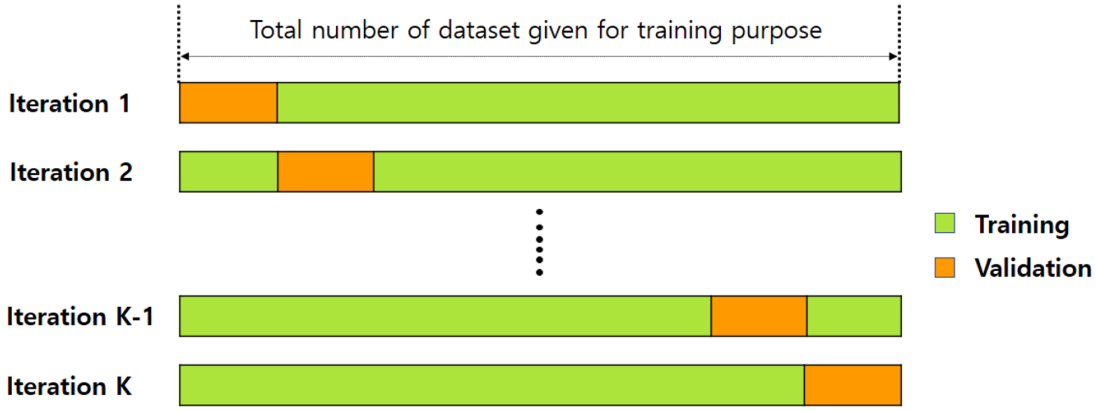


Figure 4.3.1: A k-fold cross validation scheme.

of those training samples. At this point, the output scores of PHOCNet are explicitly biased, because input images are already used to train PHOCNet. Then, those biased outputs of PHOCNet are used for training CCA. Hence, from the standpoint of the CCA, the training samples of its learning are severely biased, and those biased samples can not provide enough information for the learning enough to generalize to the test.

One of regularization techniques called cross-validation is considered as the solution to address above overfitting problem. As illustrated in figure 4.3.1, the idea of a k-fold cross-validation is to split the entire dataset into training set and validation set so that the model can be validated by unobserved samples during training. A training and validation process iterates k times for different configurations of samples until it covers entire data set. Then, a global model can be obtained by averaging all k models. The usage of cross-validation in this thesis is as follows. For each iteration, samples in the training split are used for training PHOCNet, and then samples in validation split are fed into trained PHOCNet. The output of the PHOCNet for that input is used for training CCA. After all iterations, each parameter of k different PHOCNet is averaged to yield global model. However, in general, the training of a deep convolutional network such as PHOCNet requires huge computational resources and time. According to the training details in [SF18], they run training 80 000 iterations or more for different settings. Not only is the number of iteration high, but there are many parameters that need to be updated in each iteration. Hence, with considering these limitations, the application of cross-validation is not a practical solution.

4.3.2 *CCADAP*

As well as the practical limitations mentioned above, the cascaded use of different machine learning models can occur trade-off problem between minimizing training error of preceding model and minimizing the test error of subsequent model. In this case, the capability of minimizing error between image of training sample and its label means that the model is able to give extremely biased scores for the images already learned. On the other hand, the subsequent model, CCA, requires a large number of unbiased samples of attribute scores to exploit large variation of attribute scores possibly estimated from upcoming test images. Therefore, the better the classifier is used to estimate the score, the less information from training samples the CCA can learn. No matter how many word images are provided to train PHOCNet, the CCA can be suffered from the lack of training samples which possibly lead to inferior recognition results because of generalization error.

To overcome above problem, this chapter proposes CCADAP which learns common subspace from between test images and their temporal classes rather than between training images and their ground-truth labels. An overview of CCADAP is illustrated in figure 4.3.2. It starts with training the CCA on the test document images instead of training document images. In this context, the test document is the set of segmented handwriting words to be recognized. Therefore, CCADAP requires an assumption that recognition systems process several words at once, not just one word for each time during test. However, it is reasonable assumption in many document image analysis applications which do not need to recognize each word independently. In this case, different to the conventional use of CCA (4.3.10), the learning requires ground-truth labels of test images. To know a ground-truth of upcoming input is impossible, and therefore this scenario has a explicit contradiction. To tackle this problem, CCADAP uses predicted classes of input images by DAP instead of using unknown ground-truth labels. Given test documents, the recognition system first predicts temporal classes of words with DAP. From those predicted classes, binary PHOCs are encoded from their transcription. Let $A_y^{pred.} \in \{0,1\}^{Q \times M}$ be constructed by those binary attribute representations of temporal classes, then the objective of learning of CCADAP is given by

$$w_x^{test}, w_y^{temp.} = \underset{w_x, w_y}{\operatorname{argmax}} < A_x^{test} w_x, A_y^{pred.} w_y > \tag{4.3.13}$$

where w_x^{test} and $w_y^{temp.}$ represent canonical weights which leads a maximum correlation between A_x^{test} and $A_y^{pred.}$ on their common subspace. After finding such weights,

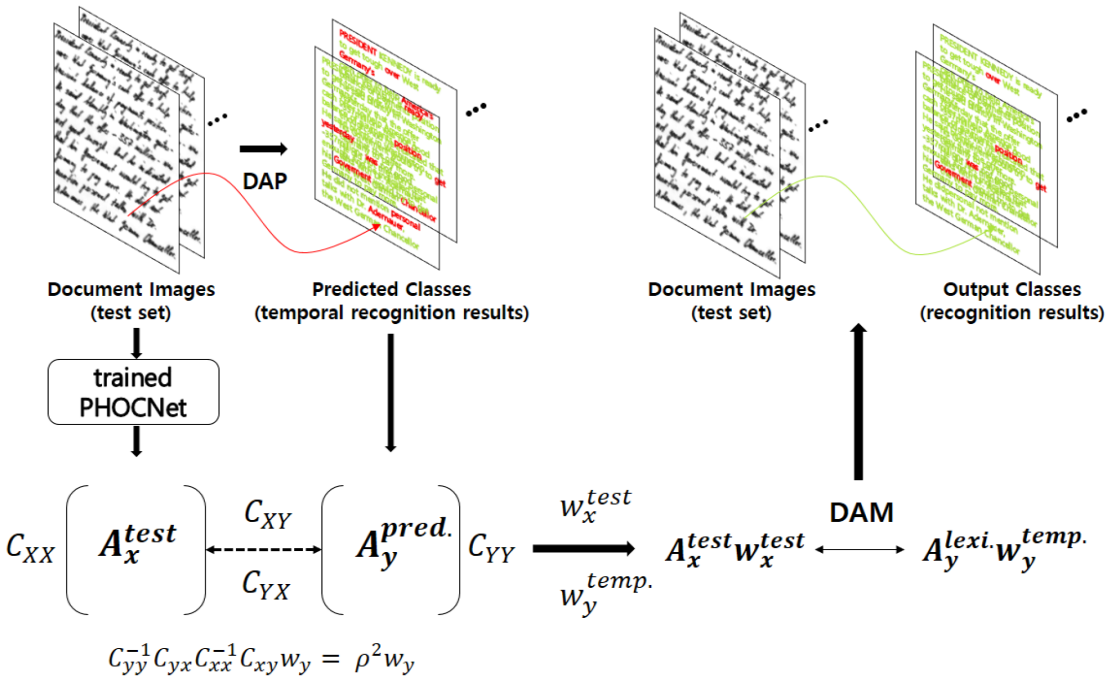


Figure 4.3.2: Overview of the CCADAP.

remaining process of recognition is same as the process explained in section 4.3.1. At first, A_x^{test} and $A_y^{lexicon}$ are transformed into new representations by learned weights w_x^{test} and $w_y^{temp.}$ respectively. Then, the recognition task is reduced to find nearest neighbour of those new representations:

$$\vec{o}^* = (o_1^*, ..., o_Q^*) = \underset{l=1,...,L}{\operatorname{argmin}} \, d_{cos}\left(A_x^{test} w_x^{test}[q] \, , \, A_y^{lexi.} w_y^{temp.}[l] \right). \tag{4.3.14}$$

Thus, CCADAP is a recognition technique to find transcriptions of handwriting word images in documents by reading them multiple times. It is close to handwriting word recognition process of human, because human also can improve recognition accuracy by repetitive reading. For instance, when handwriting documents are given to a person, a person provisionally determines the transcription of words at the first reading, and then learns the relation between handwriting words and his belief from first reading. Based on the learning, a person can change his mind to correct recognition from incorrect prediction in his second reading. From the viewpoint of machine learning,

there is no generalization problem for the CCADAP. Another advantage is that, the performance of CCADAP is proportional to that of the PHOCNet. As discussed in previous section 4.3.1, well-trained multi-label classifier can cause overfitting problem to CCA by extremely biased scores. However, In CCADAP, bias of estimated scores is advantageous for first prediction by DAP, And after all, CCADAP can offer better recognition result than such DAP, because CCAPDAP exploits and learns additional relation between data from the result of DAP. One of the limitations of CCADAP is that CCA possibly learns wrong correlations by incorrect labels when preceding classifier gives too imprecise scores and results in too wrong predictions for DAP. However, the state-of-the-art retrieval results with PHOCNet in the expriments of [SF18] imply that trained-PHOCNet can successfully estimate attribute scores. And therefore, in this thesis, CCADAP assumes that PHOCNet shows stable performance enough to learn proper correlations from predicted classes of DAP, even if there are small fraction of wrong predictions by DAP.

4.4 MLPS CLASSIFIER WITH PSEUDO SAMPLES

A MLP classifier with softmax log-loss described in section 2.1.1 is a multi-class classifier. Considering the use of this MLP classifier in the recognition system discussed in this thesis, the input layer receives attribute scores and one of lexicon words is determined as output throughout a hidden layers as depicted in figure 4.4.1. Therefore, with respect to training such network, the input pattern is the output of the PHOCNet for a word image. The ground-truth label is the one-hot encoded index of that word in lexicon. However, when this MLP classifier is cascaded to the PHOCNet, overall recognition system no longer has zero-data learning capabilities. The reason is that MLPs utilize a supervised learning technique called backpropagation as introduced in section 2.1.5. In addition to that, following the definition of attribute-based classification specified in section 3.1, the proposed recognition system belong to neither zero-data learning nor attribute-based classification, because it never use class description to transfer knowledge as well as never use attributes of a class. Hence, the conventional use of MLP classifier in this recognition system cancels out the benefits of using attributes.

Based on the idea of attribute-based classification in section 3.1, in order to make a classifier applicable for attribute-based classification, the description of the classes should be reflected to the classifier in the process of learning. To achieve such requirement, this chapter proposes training MLP classifier with pseudo score samples briefly called pseudo samples in the rest of this thesis. The idea is to generate highly

likely attribute score samples for zero-data classes in order to train the MLP for those classes. Specifically, to generate such samples, the distribution of each attribute score is estimated from the output of the PHOCNet. Because the attribute in this work represents the presence and spatial position of characters, score distribution of all attributes can be computed from training samples. After estimating the distribution of all possible state of all attribute, pseudo samples for zero-data classes are generated by random sampling from those distribution. As depicted in figure 4.4.2, true or false state of binary PHOC of zero data class is observed first. After that, a probable score value is picked up by random sampling from aligned distribution, and the score value is then assigned to a pseudo sample. Finally, the MLP classifier is trained by the generated pseudo samples as well as given training samples. The detailed process of estimating attribute score distribution and generating pseudo samples is described in algorithm 2

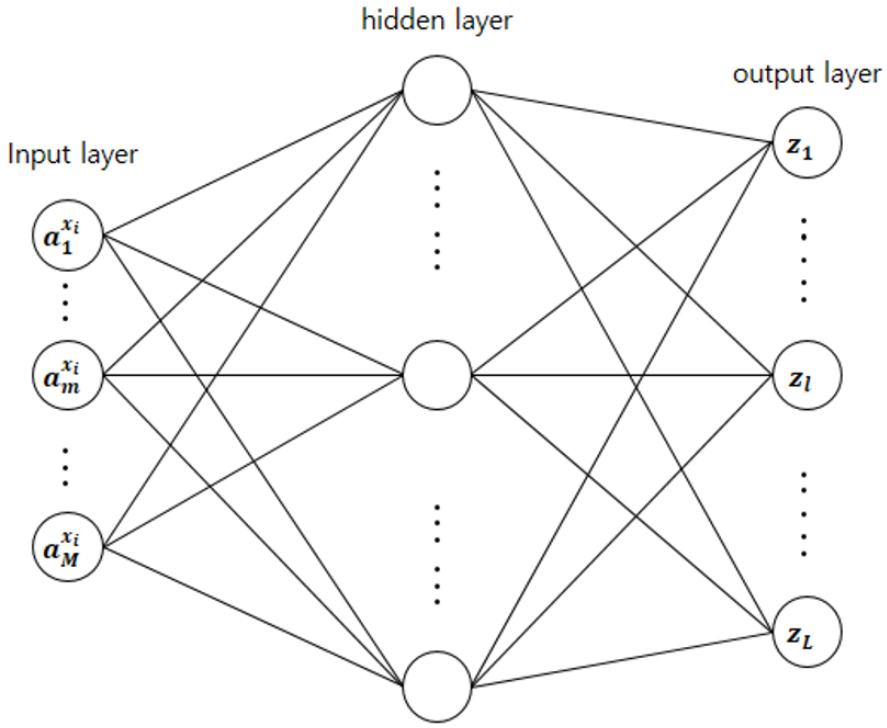


Figure 4.4.1: Graphical representation of the mlps classifier where the input is attribute scores and the output is words in the lexicon.

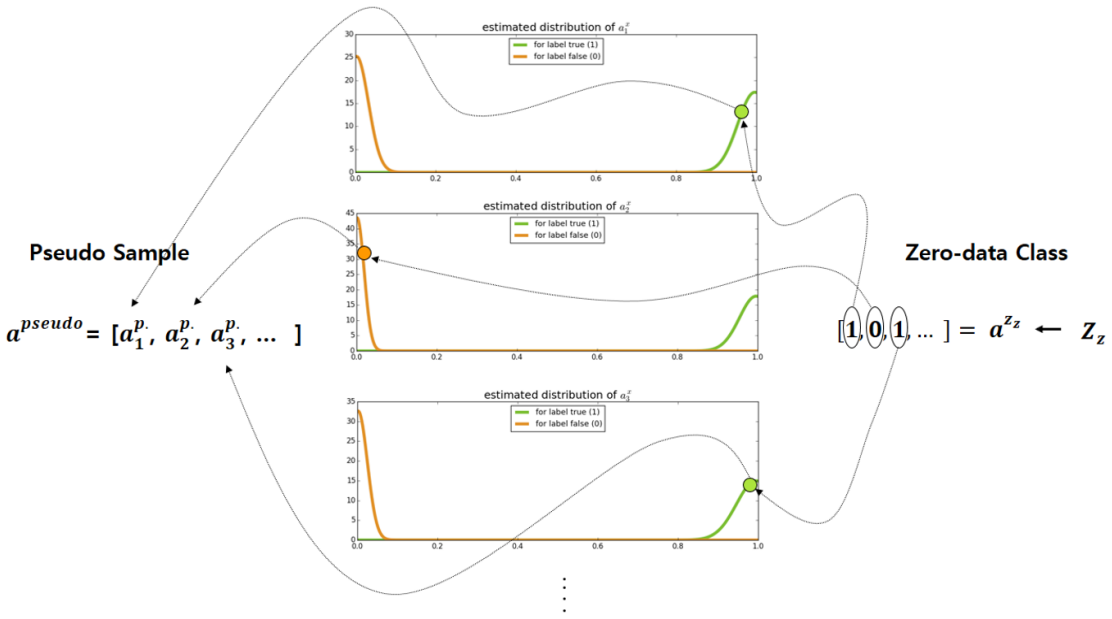


Figure 4.4.2: The visualization of the process of generating pseudo samples.

Using proposed pseudo samples enables the MLP classify input attribute scores into zero-data classes for the following reasons. At first, for the attribute-based classification, attributes are used to transfer information between classes. In this case, attributes are also used to transfer knowledge from trained classes to zero-data classes. The reason is that the score distribution is estimated from the samples of trained classes. Then the knowledge is transferred to zero-data classes by training MLPs with pseudo scores sampled from such a distribution. Second, although severely biased output of trained PHOCNet leads to overfitting problem for CCA as explained in section 4.3.1, it helps ensure reliability of pseudo samples in this case. According to the assumption specified in section 4.3.1, attribute scores estimated by the PHOCNet for trained samples have severely biased mean and have very small variances. Therefore, sampled values for pseudo scores from such distribution is highly likely to the expected output of the PHOCNet for the input of zero-data classes. In other words, if the output of the PHOCNet is not biased and distributed over wide range, then the generated pseudo samples cannot represent the expected output of the PHOCNet. However, similar to the argument in section 4.3.1, the biased output of the PHOCNet can cause generalization problems for the MLP classifier. To suppress the problem, pseudo scores are sampled from a normal distribution with estimated mean and variance rather than

one of the scores from training samples directly. This approach can be categorized as data augmentation introduced in section 2.1.6, and a large number of pseudo samples can be generated from such normal distribution without using computational resources. Then, training the network with those large number of samples from normal distribution helps its learning generalize to upcoming test inputs.

Algorithm 2 : Estimating attribute score distributions and generating pseudo samples.

Initialization:

Attribute score and label matrix : $A_x^{train}, A_y^{train} \subset \mathbb{R}^{N \times M}$

Zero-data class :$z_z \in Z^{zerod.} \subset Z^{lex.}$

Pseudo sample :$a^p = zeros(1, M) = [a_1^{p\cdot}, ..., a_M^{p\cdot}]$

Estimating attribute score distribution:

for m in $range(M)$:

$TrueAttrIdx = A_y^{train}(:, m)$

$FalseAttrIdx = 1 - A_y^{train}(:, m)$

$a^{one}, a^{zero} = \langle A_x^{train}(:, m), TrueAttrIdx\rangle, \langle A_x^{train}(:, m), FalseAttrIdx\rangle$

$\mu^{one}, \sigma^{one} = mean(a^{one}), std(a^{one})$

$\mu^{zero}, \sigma^{zero} = mean(a^{zero}), std(a^{zero})$

$X_m^{one} \sim \mathcal{N}(\mu^{one}, (\sigma^{one})^2), X_m^{zero} \sim \mathcal{N}(\mu^{zero}, (\sigma^{zero})^2)$

return X^{one}, X^{zero}

Generating pseudo sample:

for z_z in $Z^{zerod.}$:

for $a_m^{z_z}$ in a^{z_z}:

if $a_m^{z_z} == 1$:

$a_m^{p\cdot} = random.sample(X_m^{one})$

else :

$a_m^{p\cdot} = random.sample(X_m^{zero})$

return $a^{p\cdot}$

5 EXPERIMENTS

In this chapter, the recognition systems proposed in chapter 4 is evaluated on commonly used handwriting recognition benchmark. The results of different methods are not only compared with each other but compared to other works [AGFV14], [PW16]. The dimensionality of attribute representation is fixed to 540 for all experiments. The length of $36 \cdot (1+2+3+4+5) = 540$ is defined by 5 levels unigram of 26 English alphabet and 10 digits. In the first section, the database used in this thesis, IAM, is introduced. Then, the evaluation metrics for the experiments, Word Error Rate(WER), Character Error Rate(CER) and Out of Vocabulary-WER(OOV-WER) are defined. In addition, the training details of the PHOCNet is specified. Lastly, under specified protocols in section 5.2, the results of all the methods proposed in chapter 4 on the dataset in section 5.1 are presented and compared.

5.1 DATASET

The dataset used in this thesis is IAM which is widely used in the document analysis. The IAM off-line handwriting database [MB02] includes 115,320 isolated and labeled words of mostly cursive English text written by 657 different writers. The database offers writer independent official partition of train, validation and test. This official partition will be used throughout the experiments, since it is widely used and eases comparison with results from the literatures. With regard to the lexicon, a closed lexicon which contains all the words that appear in either train or test set is used, analogous to other works, [AGFV14], [PW16]. In other words, the lexicon in this thesis is defined by getting unique transcription from all words in the IAM. The number of classes in such lexicon is 9,271. Out of Vocavulary (OOV) words refer to test words of zero-data classes in the lexicon, and 1,746 OOV samples appears out of 11,601 test samples.

5.2 PROTOCOL

5.2.1 *Evaluation Metrics*

In the experiments, Word Error Rate(WER) and Character Error Rate(CER) are used to assess the performance of recognition systems. These two metrics are standard measures for determining word recognition performance. As shown in its terminology, WER refers the number of Word Error(WE) out of entire test samples:

$$WER(\%) = \frac{\#WE}{\#\text{test samples}} \times 100 \tag{5.2.1}$$

where WE means that the output class of the word image is different to its ground-truth label. Character Error (CE) is measured between the transcription of the output class and ground-truth label of test input image. To define CER, a distance metric between two strings called Levensthein distance is employed. It measures the distance by counting the minimum number of character substitutions, deletions, and insertions to transform an output string into the ground-truth label. Then, the CER of a word is obtained by normalizing distance with the length of the word. Finally, the CER of the entire test set is defined as the average of them.

As discussed in section 3.1, word recognition can be considered as a special case of zero-data classification. One reason to use attribute-based approach is to make the recognition system classify future samples of classes for which no training data is available, i.e., zero-data classes. In this context, WER cannot fully assess the performance of attribute-based word recognition systems. The reason is that WER cannot distinguish whether word errors result from samples of trained classes or result from OOV samples of zero-data classes. Therefore, to evaluate the performance of recognizing OOV words, OOV-WER is additionally used in this thesis as evaluation metric. This OOV-WER indicates the number of word error among OOV over the entire number of OOV words in the test set:

$$\text{OOV-}WER(\%) = \frac{\#(WE \cap OOV)}{\#OOV} \times 100 \tag{5.2.2}$$

Therefore, OOV-WER is 100% under standard supervised learning.

5.2.2 *Training Details of PHOCNet*

All proposed methods in chapter 4 employ a trained PHOCNet to estimate attribute scores of input images. Training details of the PHOCNet in this thesis are similar to those in [SF18] that firstly introduces PHOCNet for word spotting in handwritten documents. The ground is that the PHOCNet is used to estimate attribute scores from input word image in both [SF18] and this thesis. The network is trained using a binary logistic-loss which is derived in (3.2.7). For optimizing loss function, Stochastic Gradient Descent (SGD) is employed with using a momentum of 0.9 and batch size of 10.

As shown in the architecture of the PHOCNet in figure 3.2.1, it contains huge number of parameters that have to be trained especially in between fully connected layers. On the other hand, there is only a limited number of training samples in given dataset. Therefore, overfitting can easily occur in the PHOCNet. Three regularization techniques introduced in section 2.1.6, dropout, data augmentation and weight decay, are used to prevent overfitting in this thesis. Dropout is applied to fully connected layers with probability of 0.5 except last two layers which contain the same number of neurons as the length of attributes, 540. With respect to image augmentation, a random affine transform is applied to randomly sampled word images per class. In

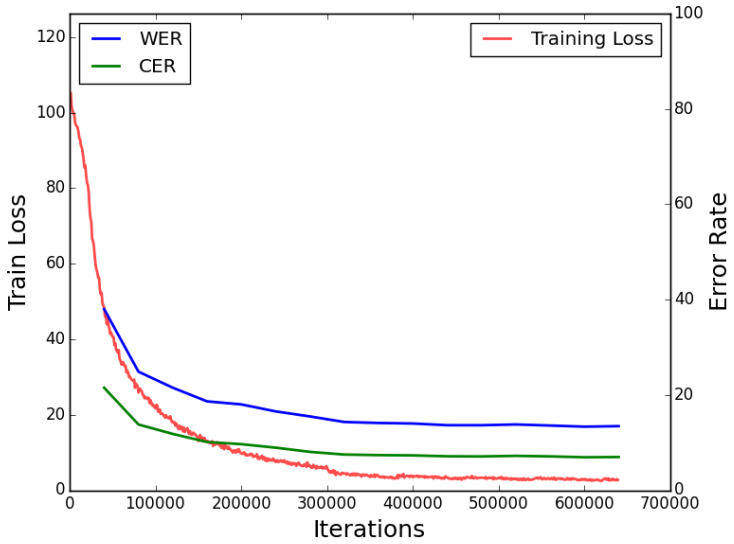


Figure 5.2.1: The train loss and Error Rates(ERs) over training iterations of the PHOCNet on the IAM dataset.

detail, three points in the middle of the sampled image are obtained by selecting fixed relative coordinates. Then, each of taken coordinates is multiplied by a random number uniformly sampled from [0.8,1.1]. From the first points, the homography is computed as new coordinates of the corresponding first points in a synthetic image. An affine transformation derived from above steps is applied to generate augmented images. The number of training images per class set to 50 with image augmentation. Lastly, for the weight decay, the L2 penalty multiplier set to $5 \cdot 10^{-5}$ which is same value used in [SZ14] because of the similarity of network architecture.

Training is run for 640,000 iterations with an initial learning rate of 10^{-4}, and the learning rate is divided by 10 every 200,000 iterations. For word spotting in [SF18], a maximum number of iterations of 240,000 is used on the IAM. In the case of recognition in this thesis, the training continues after 240,000 iterations, because training loss continues to decrease beyond 240,000 iterations without increasing generalization error as shown in figure 5.2.1. Even though there is still no problem of increasing ERs beyond 640,000 iterations, both train and test errors are already converged around 480,000 iterations. Hence, when a maximum number of iterations is larger than 480,000, it does not significantly affect the recognition results.

5.3 RESULTS

5.3.1 *DAM*

As explained in section 4.1, DAM refers to a multi-class classification method that finds nearest neighbours with distance measurement between two different attribute representations. In the experiments, test samples are fed into a trained PHOCNet to estimate their attribute scores. Then, DAM method finds the nearest word class in the lexicon to estimated scores of test samples by computing (4.1.4). DAM is the most

Method	**Specification**	**Results(%)**		
		WER	**CER**	**OOV-WER**
DAM	Cosine	13.93	7.20	19.41
	Bray-Curtis	13.72	7.09	18.90

Table 5.3.1: Recognition error rates using different configurations for DAM method on the IAM dataset.

naive of the methods proposed in this paper, and requires much less computation time than others. For that reasons, recognition errors using the DAM method shown in the table 5.3.1 are used as baseline results. Experimental results in table 5.3.1 show that the Bray-Curtis (BC) dissimilarity yields lower error rates than the case of using cosine distance. However, in this chapter, cosine distance is considered mainly as metric for DAM to compare results with other works as shown in table 5.3.2, because they use only cosine distance as metric. The results in table 5.3.2 represent the performance of different method of estimating scores. The reason is that all given experiments use DAM method to find output class on the IAM dataset. For the experiment of [AGFV14], they uses FVs with SVMs to estimate attribute scores, and then they find nearest neighbour class based on cosine distance. [AGFV14] only introduces their best recognition results with KCSR which is close to kernel CCA [Aka01]. [AGFV14] claim that KCSR clearly outperforms CSR, i.,e, CCA, because a kernalized approach enables to analyse non-linear relation between attribute scores and binary PHOCs. The first and third experiments in the table 5.3.2 which use a deep CNN as multi-label classifier give better results than [AGFV14] even without CCA. It implies that using one integrated multi-label classifier, a CNN, is a better way to estimate attribute scores from visual pattern of image for the recognition than using separated classifier per attribute with independent feature vector encoding proposed in [AGFV14]. The experiment in [PW16] which employs different CNN architecture called CNN-N-Gram shows better result than experiment of using the PHOCNet. The difference possibly comes from enlarged network architecture of CNN-N-Gram with branching fully-connected layers described in section 3.2.2.

Method measuring scores	**Multi-class classifier**	**Specification**	**Results(%)**	
			WER	**CER**
PHOCNet	DAM_{cos}	withoutCCA	13.93	7.20
FVs with SVMs [AGFV14]		KCSR	20.21	11.27
CNN-N-Gram [PW16]		withoutCCA	8.83	5.93

Table 5.3.2: Comparison of results using DAM method with different configurations on the IAM dataset.

5.3.2 *DAP*

The experiments with DAP is to compute probabilistic model equation (3.3.12) derived in section 3.3. Considering computability of joint probability $p(a^{z_l})$, attributes are assumed to be statistically independent to each other as denoted in (4.2.11). Then, the baseline DAP model which is used in the following experiments is given by

$$\hat{z}_{MAP}(x_i) = \underset{l=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)}{\prod_{m=1}^{M} p(a_m^{z_l})} \tag{5.3.1}$$

As discussed in section 4.2, $\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i)$ is determined by the output of a trained PHOCNet. Therefore, the variants of DAP experiments come from different assumptions for class prior $p(z_l)$ and attribute prior $p(a_m^{z_l})$.

The first result of DAP in table 5.3.3 is induced by assuming uniform distribution for $p(z_l)$ and unbiased for $p(a_m^{z_l})$ which are given by (4.2.5) and (4.2.13) respectively. With these assumptions, the output class of DAP is only determined by the output score of the PHOCNet, because $p(z)$ and $p(a_m)$ are same for all classes:

$$\hat{z}_{MAP}(x_i) = \underset{i=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)}{\prod_{m=1}^{M} p(a_m^{z_l})} = \frac{c_1}{c_2} \underset{i=1,\dots,L}{\operatorname{argmax}} \prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) \tag{5.3.2}$$

where $c_1 = \frac{1}{L}$ and $c_2 = \left(\frac{1}{2}\right)^M$. Even though (5.3.2) is the most simplified form of DAP, it already gives better WER and CER than DAM as shown in table 5.3.3. As

Method	**Specification**	**Results(%)**		
		WER	**CER**	**OOV-WER**
DAM	Cosine	13.93	7.20	**19.41**
DAP	p(z) : uniform $p(a_m)$: unbiased	12.79	6.79	21.53
	p(z) : empirical $p(a_m)$: unbiased	**11.89**	**6.21**	22.45

Table 5.3.3: Results for the DAP under two different assumptions comparing with cosine DAM.

Method	p(z)	Results(%)	
		WER	CER
DAP	uniform	12.79	6.79
	empirical	11.89	6.21
	DAM results	11.86	6.19

(a) Results for DAP with respect to different $p(z)$

Method	$p(a_m)$	Results(%)	
		WER	CER
DAP	unbiased	11.89	6.21
	empirical	16.31	9.73

(b) Results for DAP with respect to different $p(a_m)$

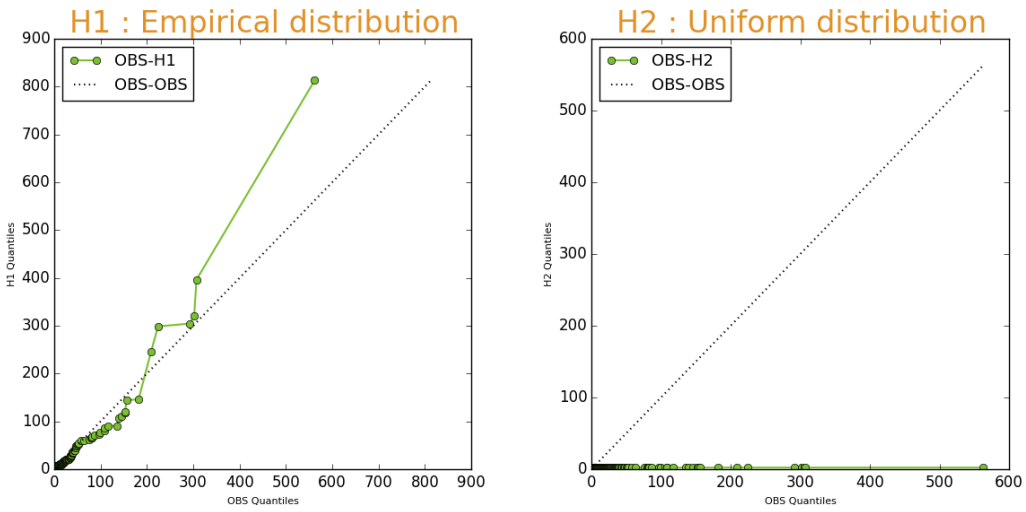
Table 5.3.4: Results for the DAP under different assumption for either $p(z)$ (a) or $p(a_m)$ (b).

discussed in section 4.2, a very likely reason is that DAP exploits all scores of both true and false attributes, whereas DAM only considers true attributes.

The second scenario is to assume that class prior follows smoothed empirical distribution derived by (4.2.6) and (4.2.9), but $p(a_m)$ is still assumed to be unbiased:

$$\hat{z}_{MAP}(x_i) = \frac{1}{c_2} \underset{i=1,\dots,L}{\operatorname{argmax}} \prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)_{train} \tag{5.3.3}$$

According to second result of DAP in table 5.3.3, using $p(z_l)_{train}$ gives better WER and CER than assuming uniform distribution. It implies that $p(z_l)_{test}$ is close to $p(z_l)_{train}$. It can be demonstrated by Chi-Square goodness of fit test χ^2. Let observation O is DAM recognition result that has about 86% word accuracy from the table 5.3.3, and first and second hypotheses, H1 and H2, are empirical and uniform distribution



(a) OBS-H1 plot (b) OBS-H2 plot

Figure 5.3.1: Q–Q plots comparing an observed word frequency by DAM with a hypothesized distribution

respectively. In this Chi-Square test, DAM recognition result of 86% word accuracy is assumed to be enough to estimate word frequency of test set. Then χ^2 is given by

$$\chi^2 = \sum \frac{(O-E)^2}{E} \tag{5.3.4}$$

where E represents an expectation asserted by hypothesis H. The computed value of (5.3.4) with each hypothesis is compared with a chi-squared distribution to evaluate the goodness of fit:

$$\begin{aligned} \chi^2_{H1} &= 915 < 36028, \\ \chi^2_{H2} &= 448713 > 36028 \end{aligned} \tag{5.3.5}$$

where 36028 is value of the Chi-Squared distribution with $L-1$ degree of freedom and 0.05 significant level. Chi-Squared test statistic of first hypothesis χ^2_{H1} is smaller than critical value 36028. It indicates that empirical distribution is acceptable, and the frequency of words in train set is close to test set. However, hypothesis of uniform distribution, yields much larger difference statistics χ^2_{H2} than critical value. This means that term frequency of test is not uniformly distributed, and therefore H2

is not acceptable. These closeness of between distributions can be visualized by a Quantile-Quantile (Q-Q) plot as depicted in figure 5.3.1. In Q-Q plot, the linearity of the points on the 45° line suggests that the data obtained by hypothesis is close to observation. Therefore, the left sub-figure proves that word distribution in test documents generally follows empirical distribution of train set except a few outliers. Contrary to (a) of figure 5.3.1, (b) shows that observed word frequency in the test is completely different to uniform distribution. Hence, on the IAM dataset, the word frequency of train set help predict the number of appearances of word in the test. As a result, using empirical distribution for p(z) gives better recognition result than ignoring it by assuming uniform. However, $p(z)_{test} \sim p(z)_{train}$ on the IAM dataset cannot always generalize to other applications. It can be argued that above assumption possibly gives inferior results than using uniform distribution, depending on datasets or test documents. There are two solutions to address above limitations. First of all, it is possible to prepare a number of p(z) distributions depend on document categories. As specified in (4.2.6), p(z) refers term frequencies (TF) which are used to categorize a text document in bag-of-words model. Therefore, assigning the most likely predefined TF histogram as p(z) depend on document categories can help the recognition task of DAP. If no prior knowledge of the category of documents is given, then TF can be constructed from output classes of DAM recognition. Although the result of DAM is different to ground-truth label of test images, it can offer comprehensive information about TF of test documents. The result of table 5.3.4 (a) shows that this solution gives better recognition results than ignoring class prior $p(z)$.

On the other hand, employing empirical distribution for $p(z)$ gives inferior results in OOV-WER. It is because one-additive smoothing enables DAP avoid the problem of zero posteriors but still assumes relatively very small prior for zero-data classes. In other words, weighting factors for computing posterior written in (5.3.3) set to the smallest value for zero-data classes. Thus, using word frequency information of train set can help reduce WER and CER, but can be disadvantageous for recognizing OOV words.

Up to this point, $p(a_m)$ is assumed to be unbiased for all m. As derived in (4.2.14), empirical approach also can be applied to $p(a_m)$. Let class prior $p(z)$ be assumed to follow empirical distribution. Then, DAP with empirical approach for both $p(z)$ and $p(a_m)$ is given by

$$\hat{z}_{MAP}(x_i) = \underset{i=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)_{train}}{\prod_{m=1}^{M} p(a_m^{train} = a_m^{z_l})} \tag{5.3.6}$$

with M=540 different $p(a_m^{train} = 1)$ defined in (4.2.14). However, as shown in table 5.3.4 (b), the result of experiment with (5.3.6) inferior to the result of neglecting $p(a_m)$ with unbiased probability, 0.5. In this case, it is hard to analyse the reason for increasing error rates. Different to the case of $p(z)$ which estimates only one distribution, M number of fitness between $p(a_m^{test} = 1)$ and $p(a_m^{train} = 1)$ should be considered. Furthermore, it is hard to conclude that which of $p(a_m)$ largely affects word errors, because all M estimated $p(a_m)$ are multiplied. Meanwhile, when $p(a_m^{test} = 1)$ is assumed to be observed, WER of this ideal case is even worse than 16.31% of WER in table 5.3.4 (b). This implies that assuming independence of attributes in DAP model hurts the recognition performance expect for unbiased attributes:

$$\hat{z}_{MAP}(x_i) = \underset{i=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)}{p(a_1^{z_l}, \dots, a_M^{z_l})} \neq \underset{i=1,\dots,L}{\operatorname{argmax}} \frac{\prod_{m=1}^{M} p(a_m^{z_l} \mid x_i) p(z_l)}{\prod_{m=1}^{M} p(a_m^{z_l})} \tag{5.3.7}$$

To sum up: two assumptions, empirical distribution for $p(z)$ and unbiased for $p(a_m)$, gives the best WER and CER in DAP experiments. Hence, the second result of DAP in table 5.3.3 is considered as baseline result of DAP in the rest of thesis.

5.3.3 *CCADAP*

Table 5.3.5 compares the result of CCA to DAM. CCA method offers slightly better error rates than DAM with the optimal set of hyperparameters derived by algorithm in section 1. Please note that CCA method in the rest of thesis refers regularized CCA which is explained in section 2.2.3. As denoted in section 4.3.1, even though there are a large number of training samples on the IAM dataset, CCA can be ill-posed in underdetermined setting. It is because most of attributes are not only co-linear with level-wise representation of PHOC, but also biased by the PHOCNet. For the specification of CCA in section 5.3.5, $r_1 = r_2 = r$ refers the dimensionality of new representation on the common subspace after applying CCA, and $\lambda_1 = \lambda_2 = \lambda$ denotes regularisation parameters. This very large value of parameters hinders improving performance of CCA. As discussed in section 2.2.3, non-negative regularisation parameters λ is used to avoid singular problem of covariance matrices in regularized CCA. However, large parameters which are needed to suppress overfitting dominates learning of CCA. On account of this limitation from regularization, DAP method can achieve better recognition results than CCA in WER and CER. Whereas CCA shows better recognition for OOV than DAP, the standard measurements, WER

Method	Specification	Results(%)		
		WER	CER	OOV-WER
DAM	Cosine	13.93	7.20	19.41
CCA	r : 150 $\lambda : 10^6$	13.61	6.97	18.55
DAP	p(z) : empirical $p(a_m)$: unbiased	11.89	6.21	22.45
CCADAP	r : 120 $\lambda : 10^3$	**10.06**	**5.03**	**16.66**

Table 5.3.5: Comparison of the best results of each of four methods.

and CER, explicitly have higher priority to evaluate recognition performance of methods. Hence, in the experiments of this thesis, using CCA is inferior option than using DAP in attribute-based recognition.

As listed in table 5.3.5, CCADAP achieves 10.06% WER, 5.03% CER and 16.66% OOV-WER that are lower error rates than results of any other methods presented previously in this thesis. For the temporal reading of CCADAP, the result of DAP in table 5.3.5 is used. Since CCA learns the correlation between attributes rather than words, CCA can learn proper relations from correct attribute predictions of error words. This means that the performance of CCADAP is largely affected by CER rather than WER of DAP. In this case, from 6.21% CER of DAP, CCADAP can learn correlation from 93.79% accurate temporal learning by DAP. In addition, the dimensionality of new representation by CCADAP is set to 120 which is much smaller than 540, but CCADAP still gives superior results after a large degree of dimensionality reduction. Because CCADAP learns relation from temporal predictions of test documents instead of training set, the ovefitting problem in the CCADAP is not as decisive as in the CCA. Therefore, regularisation parameters λ are set to 10^3 which is much smaller than 10^6 of CCA. Compared to DAP, the upper bounds of error rates of CCADAP are those of DAP that are already superior than DAM. Especially, CCADAP reduces OOV-WER by about 6% over DAP method. Indeed, CCADAP compensates increasing OOV-WER problem of DAP.

One limitation of CCADAP is the value of Q. Here, as defined in section 4.3.1, Q indicates the number of input image that goes into the recognition system at one time during test. Therefore, CCADAP requires proper size of Q enough to learn the correlation by CCA. For the CCADAP in table 5.3.5, Q is set to be same as the number of test samples in the IAM data set, 11,601. This is the best scenario to use CCADAP on the IAM dataset, so that CCADAP fully exploits test samples for its learning. In order to assess the performance of CCADAP according to the value of Q, 6 test batches of different sizes are constructed by random sampling from IAM test set. Table 5.3.6 shows that results of three methods for different batch size, and is visualized by the graph in figure 5.3.2. Because test batches includes randomly selected samples from entire test set, three batches are generated per size. Then, the results are averaged for each size to increase reliability of experiments. Please note that the comparison of three different methods for the value Q is only matter in this experiments, and the absolute number of error rates in table 5.3.6 are not related to the objective of experiments. When Q is smaller than the number of attributes, CCADAP does not need to be considered because CCA is explicitly ill-posed in underdetermined setting. In this case, CCADAP determines its output as temporal results by DAP. Then, the recognition results are same as DAP. When Q is same as M=540 or slightly larger than M, CCA is applicable for the results of DAP. However, the improvement is still trivial,

Q	**ratio**	**Results(%)**					
		DAM		**DAP**		**CCADAP**	
		WER	**CER**	**WER**	**CER**	**WER**	**CER**
116 $<$ M	1%	12.64	6.35	10.62	4.88	10.62	4.88
540 $=$ M	5%	13.08	6.64	11.47	6.24	11.10	5.91
812 $>$ M	7%	13.74	6.85	11.9	6.13	11.07	5.56
2320 $>$ M	20%	14.62	7.42	12.22	6.20	11.21	5.62
5800 $\gg$ M	50%	13.83	7.14	11.75	6.17	10.29	5.06
11601 $\gg$ M	100%	13.93	7.20	11.89	6.21	10.06	5.03

Table 5.3.6: Comparison of the results of CCADAP and two other methods on Q where Q indicates the number of input images that goes into the recognition system at one time during test.

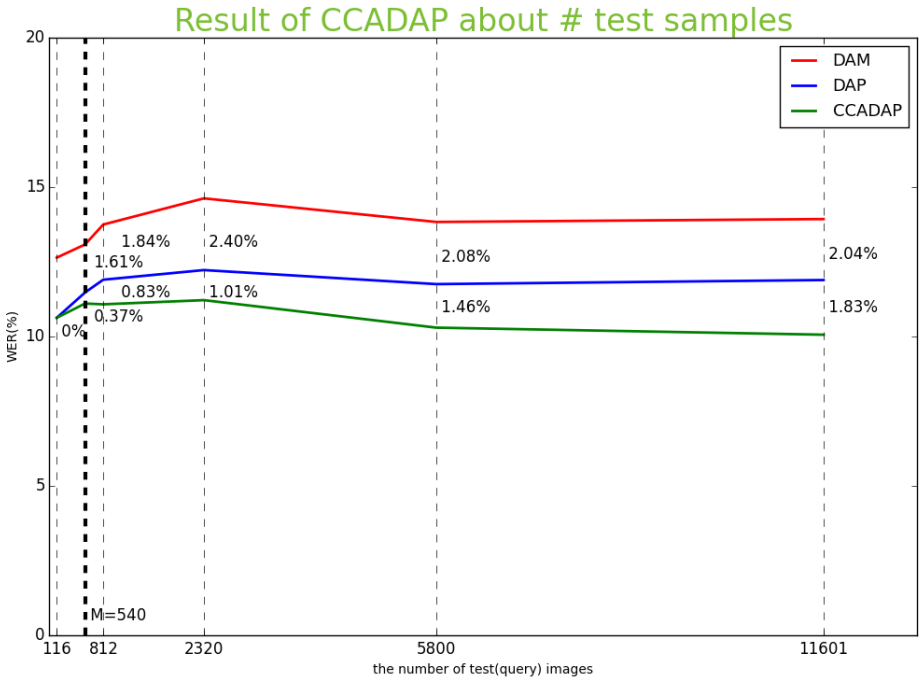


Figure 5.3.2: The graph of recognition errors for CCAPDAP over different number of test samples Q randomly sampled from IAM test set.

because the learning of CCA from Q number of samples is not enough to correct wrong prediction of temporal results. Afterwards, the performance of CCADAP is proportional to the number of Q. In the figure 5.3.2, the green line representing WERs of CCADAP is always below the blue line of DAP. Furthermore, as the Q value increases, the gap between the two lines also increases. In the best case of entire IAM test set, CCADAP can give about 1.8% lower WER than DAP and about 3.8% lower WER than DAM. Hence, the results of experiments demonstrates that DAP is better method than DAM and CCADAP is superior method than such DAP, regardless of the number of input images.

5.3.4 *Classification with MLPs*

For the MLPs classifier described in figure 4.4.1, its architecture and training details are determined by the results on the validation set with changing hyperparameters.

Method	Number of S^{z_z}	Results(%)		
		WER	**CER**	**OOV-WER**
MLPs	0	24.23	10.49	100
	50	12.44	**6.32**	20.56
	150	**12.17**	6.37	17.64
	500	**12.17**	6.45	**14.66**

Table 5.3.7: Recognition error rates using different number of pseudo samples per zero-data class S^{z_z} for MLPs method on the IAM dataset.

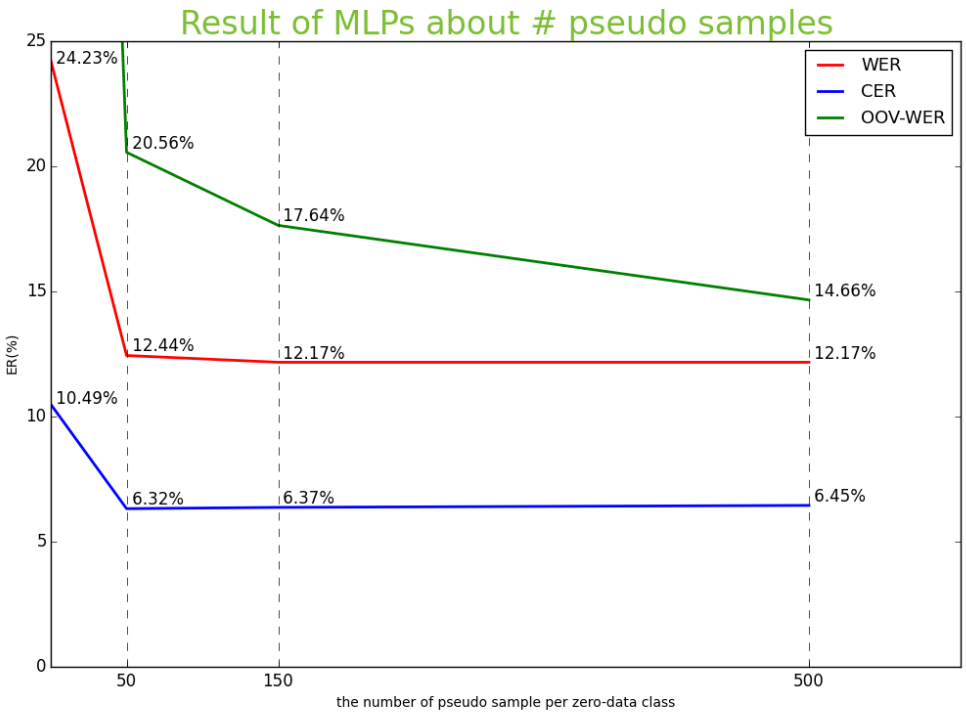


Figure 5.3.3: The graph of recognition errors for MLPs over different number of pseudo samples per zero-data class S^{z_z}.

As specified in section 4.4, the number of neurons in input and output layers are the dimension of attribute representation $M = 540$ and the number of classes in lexicon $L = 9,271$ respectively. Based on heuristics, a hidden layer is defined by one layer of $4 \cdot L \approx 34,000$ neurons. The mini-batch size of SGD sets to 800, and the momentum sets to 0.9. An initial learning rate of 0.1 is used. Then it is divided by 10 when the training error plateaus. The maximum number of iteration for training sets to 10,000 with approximately 500,000 training samples, but the training is only run for about four epochs until training error is converged.

Table 5.3.7 shows the result of MLPs method trained with 50 training samples per data class S^{z_l} and different number of pseudo samples per zero-data class S^{z_z}. As argued in section 4.4, trained MLPs cannot recognize OOV words when pseudo samples are not considered. 100% of OOV-WER results in inferior overall WER and CER to any other experiments. Then, with using pseudo samples to train MLPs, the network begin to recognize OOV words. When each zero-data class is trained by 50 pseudo samples, MLPs can recognize about 79.4% of OOV words, and WER and CER decreases to 12.44 and 6.32 respectively. Then, with using more pseudo samples, the network can recognize more OOV words. In general, to learn classifying OOV words, MLPs requires larger number of pseudo samples per zero-data class than the number of given training samples per class. The reason is that the output scores of the PHOCNet for OOV words have larger variance than scores for images in trained classes. This large variance of scores of OOV inputs is covered by learning based on a large number of pseudo samples. For more than 500 of S^{z_z}, the recognition error rates hardly change. The graph in figure 5.3.3 visualizes the results in table 5.3.7. Below two lines present that WER and CER are not improved beyond 50 of S^{z_z}. On the other hand, the top graph in the figure shows that the performance of recognizing OOV words is proportional to the number of S^{z_z}.

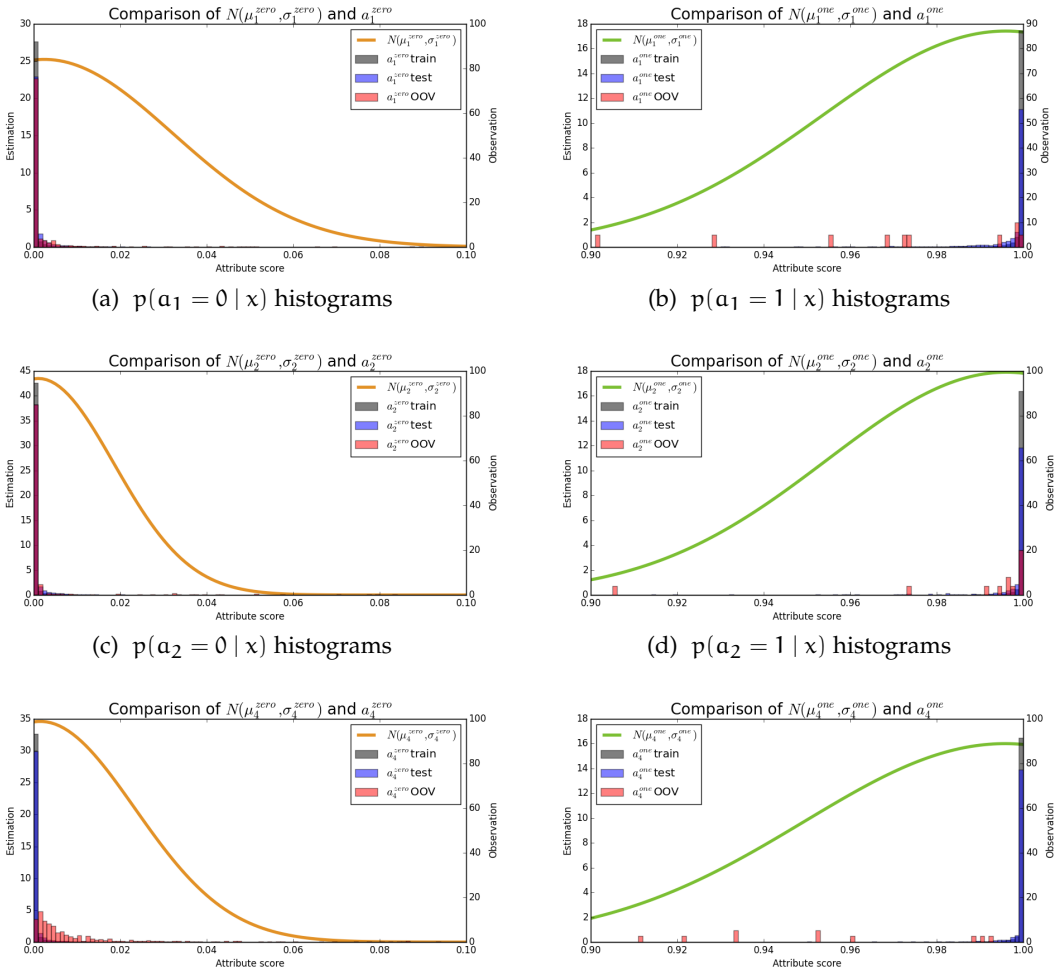
To sum it all up: increasing the number of pseudo samples helps to recognize OOV words, but it can cause similar amount of additional recognition errors for the words of trained classes. Although increasing S^{z_z} does not decrease WER and CER, it is explicitly advantageous to use a large number of pseudo samples. The reason is that lower OOV-WER for the same WER enables the recognition system guarantee stable results for any input documents regardless of the number of OOV words.

Following the algorithm 2, the pseudo samples are sampled from estimated normal distribution $\mathcal{N}(\mu^{train}, \sigma^{train})$ rather than taken from scores of training samples a^{train}. Here, mean and standard deviation of the normal distribution, μ^{train} and σ^{train}, are derived from scores of all training samples. The results of table 5.3.8 shows that sampling from the normal distribution always gives better results than arbitrarily taking one of the scores on the training sample. The histogram in the figure 5.3.4 shows

Method	Number of S^{z_z}	Sampling from	Results(%)		
			WER	CER	OOV-WER
MLPs	50	a^{train}	12.78	6.55	22.45
		$\mathcal{N}(\mu^{train}, \sigma^{train})$	**12.44**	**6.32**	**20.56**
	150	a^{train}	12.29	6.37	18.61
		$\mathcal{N}(\mu^{train}, \sigma^{train})$	**12.17**	**6.37**	**17.64**
	500	a^{train}	12.39	6.52	15.69
		$\mathcal{N}(\mu^{train}, \sigma^{train})$	**12.17**	**6.45**	**14.66**

Table 5.3.8: Comparison of results of different number of pseudo samples sampling from estimated normal distribution $\mathcal{N}(\mu^{train}, \sigma^{train})$ and attribute scores of training samples a^{train}

the reason why above results came out. The data of histograms in the figure is the output of a trained PHOCNet also called attribute scores of all samples in IAM dataset. For example, sub-figure(e) represents the score of non-presence of character 'd' in an image x, $p(a_4 = 0 \mid x)$. For already trained images, the PHOCNet yields severely biased score to zero as shown in grey color histogram. However, estimated scores of OOV words by the PHOCNet, red color histogram, are not that much biased to zero, and are spread out over rather wide range of score. Thus, if pseudo samples are generated from the scores of grey histogram and MLPs is trained by them, then MLPs is hard to classify widely spread scores of OOV images. On the other hand, when the large number of pseudo samples are randomly sampled from normal distribution, the learning of MLPs can cover spread scores of OOV images as shown in yellow color distribution. Sub-figure(f) also represents the scores of fourth attribute but for the presence. Sub-figure(a),(b) and (c),(d) depict that above statement is also valid for first and second attributes respectively. The figure 5.3.4 only includes histograms of representative three attributes, but overall tendency of biasing score of trained samples and spreading score of OOV samples are same for most of 540 attributes.



(a) $p(a_1 = 0 \mid x)$ histograms

(b) $p(a_1 = 1 \mid x)$ histograms

(c) $p(a_2 = 0 \mid x)$ histograms

(d) $p(a_2 = 1 \mid x)$ histograms

(e) $p(a_4 = 0 \mid x)$ histograms

(f) $p(a_4 = 1 \mid x)$ histograms

Figure 5.3.4: The histograms of attribute scores estimated from all samples in IAM dataset by PHOCNet

5.3.5 *Comparison*

The methods proposed in this thesis are evaluated by comparing their experimental results with results of DAM as shown in table 5.3.9. DAP results in lower WER and CER than DAM regardless of the assumptions for class prior, but it gives inferior results in terms of OOV-WER. In this thesis, CCA cannot make a significant improvement

in the error rates because of overfitting. By solving this problem with combining CCA and DAP, CCADAP offers the best results for WER and CER. Table 5.3.9 also shows that MLPs with pseusdo samples yields higher WER and CER than the best results of DAP and CCADAP. However, the MLPs method can make better results than DAM, CCA and DAP without using a bag of words of training set. Even though MLPs method does not give the best results in terms of WER and CER, it offers the best recognition performance for OOV words.

Table 5.3.10 shows segmentation-based word recognition results on the IAM to compare the best result of this work by CCADAP with results of other works. All presented results use the same evaluation protocols such as using closed lexicon for making predictions and using WER and CER as metrics. The reuslts of [AGFV14] which uses Fisher Vecotrs (FV) and linear SVMs are inferior to any other methods in table 5.3.10 that employ ANN. [DKN14] introduces to use Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN), and it offers superior results than [AGFV14]. Afterwards, a method of employing a CNN for handwriting word recognition was proposed by Poznanski et al. in [PW16]. It outperforms the RNN based approach

Method	**Specification**	**Results(%)**		
		WER	**CER**	**OOV-WER**
DAM	Cosine	13.93	7.20	19.41
CCA	$r : 150$ $\lambda : 10^6$	13.61	6.97	18.55
DAP	$p(z)$: uniform $p(a_m)$: unbiased	12.79	6.79	21.53
MLPs	$\#S^{z_z} : 500$	12.17	6.45	**14.66**
DAP	$p(z)$: empirical $p(a_m)$: unbiased	11.89	6.21	22.45
CCADAP	$r : 120$ $\lambda : 10^3$	**10.06**	**5.03**	16.66

Table 5.3.9: Comparison of all proposed methods.

Method	Results(%)	
	WER	**CER**
CCADAP	10.06	5.03
Almazan et al. [AGFV14]	20.01	11.27
Doetsch et al. [DKN14]	12.20	4.70
Poznanski et al. [PW16]	6.45	3.44
Dutta et al. [DKMJ18]	**4.80**	**2.52**

Table 5.3.10: Comparison of the result of CCADAP to results presented by other works on the IAM dataset.

from [DKN14]. CCADAP proposed in this thesis also gives better result than [DKN14] in terms of WER. However, compared to the CNN based approach, 6.45% of WER by [PW16] is lower than 10.06% of WER by CCADAP. Although the quantitative error rates of CCADAP cannot be matched to the numbers of [PW16] in table 5.3.10, the work of this thesis presents competitive results with employing the PHOCNet that has a more compact architecture than the CNN-N-Gram. In other words, the CNN of this thesis uses fewer number of neurons and parameters that have to be trained, especially in the fully connected layers. This means that it spends fewer computational resources and learning time. Furthermore, when both this work and [PW16] employs a CNN to estimate attribute scores, CCADAP is explicitly superior to CCA method of [PW16] in classifying estimated scores to the output classes. It is because CCADAP can reduce WER by about 3.9% from DAM, whereas [PW16] presents improving only about 2.4% of WER from DAM based on CCA. Currently, the state-of-the-art method of handwriting word recognition on the IAM dataset is to employ CNN-RNN hybrid networks. One of such approach, [DKMJ18], presents 4.80% of WER and 2.52% of CER as shown in the best results of table 5.3.10.

6 CONCLUSION

This thesis introduces an approach to recognize segmented handwritten words based on intermediate representation of attribute. This approach formalizes the way of applying attribute-based classification to handwriting recognition, and it is dubbed as attribute-based word recognition.

In this thesis, a PHOC is used as attribute representation, because it is an explicit word discriminative representation based on the presence of character and its spatial position. Then, the proposed recognition system is defined by two cascaded machine learning models. First, the PHOCNet is employed to estimate attribute scores from an image. Second, this thesis introduces four different methods of classifying estimated scores into a class of lexicon and evaluates them by comparing their experimental results on the IAM dataset.

DAM refers nearest neighbour search with distance measurement. Because this simple method is also used in similar work [PW16] with different architecture of CNN called CNN-N-Gram, the recognition results of DAM allow to compare the performance of two different CNN architectures. Even though employing CNN-N-Gram shows better recognition accuracies than employing PHOCNet under same evaluation settings, PHOCNet has more compact architecture with much smaller number of network parameters that have to be trained. However, the work of this thesis demonstrates that the performance of an attribute-based word recognition system is determined by the way of multi-class classification as well as the architecture of CNNs. The experiments of DAP shows that using a probabilistic classifier can offer better recognition results than using simple DAM regardless of prior assumptions. It is because DAP reflects predictions of non-present attributes which are neglected in DAM. This means that DAP fully utilizes estimated attribute score of the input images but DAM is not. Another experiment proves that adapting the term frequency(TF) of training documents as class prior can improve the performance of DAP.

This thesis not only discusses above methods already introduced in other papers but also introduces two novel methods called CCADAP and *MLPs with pseudo samples*. CCADAP is the way of using CCA for a temporal recognition of input handwriting documents by DAP. This method enables to resolve overfitting problem of standard CCA and to learn correlation between estimated scores and predicted binary representation of target documents. As a result, CCADAP shows the best recognition

performance among all proposed method in this thesis. MLPs with pseudo samples proposes to train a conventional softmax MLPs with pseudo samples for zero-data learning. The recognition performance of this method is proportional to the number of pseudo samples per zero-data classes. Even though MLPs does not give the best results in terms of WER and CER, it offers the best recognition performance with respect to OOV words.

Finally, this thesis concludes that the performance of attribute-based word recognition can be improved by analysing attribute representations and by introducing optimal multi-class classifier. It is based on the fact that DAP, CCADAP and MLPs outperform DAM. Furthermore, CCADAP noticeably decreases WER and CER by about 3.9% and 2.2%, respectively, from DAM. Even though this thesis achieves improvement from its own baseline method, the recognition accuracy still cannot match to the state-of-the-art results of [DKMJ18]. Recent works show that sequence model such as CNN-RNN [DKMJ18] outperform holistic method in different word recognition tasks.

Regarding future work, employing architectures based on Residual Network (ResNet) [HZRS16] can be designed and applied to estimate attribute scores. The architecture of the convolutional part of PHOCNet is inspired by VGG. Given that generally ResNets show better results than VGG for image recognition, [Sud18] proposes aforementioned network called PHOCResNet. Experiments of [Sud18] show that this network outperforms PHOCNet in word spotting. Therefore, employing PHOCResNet is highly likely to improve handwriting word recognition. In addition, considering kernel methods for CCA of CCADAP give improvement. The reason is that kernel CCA can reflect non-linear relation between attribute scores and the binary attributes. However, since the given data for the CCA, attribute representations, are composed by large number of samples with high dimensionality, the problem of requiring huge computational cost should be addressed whether kernel methods use Gram matrices or feature maps.

BIBLIOGRAPHY

[ACW14] Aarts, Bas ; Chalker, Sylvia ; Weiner, Edmund: The Oxford Dictionary of English Grammar. In: *OUP Oxford.* (2014), 16 January, S. pp. 436–. ISBN 978–0–19–107900–9

[AGFV14] Almazán, J. ; Gordo, A. ; Fornés, A. ; Valveny, E.: Word Spotting and Recognition with Embedded Attributes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), Dec, Nr. 12, S. 2552–2566. – ISSN 0162–8828

[Aka01] Akaho, S.: A Kernel Method For Canonical Correlation Analysis. In: *In Proceedings of the International Meeting of the Psychometric Society (IMPS2001*, Springer-Verlag, 2001

[BJ02] Bach, FR ; Jordan, MI: Kernel independent component analysis. In: *Journal of machine learning re- search 3* (Jul 2002), S. 1–48

[BS12] Bochkarev, Shevlyakova A. V. V. V. V. V. V. ; Solovyev, V. D.: Average word length dynamics as indicator of cultural changes. In: *in society. CoRR, abs/1208.6109.* (2012)

[DKMJ18] Dutta, Kartik ; Krishnan, Praveen ; Mathew, Minesh ; Jawahar, C.V.: Improving CNN-RNN Hybrid Networks for Handwriting Recognition, 2018, S. 80–85

[DKN14] Doetsch, Patrick ; Kozielski, Michal ; Ney, Hermann: Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition. In: *2014 14th International Conference on Frontiers in Handwriting Recognition* (2014), S. 279–284

[DR88] DE Rumelhart, RJ W. GE Hinton H. GE Hinton: Learning representations by back-propagating errors. In: *Neurocomputing: foundations of research Pages 696-699*. MIT Press Cambridge, MA, USA, 1988

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016

[GZ95] Golub, GH ; Zha, H: The canonical correlations of matrix pairs and their numerical computation. In: *In Linear algebra for signal processing. Springer* (1995), S. 27–49

[HK70] Hoerl, A. E. ; Kennard, R. W.: Ridge Regression: Biased Estimation for Nonorthogonal Problems. In: *Technometrics* 12 (1970), S. 55–67

[Hot36] Hotelling, Harold: Relations Between Two Sets of Variates. In: *Biometrika* 28 (1936), Nr. 3/4, S. 321–377. – ISSN 00063444

[HZRS16] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), S. 770–778

[JRFR13] J Rousu, O Sodeinde J Shawe-Taylor DD A. DD Agranoff ; Fernandez-Reyes, D: Biomarker discovery by sparse canonical correlation analysis of complex clinical phenotypes of tuberculosis and malaria. In: *PLoS Comput Biol 9, 4* (2013), S. e1003018

[JSVZ14] Jaderberg, M. ; Simonyan, K. ; Vedaldi, A. ; Zisserman, A.: Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition. In: *ArXiv e-prints* (2014), Juni

[Knu92] Knuth, Donald E.: Two Notes on Notation. In: *Am. Math. Monthly* 99 (1992), Mai, Nr. 5, 403–422. http://dx.doi.org/10.2307/2325085. – DOI 10.2307/2325085. – ISSN 0002–9890

[KSH] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, S. 2012

[Lar31] Larson, SC: The shrinkage of the coefficient of multiple correlation. In: *Journal of Educational Psychology 22, 1* (1931)

[LBBH98a] Lecun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*, 1998, S. 2278–2324

[LBBH98b] Lecun, Yann ; Bottou, Léon ; Bengio, Yoshua ; Haffner, Patrick: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*, 1998, S. 2278–2324

[LBD+90] LeCun, Yann ; Boser, Bernhard E. ; Denker, John S. ; Henderson, Donnie ; Howard, R. E. ; Hubbard, Wayne E. ; Jackel, Lawrence D.: Handwritten Digit Recognition with a Back-Propagation Network. Version: 1990. http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf. In: Touretzky, D. S. (Hrsg.): *Advances in Neural Information Processing Systems 2*. Morgan-Kaufmann, 1990, 396–404

[LCJ15] L Cao, J Li R J. Z Ju J. Z Ju ; Jiang, C: Sequence detection analysis based on canonical correlation for steady-state visual evoked potential brain computer interfaces. In: *Journal of neuroscience methods 253* (2015), S. 10–17

[LEB08] Larochelle, Hugo ; Erhan, Dumitru ; Bengio, Yoshua: Zero-data Learning of New Tasks. In: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI Press, 2008 (AAAI'08). – ISBN 978–1–57735–368–3, S. 646–651

[LNH14] Lampert, C. H. ; Nickisch, H. ; Harmeling, S.: Attribute-Based Classification for Zero-Shot Visual Object Categorization. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), March, Nr. 3, S. 453–465. – ISSN 0162–8828

[MB02] Marti, Urs-Viktor ; Bunke, Horst: The IAM-database: an English sentence database for offline handwriting recognition. In: *International Journal on Document Analysis and Recognition* 5 (2002), S. 39–46

[Nie15] Nielsen, Michael A.: "Neural Networks and Deep Learning", Determination Press, 2015

[PW16] Poznanski, A. ; Wolf, L.: CNN-N-Gram for HandwritingWord Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 2305–2314

[PZG+16] Pratikakis, Ioannis ; Zagoris, Konstantinos ; Gatos, Basilis ; Puigcerver, Joan ; Toselli, Alejandro H. ; Vidal, Enrique: ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016). In: *15th International Conference on Frontiers in Handwriting Recognition, ICFHR 2016, Shenzhen, China, October 23-26, 2016*, 2016, 613–618

[Roj96] Rojas, Raul: Neural networks: a systematic introduction. Springer Science & Business Media, 1996

[Ros58] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. In: *Psychological Review* (1958), S. 65–386

[Saa11] Saad, Y: Numerical methods for large eigenvalue problems. In: *Vol. 158. SIAM* (2011)

[SF16] Sudholt, Sebastian ; Fink, Gernot A.: PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016

[SF17] Sudholt, Sebastian ; Fink, Gernot A.: Evaluating Word String Embeddings and Loss Functions for CNN-based Word Spotting. In: *Proc. Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017

[SF18] Sudholt, Sebastian ; Fink, Gernot A.: Attribute CNNs for Word Spotting in Handwritten Documents. In: *Int. Journal on Document Analysis and Recognition* 21 (2018), Nr. 3, S. 159–160

[Sud18] Sudholt, Sebastian: *Learning attribute representations with deep convolutional neural networks for word spotting*, Technical University of Dortmund, Germany, Diss., 2018

[SWZ08] S Waaijenborg, PC Verselewel de Witt H. ; Zwinderman, AH: Quantifying the association between gene expressions and DNA-markers by penalized canonical correlation analysis. In: *Statistical Applications in Genetics and Molecular Biology 7, 1* (2008)

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *CoRR* abs/1409.1556 (2014). http://arxiv.org/abs/1409.1556

[UMK+17] Uurtio, V. ; Monteiro, J. M. ; Kandola, J. ; Shawe-Taylor, J. ; Fernandez-Reyes, D. ; Rousu, J.: A Tutorial on Canonical Correlation Methods. In: *ArXiv e-prints* (2017), November

[Vin76] Vinod, HD: Canonical ridge and econometrics of joint production. In: *Journal of Econometrics 4, 2* (1976), S. 147–166

[Wau42] Waugh, FV: Regressions between sets of variables. Econometrica. In: *Journal of the Econometric Society* (1942), S. 290–310

[XBSY13] X-B Shen, Q-S S. ; Yuan, Y-H: Orthogonal canonical correlation analysis and its application in feature fusion. In Information Fusion (FUSION). In: *16th International Conference on. IEEE* (2013), S. 151–157