technische universität
dortmund

TRANSFER LEARNING FOR WORD SPOTTING IN HANDWRITTEN DOCUMENTS

Master Thesis

Neha Gurjar
May 26, 2017

Supervisors:
M. Sc. Sebastian Sudholt
Prof. Dr.-Ing. Gernot A. Fink

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

CONTENTS

1 INTRODUCTION

Document analysis is a broad field which includes the study of efficient indexing and digitizing of handwritten documents. Image libraries contain several handwritten documents of cultural and historical significance. Owing to the handwritten nature of the documents, the information contained in them cannot be easily accessed. Hence, unlocking this information through digitization and making it easier to browse or search through the documents is the primary motivation for the study of document analysis. One of the approaches used for digitization and indexing such documents which has achieved great success is called word spotting. Word spotting is defined as an approach which annotates words in handwritten documents, typically to generate an index, by matching them to other words in the collection and clustering them. Word spotting is more efficient as compared to the traditional handwriting recognition approaches, since it does not require an exhaustive processing and recognition of all the document contents.

Historic documents are of particular interest for document analysis. However, they are also quite challenging to analyze because the images obtained from them are often of poor quality. For example, as stated in [RM07], the George Washington collection is not scanned from the originals, but from microfilms, further decreasing the quality of the images. Several supervised and unsupervised approaches have been developed for word spotting which achieve good results for such datasets. Supervised approaches require training data which is labelled, as opposed to unsupervised approaches where no annotation is required. Some unsupervised techniques in word spotting such as [RVF12, RATL15] make use of local features extracted from word images which are used in Bag-of-Features representations. On the other hand supervised approaches use annotated training data to train a model for word spotting. Such a supervised approach which makes use of Support Vector Machines is introduced in [AGFV14]. Another supervised approach which learns a Convolutional Neural Network(CNN)-based model is described in [SF16]. Supervised approaches outperform unsupervised approaches in word spotting. They are designed to be robust to a large variation within a dataset such as that which occurs in handwritten documents. However, this means that the dataset used for learning the model must be large enough to span this range of variation. Annotating a large amount of training data is a tedious task which requires a large amount of manual effort and time. Hence, it is desirable to

explore techniques which allow a reduction in the amount of handwritten training data required while retaining the high performance shown by a supervised approach. This is the motivation behind the methodology presented in this work.

A CNN-based model is learned for word spotting in this approach. In order to reduce the amount of handwritten data, this work uses the approach of transfer learning. In transfer learning, knowledge from one domain is used to improve a task in another domain. In the method presented here, the CNN is first trained on synthetically generated data which emulates handwriting and the variation in handwritten documents. Since this data is generated by rendering word strings using various fonts in a specific manner, no extensive manual effort is required to obtain annotations for this dataset. The network performance is then evaluated on handwritten data. Hence, in this approach, the *knowledge* obtained by the network using the synthetic data is used to accomplish the task of word spotting in handwritten documents. It is expected that the network faces limitations because of the difference in the visual appearance of the synthetic and handwritten datasets. In this case, it is finetuned with a small amount of handwritten data in order to boost its performance. Thus, with this approach, the CNN is trained using weak supervision, i.e., a fraction of the training data normally used for supervised approaches. As a result, the amount of manual effort required for annotating handwritten training data and the time required for training the network are also reduced. Thus, transfer learning and weak supervision are used to overcome the challenges which occur in word spotting using a CNN.

The next chapters describe the tools and concepts which are required for this approach of transfer learning for word spotting in handwritten documents and discuss the experiments ad results supporting this approach. Chapter 2 explains the fundamental concepts in machine learning which have been used in this method. It gives a detailed explanation of the origin and development of connectionist systems as well as that of transfer learning, whereas chapter 3 defines word spotting and the terminology, and describes various approaches in word spotting. Chapter 4 explains the approaches which have most prominently inspired this thesis. The proposed approach uses a Convolutional Neural Network called the PHOCNet introduced by [SF16]. Hence, the approaches showing merits of Convolutional Neural Networks in the field of word spotting are discussed in this chapter. Additionally, the approaches [AF15, KJ16] discussed in the chapter are credited as the primary sources of inspiration for the idea of implementing transfer learning using synthetic training data. Chapter 5 explains the method proposed in this work, whereas chapter 6 provides the details of the experimental set-up. This chapter further discusses the results of the experiments. Finally, chapter 7 provides a summary of the thesis and the conclusions drawn.

2

FUNDAMENTALS

Word spotting in handwritten documents uses several supervised and unsupervised learning techniques. In this work, the supervised approach using a Convolutional Neural Network is explored. Transfer learning is implemented in order to reduce the amount of human effort needed to prepare training data. This chapter aims at familiarizing the reader with the fundamental tools, concepts and terminology associated with machine learning, and transfer learning for word spotting using a Convolutional Neural Network.

In this chapter, section 2.1 gives a brief overview about the concepts in Machine Learning which have been used in this work. Section 2.2 explains the evolution of the concept of Artificial Neural Networks, whereas section 2.3 introduces Convolutional Neural Networks and their salient features. Finally, section 2.4 provides an overview of the scope of transfer learning and its types.

2.1 BASICS OF MACHINE LEARNING

Arthur Samuel (1959) coined the term *machine learning* and defined it as the "field of study that gives computers the ability to learn without being explicitly programmed" [Mun14]. In simple terms, machine learning is the study of of systems that improve their behaviour by learning through experience [DR08]. In several cases, the experience is a collection of observations or examples within a domain relevant to the task at hand. One aims to observe a pattern and define classification rules over this data in order to provide insights into that domain. The learning process typically involves refining of the classification rules with the help of the aforementioned experience. Some of the tasks under machine learning include clustering of data, classification, and regression over data. Two broad categorizations of learning techniques are supervised and unsupervised Learning. Supervised Learning techniques require labeled training examples, i.e. the data samples used for training need to be manually assigned the correct label prior to training. On the other hand, unsupervised learning techniques do not require an assignment of labels for the data. The key difference between supervised and unsupervised learning defined by [GBC16] is that unsupervised learning involves observing several examples of a random vector, and attempting to learn the probability

distribution, or some properties of that distribution of that random vector, whereas supervised learning is concerned with observing several examples of a random vector and an associated label, and learning to predict the label for a given example, usually by estimating the probability of the label given the data sample.

A branch of machine learning which focuses on the recognition of regularities and patterns in given data is pattern recognition. Specifically, pattern recognition aims at categorizing input data into classes through the extraction of significant attributes of the input. Two broad categories of tasks in pattern recognition are recognition and retrieval. In the recognition task, a system is developed in order to directly recognize a pattern, i.e. assign a label to it. However, in the retrieval task, a pattern sample is used as a query for the system. The relevant patterns, i.e. patterns which are similar to the query with respect to the significant attributes, from a given collection of patterns are retrieved. This helps to cluster and label similar patterns without explicitly recognizing and labeling each of them individually. Tools and techniques defined in machine learning are commonly used to accomplish tasks in pattern recognition. A neural network is one of the supervised machine learning techniques capable of learning a model from the given data and classifying the data samples into an arbitrary number of classes.

Consider a task of classifying input data samples from a certain domain D (e.g. animal image domain, text image domain etc.) into n classes and assigning each of them a label y. For this purpose, it requires labeled training data, i.e. data samples from the same domain D with suitable corresponding labels, known as the target output. For a classification task, this may be in the form of a *1-hot* vector in which each label has all zero elements except for a single element of value 1. The index of the element with value 1 indicates the class label which corresponds to the given data sample. Other examples of target output are embedding vectors or descriptor vectors pre-defined for the class to which the data sample belongs.

Given the labeled data, a model must be learned for classification of the data. Classification with a neural network is an optimization task, i.e., it involves minimization or maximization of some function by altering the variables on which the function depends. The question that now arises is: considering that the variables in this case are the parameters of the neural network, what function can one define over these parameters the minimization of which achieves a good classification over given data? In this example, the input is first fed to the network and the output is calculated at the last layer. This output depends on the input as well as the parameters of the network. This step is known as the *forward pass*. The network output and the target output are compared using a loss function which is a computationally feasible function representing the penalty for inaccuracy of predictions. Mathematically, the loss function maps

the values of the target output and network output on to a real value representing the cost or penalty of the event. Few of the typically used loss functions for the network output y and target output $\hat{y}$ are listed below. Here, y_i and $\hat{y}_i$ are the i^{th} elements of the network output and target output respectively.

- Total Sum Squared Error (TSSE)

$$TSSE = \sum_{i=1}^{n} \|y_i - \hat{y}_i\|^2 \tag{2.1.1}$$

  where n is the number of training examples, and $\|.\|$ is the Euclidean distance between the vectors y and $\hat{y}$.

- Total Mean Squared Error (TMSE)

$$TMSE = \frac{1}{n \cdot l} \sum_{i=1}^{n} \|y_i - \hat{y}_i\|^2 \tag{2.1.2}$$

  where n is the number of training examples, l is the number of output neurons and $\|.\|$ is the Euclidean distance between the vectors y and $\hat{y}$.

- Cross Entropy Loss (CEL)

$$CEL = -\frac{1}{l} \sum_{i=1}^{l} [\hat{y}_i \log y_i + (1 - \hat{y}_i) \log(1 - y_i)] \tag{2.1.3}$$

  where l is the number of output neurons.

Depending on factors such as the nature of the output, one function may be chosen. For example, the cross entropy loss is typically chosen for multi-class labeling tasks.

An error value is calculated for each output neuron. It is propagated backwards from the output layer to the input layer till each neuron has an error value associated with it which approximately represents its contribution to the output error. The parameters of the network are adjusted in order to minimize this error. This step is known as the *backward pass*. One forward pass and one backward pass are together considered to be one iteration. The parameters are updated at each iteration for several such iterations using all the training data samples to acquire a trained model which maps the data from domain D on to the correct corresponding label.

The primary tool used to accomplish the task of word spotting in this thesis is a Convolutional Neural Network. A Convolutional Neural Network has a different

structure and types of layers as compared to a classic neural network described above. The input given to a neural network is required to be in the form of a vector. Each element of the input represents one input neuron. For example, pixel intensities of an image may be restructured into a 1-dimensional vector before being fed to the network. However, in a Convolutional Neural Network, the input is visualized in the form of a 2-dimensional structure. The merits, specific terminology and details of the layers of a Convolutional Neural Network will be discussed in the section 2.3.

2.2 ARTIFICIAL NEURAL NETWORKS

The previous section provided an overview of the working of a neural network. This section discusses the details of the structure of a neural network, the parameters as well as the technique used for training a neural network. To understand the concept of neural networks, it is important to know how it evolved from the basic idea of a neuron.

The basic unit of an artificial neural network, the artificial neuron was introduced by Warren McCulloch and Walter Pitts in 1943 [MP43]. It was modeled after the biological neuron. The McCulloch-Pitts neuron is shown in Fig.2.2.1 [Nie15]. This simple model considered only binary inputs and outputs. The neuron is activated (value equals 1) if the sum of the binary inputs exceeds a pre-defined threshold. Mathematically, it is expressed as follows:

$$f(x_1, ..., x_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} x_i \geqslant \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.2.1}$$

where the threshold $\theta > 0$. The function f of the inputs is called the activation function. The value of the function for any arbitrary inputs is called the output of the neuron. Since the model parameters are not learned, this model is static in nature. Frank Rosenblatt (1958) [Ros58] developed the perceptron, which was inspired by the McCulloch-Pitts Neuron. The following sections discuss the concept and evolution of the perceptron into neural networks.

2.2.1 *Perceptron*

The perceptron introduced weights to the model corresponding to each of the inputs. The weights are real numbers demonstrating the impact each input has in order to determine the output. The neuron is activated when the weighted sum of inputs

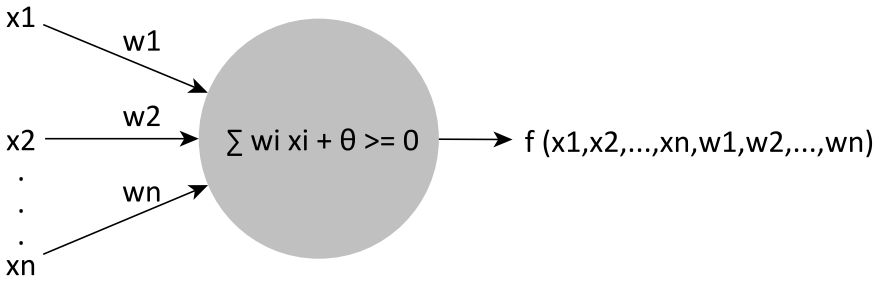


Figure 2.1.1: Perceptron as developed by Frank Rosenblatt takes weighted inputs. The condition for which the output of the neuron is 1 is shown within the neuron in the image.

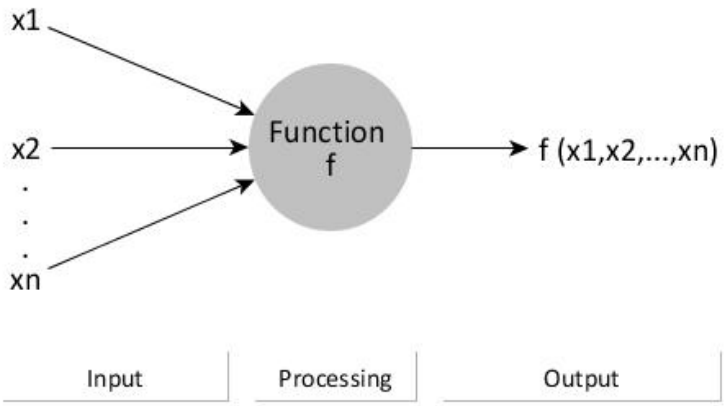


Figure 2.2.1: Model of the McCulloch-Pitts Neuron. The neuron may have an arbitrary number of binary inputs. The activation function f for the neuron has been specified in equation 2.2.1

exceeds the threshold value, which is also a real number. The perceptron is visualized in Fig.2.1.1 The mathematical model for such a perceptron can be given by:

$$f(x_1, ..., x_n, w_1, ...w_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i + \theta >= 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.2.2}$$

where θ is the threshold and w_i are the weights. This is visualized in Fig.2.1.1.

The higher the weight associated with an input, the greater is the influence which that input has over the output. This perceptron is used to model the logic gates AND, OR, NOR, NAND and NOT.

A further development was made to this model by Marvin Minsky and Seymor Papert (1969) [MP69]. The perceptron by Rosenblatt considered only binary values as inputs. However, the Minsky-Papert perceptron (MPP) accepted real values within the interval [0,1] as input. Geometrically, this perceptron was capable of dividing an n-dimensional space into two parts or classes by modeling a separating function. This allowed the MPP to model linear equations in addition to Boolean functions and the logic gates. Hence, the MPP is found to be atleast as powerful as the McCulloch-Pitts neuron.

2.2.2 *Multi-Layer Perceptron*

Although the perceptron can be used to model simple logic and linear functions, it has limitations in the face of more complex functions. Some of the limitations include modeling of the exclusive-OR (XOR) logic gate and determination of parity. These limitations can be overcome by using non-linear separating functions or by using multiple layers of perceptrons. For example, an XOR logic gate can be modeled with a multi-layer perceptron (MLP) as shown in Fig.2.2.2.

The parameters of a simple perceptron were traditionally tuned manually. However, learning algorithms can be developed to tune the parameters automatically. This tuning takes place in response to external stimuli, without direct manual intervention [Nie15]. Instead of explicitly calculating the parameters or laying out a circuit of logic gates, networks with multiple layers of perceptrons can simply learn to solve problems of arbitrary complexity.

Multi-layer perceptron networks are commonly known as artificial neural networks. Each of the layers may contain one or more neurons all connected to each neuron of the preceding and succeeding layers with certain weights. The inputs of a network are typically shown to be neurons with no input. However, they can be considered to

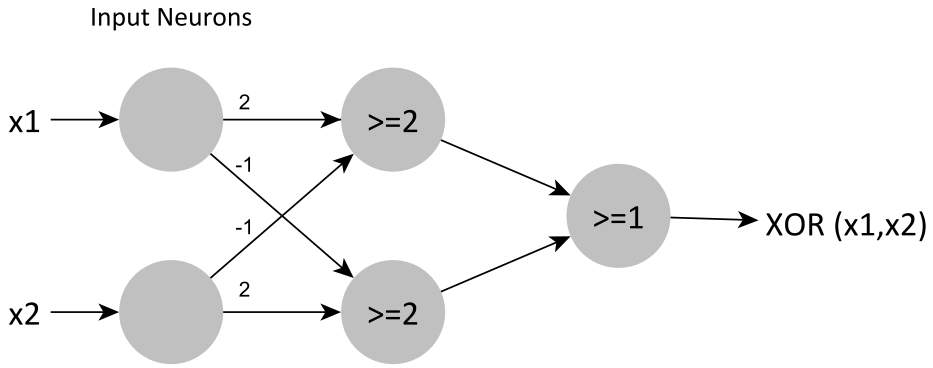


Figure 2.2.2: XOR logic gate over arbitrary binary inputs x1 and x2 modeled by a multi-layer perceptron. The input neurons shown in the image are special units used to define input values [Nie15].

be special units used to define the input values [Nie15]. The layers between the input and output are known as hidden layers. A neural network with one hidden layer is shown in Fig.2.2.3.

2.2.3 *Learning with Gradient Descent*

Multi-layer perceptrons are capable of learning models of arbitrary complexity. Their parameters can be trained using a training algorithm. One of the underlying principles of network training algorithms most commonly used is back propagation of error with gradient descent. As explained in section 2.1, the error between the target output and the network output is calculated and the parameters are corrected in order to minimize the error. The parameters of the output layer are updated first and subsequently the previous layers till the first layer is reached. Hence, this is called back-propagation of error and was first introduced by [WH86].

In order to understand the method of training of multi-layer perceptrons, the nature of a single perceptron must be explained. The activation condition of the Rosenblatt perceptron from equation 2.2.2 can be written as follows.

$$\sum_{i=1}^{n} w_i x_i + b > 0 \tag{2.2.3}$$

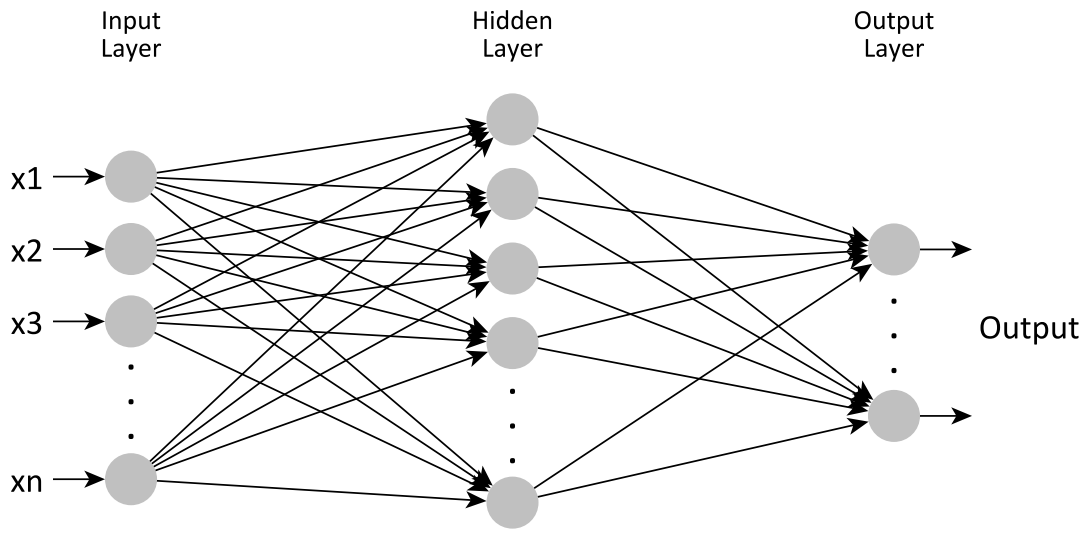


Figure 2.2.3: An Artificial Neural Network such as the one shown in this figure has an arbitrary number of input, hidden, and output neurons. Each neuron of each layer is connected to each neuron of the preceding and succeeding layers. It may also have multiple hidden layers.

where b is called the bias. The parameters, i.e. the weights and biases must be adjusted gradually such that the error approaches its minima. The error is, thus, defined as a smooth function of the parameters. This acts as the cost function to be minimized.

$$C(w, b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2 \tag{2.2.4}$$

where C is the cost function of weights w and biases b, n is the number of training inputs, $y(x)$ is the desired or target output vector, a is the output vector of the network with input x. The network output a is also dependent on w, b and the activation function of the neuron h such that $a = h(wx + b)$. $\|.\|$ denotes the L2 norm of a vector. C is called the quadratic cost function or the mean squared error (MSE) [Nie15]. The cost function $C(w, b)$ is non-negative, since every term in the sum is a square. Moreover, the cost function $C(w, b)$ is approximately 0, when $y(x)$ is approximately equal to the output, a, for all training inputs x. Thus, the goal of the training algorithm is defined to find weights and biases for which $C(w, b) \approx 0$. This is achieved using an algorithm known as gradient descent using back-propagation of error.

In order to find the minimum of the cost function 2.2.4, the gradient vector of ∇C is determined which indicates the change in the value of the function. The network parameters are updated such that the function value decreases, i.e. the gradient is followed along the negative direction. Since C is a function of the weights w and biases b, the gradient vector has the components $\partial C/\partial w$ and $\partial C/\partial b$. Hence for an arbitrary weight and bias, the gradient descent update rule can be defined as:

$$w_i \rightarrow w_i' = w_i - \eta \frac{\partial C}{\partial w_i} \tag{2.2.5}$$

$$b_i \rightarrow b_i' = b_i - \eta \frac{\partial C}{\partial b_i} \tag{2.2.6}$$

where η is the learning rate. It is a small positive scalar which is typically used to limit the rate at which the parameters are updated. A large change in the parameters may cause the function value to change drastically which may result in a divergence or an oscillation of the error between two values. By repeatedly updating the weights and biases with this rule, the direction of the negative gradient of the function is followed in order to seek the minimum. However, in order to obtain the partial derivative terms $\frac{\partial C}{\partial w_i}$ and $\frac{\partial C}{\partial b_i}$, the terms $\frac{\partial a}{\partial w_i}$ and $\frac{\partial a}{\partial b_i}$ must be calculated. Since $a = h(x)$, where h is the activation function of the layer, the partial derivative terms can be further

expressed as $\frac{\partial h(x)}{\partial w_i}$ and $\frac{\partial h(x)}{\partial b_i}$. However, the classically defined activation function for an arbitrary input u is given by:

$$h(u) = \begin{cases} 1, & \text{if } u > \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.2.7}$$

This function, also called the signum function, is not continuous in nature. This is evident from the graphical representation of the function in Fig.2.2.4. Hence, it can be differentiated across the domain except at 0, its point of discontinuity. Although the derivative for the function can be approximated to 0 across the domain, it cannot be used for back-propagation. Hence, a class of functions known as sigmoid functions is used as a substitute for the signum function. The sigmoid functions are approximations of the signum function. They are continuous in nature and, hence, differentiable. The following are examples of sigmoid functions:

- **Logistic Function**

  The following equation shows the logistic function for an arbitrary input u.

  $$\text{output} = \frac{1}{1 + \exp(-u)} \tag{2.2.8}$$

  It can be visualized in Fig.2.2.5

- **Hyperbolic Tangent**

  The Hyperbolic tangent function is shown in Fig.2.2.5. The asymptotes of the function at $y = -1$ and $y = 1$ limit the values at the Y-axis between that interval.

  $$\text{output} = \tanh(u) \tag{2.2.9}$$

Neurons with sigmoid activation functions are referred to as sigmoid neurons. Due to the almost linear nature of their output for input values close to zero, a small change in the weights and biases results only in a small change in output as opposed to the flip between 0 and 1 caused when the input crosses 0 using a signum function. Moreover, input values farther from 0 result in an output bound between the limits of the function. Therefore, the output of a sigmoid neuron in terms of the inputs, weights and biases with, in this case, a logistic activation function can be given by:

$$a_{j+1} = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)} \tag{2.2.10}$$

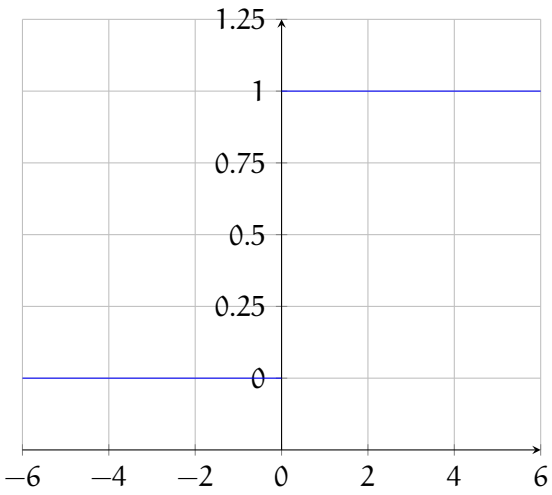
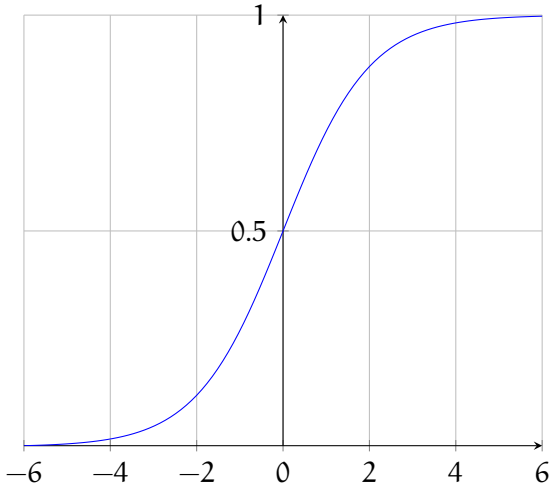


Figure 2.2.4: The figure shows that the Signum Function is not continuous in nature. The discontinuity exists at 0 where the value of the function is both 0 and 1.

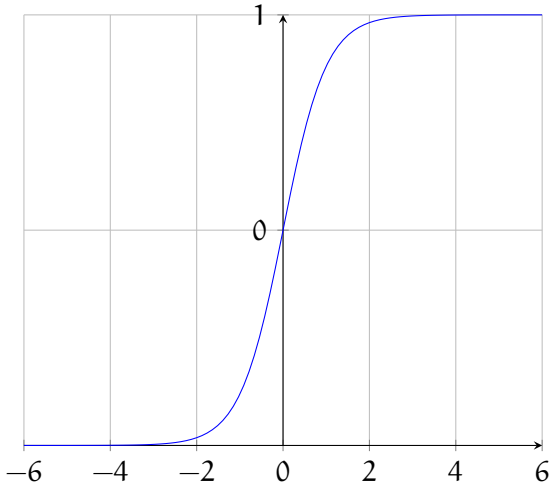
The parameters of the network are updated first at the output layer, next at the layer preceding the output layer and so on till the input layer. Thus, the network is trained till the error is sufficiently low or till the training completes a certain number of iterations. One iteration consists of passing one input sample through the network, computing the gradient and updating the parameters.

As shown in 2.2.4, the cost function is an average of cost functions computed for each individual training input. However, in practice the gradient is calculated for each individual training input and is averaged over the number of training samples. If the number of training samples is large, the computation of gradient for each training example at a time can slow down the learning process. In this case, Stochastic Gradient Descent is used to lower the number of parameter updates during the learning process. The training data is divided into random mini-batches. All training samples are fed to the network to generate a cumulative error. The gradient is then computed for this entire mini-batch instead of a single input sample and averaged over the number of samples. Provided the batch size is large enough, it can be expected that the average value of the gradient calculated for a batch will be roughly equal to the average over the entire training set [Nie15].

$$\frac{\sum_{j=1}^{m} \nabla C_{X_j}}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C, \tag{2.2.11}$$



(a) Logistic Function



(b) Hyperbolic Tangent Function

Figure 2.2.5: The Logistic function and the Hyperbolic Tangent function are continuous in nature. The Logistic function is bounded on the Y-axis between 1 and 0, whereas the Hyperbolic tangent function is bounded between 1 and -1.

where m is the number of training samples in the mini-batch. Thus, Stochastic Gradient Descent increases the speed of learning. The update rule for the network parameters according to Stochastic Gradient Descent can be given by:

$$w_i \rightarrow w_i' = w_i - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_i} \tag{2.2.12}$$

$$b_i \rightarrow b_i' = b_i - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_i}, \tag{2.2.13}$$

2.3 CONVOLUTIONAL NEURAL NETWORKS

So far, the working of Artificial Neural Networks with fully connected layers has been discussed. That is, all neurons of a layer are connected to all other neurons in the preceding and succeeding layers. However, this structure treats input data points that are spatially far apart the same as if they were close together and vice versa. This means that the network does not take into account the spatial structure of the input data. In order to include this information in the model to be learned, the structure called Convolutional Neural Network (CNN) can be used.

CNNs typically consist of several types of layers. The architecture of a particular network can be determined by defining a combination of these layers which is best suited to the task for which the model is to be trained. This sections introduces the types of layers in a CNN.

2.3.1 *Convolution*

Classically, convolution is defined as a mathematical operation over two continuous functions which gives the integral of the product of the functions when one is reversed and translated as shown in equation 2.3.1.

$$f * g(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{2.3.1}$$

where f and g are arbitrary continuous functions in domain t.

Convolution can also be defined for discrete spaces. In this case, the function may have any value at discrete points of the independent variable. An example of such a discrete function in space is an image. Each of the pixel intensities occur at a discretely defined location, known as a pixel, in a 2-dimensional space. Moreover, this is an

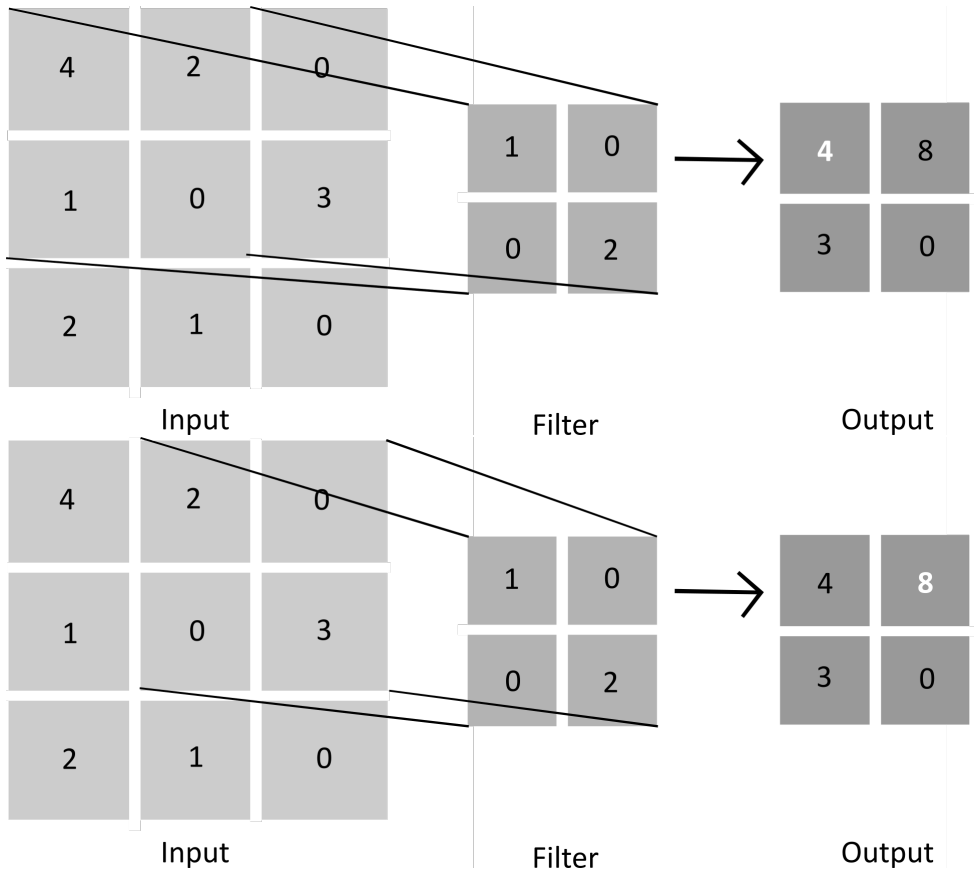


Figure 2.3.1: This figure gives an example of convolution in a 2-dimensional discrete space. The top figure shows the first step of convolution with a 2×2 filter. The figure on the bottom shows the second step with stride 1.

example of a finite signal since each image has a finite number of pixels. Convolution for such a function with another 2-dimensional kernel is shown in Fig.2.3.1. Consider the input to be pixel intensities of a 3×3 gray image. The filter parameters are multiplied element-wise with the pixel values of a part of the input which has the same dimensions as that of the filter. The sum of these products represents the corresponding output element. The filter is then slid over the input by 1 pixel and the process is repeated to give the next output element. The number of pixels by which the filter slides at each step is known as the stride. In the example shown in Fig.2.3.1, the filter slides over the input with stride 1. This process is repeated for the entire 2-dimensional input. This is known as discrete convolution. It is used in the convolutional layers of a CNN as discussed further in this section.

2.3.2 *Convolutional Layer*

Consider the input layer to be a 2 dimensional matrix of neurons. A convolutional layer of a CNN consists of multiple filters which, when convolved with the input during the forward pass, extract one feature each from any region of the input. The result of this convolution is used to activate the neurons in that layer. The activations of these layers are known as feature maps. In contrast to the vector output of a neural network, the feature maps from the convolutional layers are 2-dimensional. The values of the filters within the convolutional layer are called the weights of the network. Similar to neural networks, the layers between the output and input layers are called hidden layers.

In contrast to a fully connected network, the filters in a convolutional layer connect to the input at only a small local neighborhood at a time. In the context of CNNs, it is known as the local receptive field. Each local receptive field at the input is associated with one neuron in the next layer. This field is slid over the entire input in steps in order to extract features. Typically, the field moves one pixel at a time, i.e., a stride of 1 as shown in Fig.2.3.2. But, the stride length can be adjusted if necessary.

Each of the hidden neurons has weights and a bias associated with it which maps the local receptive field onto it. The same weights are used to map all local receptive fields onto respective neurons in that hidden layer. This means that the same filter values determine the mapping of an input layer onto the particular hidden layer. In other words, all the weights and biases in a convolutional layer are shared. An advantage of sharing parameters is that the number of parameters necessary is greatly reduced as compared to a fully connected network [Nie15]. This limits the degrees of

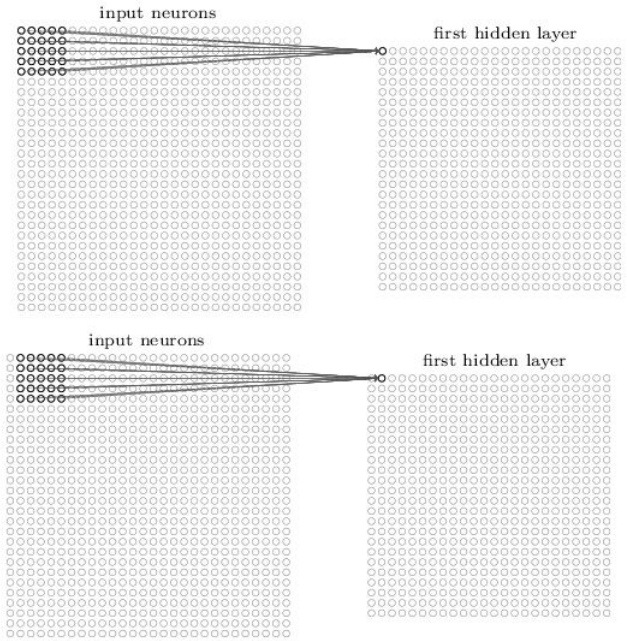


Figure 2.3.2: Local receptive field sliding over an input with stride = 1 [Nie15]. In the top figure, a 5×5 local receptive field is shown to be associated with the first hidden neuron in the first row whereas. The bottom figure shows that when the local receptive field is slid over the input by 1 pixel, the new 5×5 neighbourhood is associated with the second hidden neuron in the first row.

freedom of the model and, in turn, avoids overfitting. For neuron (j, k) in a particular hidden layer, the output is given by

$$\text{output} = \sigma\left(b + \sum_{l=0}^{n}\sum_{m=0}^{n} w_{l,m}\, a_{j+l,k+m}\right). \tag{2.3.2}$$

where σ is the activation function of the neurons, w, b are weights and biases respectively, n is the length of the local receptive field, and a is the input. This implies that all the neurons in a hidden layer detect the same feature at different locations of the input [Nie15]. This makes CNNs robust to translational shift in input. That is, it can detect a feature in the input regardless in which region of the input the feature exists. Moreover, a layer of a network consists of several feature maps resulting from convolutions with several filters in order to detect more features, especially for a recognition task.

2.3.3 *Pooling Layer*

The pooling layer does not have any trainable parameters. It is visualized as a matrix of neurons similar to a convolutional layer. Each neuron is activated by a suitable value from a small local neighborhood of the previous layer. The value which activates the neuron is determined by the pooling function. The local receptive field of a pre-defined size slides over the input feature map with a pre-defined stride. Note that here, the input feature map refers to the entire set of output activations of the previous layer.

There are several pooling functions which are used in practice. The neurons in a max-pooling layer output the maximum value within the associated local receptive field. An example of max-pooling is shown in Fig.2.3.3. Similarly, for an average-pooling layer, the neurons output the average of the receptive field. L2 pooling layer is also a commonly used layer [Nie15] which takes the square root of the sum of the squares of activations in the receptive field of the input feature map.

The feature map contains the exact locations of the features extracted from the input. However, to classify an input or generate a descriptor, it is enough to understand the location of the extracted features relative to each other. The pooling layer discards the exact spatial information about the features, but retains the relative positions of the features. Thus, pooling condenses the information in the feature maps and down-samples them.

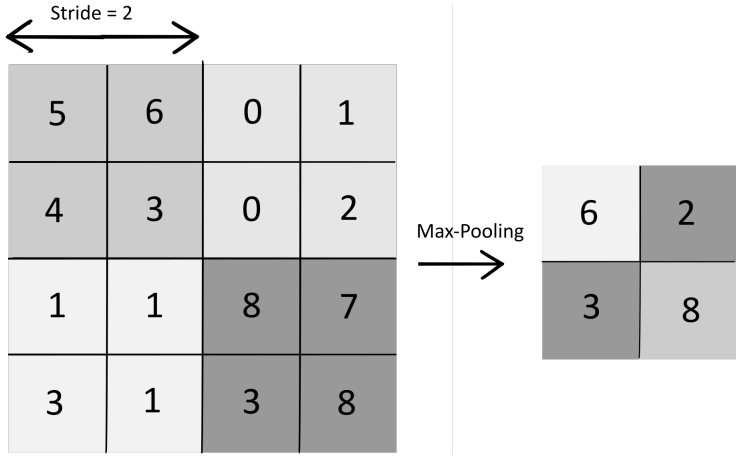


Figure 2.3.3: Max-pooling advancing with stride 2. The maximum value from a 2×2 local neighbourhood is taken as the output of a max-pooling layer.

2.3.4 *Rectified Linear Unit*

The Rectified Linear Unit (ReLU) shown in Fig.2.3.4 is an activation function used to introduce a non-linearity. Mathematically, ReLU function for input u is given by:

$$\text{output} = \max(0, u). \tag{2.3.3}$$

ReLU is a substitute for the sigmoid neuron. The output of a traditionally used sigmoid neuron saturates to 1 when the weighted sum of the inputs to the neuron is high. The gradients calculated for sigmoid neurons are typically small fractions especially for such inputs. Moreover, the gradients of a layer in the network are calculated via the chain rule as products of the gradients of the succeeding layers. Hence, as a result of the multiplication of the fractional gradient values of the outer layers, the gradients of the layers closer to the input layer are much smaller as compared to those of the outer layers. This is known as the problem of vanishing gradients. It results in the inner layers of the network learning much slower as compared to the outer layers. Since the sigmoid neurons are susceptible to the problem of vanishing gradients [PMB13], a network built with sigmoid neurons can be of limited complexity since a deeper structure results in extremely slow learning of the inner layers. ReLU can be used to overcome this limitation, since it can never be saturated. It can produce an output which is not bounded by an upper limit when the weighted input exceeds 0. Since the change in inputs is not inhibited at the output of ReLU, neither are the gradient values.

Hence, the sigmoid activation function is replaced by ReLU for a deep structure such as in a CNN.

2.3.5 *Fully connected Layer*

The fully connected layer consists of a number of neurons all connected to the neurons of the preceding and succeeding layers. Multiple fully connected layers, which are structurally similar to an MLP, are commonly used as a classifier at the end of the CNN.

A CNN is typically built using a combination of the above mentioned layers. The complexity of the task dictates the complexity and depth of the network. Each of the layers can be used in the network architecture as many times as required in any desired order. Together it forms a system which is capable of detecting features and classifying the input into pre-defined classes. It may also be tuned to assign embeddings of an arbitrary dimensionality to the input examples. This system can be trained end-to-end, thus tuning the feature detection and classification together.

In this work, a CNN is used to achieve a competitive performance in word spotting using the technique of transfer learning. The concept of transfer learning as well as its types have been explained in the next section.

2.4 TRANSFER LEARNING

Transfer learning is a broad term which encompasses the concept of using the knowledge gained from a source task and domain in order to achieve a high performance for a target task and domain.

According to [PY10] *domain* is defined to have two components, the feature space and the probability distribution of the elements in the feature space. This term is related to the nature of the data which is associated with the learning task at hand. A *task* consists of a label space and a predictive function which maps the data to the labels. This function is learned from the training data. Domains are said to be related if there is any relationship between their feature spaces. On the other hand, tasks are said to be the same if the label spaces and the predictive functions of both the tasks are the same.

In traditional machine learning, the source and target domains and tasks are the same. For example, the training and test samples belong to the same dataset and the exact model which learns from the training samples is used to map the test data samples to the labels. However, in several real-world applications, this common

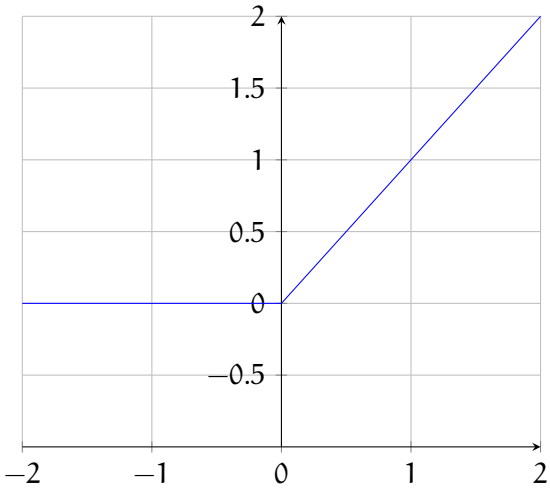


Figure 2.3.4: The Rectified Linear Unit (ReLU) activation function has a linear response to positive inputs. This helps to avoid the problem of vanishing gradients and is, hence, used as a substitute for the sigmoid activation function.

domain may not have enough training data which a different but related domain may contain. In another case, it may be much more difficult to produce labeled training data in one domain, but relatively easier in another related domain. The technique of transfer learning may be implemented in such cases in order to use the knowledge gained from a related source domain and task to accomplish a desired target task with the target domain.

Transfer learning or knowledge transfer have been categorized into 3 basic types by [PY10] as shown in Table 2.4.1.

Table 2.4.1: Types of Transfer learning

Types	Source and Target Domains	Source and Target Tasks
Traditional Machine Learning	the same	the same
Inductive Transfer Learning	the same	different but related
Unsupervised Transfer Learning	different but related	different but related
Transductive Transfer Learning	different but related	the same

- **Inductive Transfer Learning**

  Inductive learning aims at improving the learning of the target function using the knowledge of the source domain and target where the source and target tasks are different. The source and target domain labels are available in this case.

  Transfer learning is a setting in which several approaches can be implemented. The instance-transfer approach with Inductive learning states that although the source domain cannot be directly used for transferring knowledge, certain parts of it along with labeled target data can contribute to the target domain [PY10]. Another approach is the feature-representation-transfer, which aims at finding good feature representations that minimize the domain divergence and model error.

- **Unsupervised Transfer Learning**

  Unsupervised transfer learning aims at improving the learning of the target predictive function using the knowledge of the source domain and task where the labels of the source and target domain are not observable.

  Examples of unsupervised transfer learning can be seen in the examples of the feature-representation-transfer approach such as Self-taught clustering [DYXY08] and transferred discriminative analysis [WSZ08] algorithms. They transfer the problems of clustering and dimensionality respectively.

- **Transductive Transfer Learning**

  According to [PY10], transductive transfer learning aims to improve the learning of the target predictive function given the source task and domain where the source and target domains are different but the tasks are similar. Some unlabeled target domain data is required at training time in addition to labeled training data.

  An example of transductive learning is shown by the structural correspondence learning algorithm proposed by [BMP06]. It makes use of unlabeled target domain data to extract relevant features in order to reduce the difference between the domains [PY10].

Transfer learning is similar to the multitask learning approach i.e., the source and target domain are used to induce a predictive model in the target domain. However, the multitask learning approach aims at learning the target and source models simultaneously, whereas transfer learning attempts to achieve a high performance in only the target task by transferring knowledge from the source task.

An example of transfer learning in word spotting in the field of document analysis is shown in [KJ16]. One of the experiments described in this work uses synthetic data (source domain) to pre-train a CNN for word spotting in handwritten documents (target domain). Here, the network pre-trained using synthetic data is finetuned using the training partition of the handwritten dataset before evaluating the performance using the same handwritten dataset. The authors state that, in this case, transfer learning is performed from the synthetic domain to the real world domain. The performance of this CNN surpasses the state-of-the-art performance of the Support Vector Machine-based approach in word spotting described in [AGFV14].

Most of the tools and concepts introduced in this chapter are used for word spotting in handwritten documents in this work. The following chapter describes the concept of word spotting in detail and gives a brief idea about some of the established approaches in word spotting.

3

WORD SPOTTING

The term *word spotting* originally stems from the field of audio processing where it was used to detect certain key words in an audio stream. The concept of word spotting has received attention in the field of document analysis and handwriting recognition. It was first introduced to the field of handwritten document analysis by [MHRC96] and its use for indexing historic documents is discussed in [RM07]. Word spotting is defined by [RM07] as an approach which involves grouping of word images into clusters of similar words by using image matching to find the similarity. The word spotting approach in [MHRC96] treats collections of documents as a pool of word images. By comparing all images in this word image collection pair-wise, they are clustered in such a manner that ideally, images in a cluster are all assigned the same annotation. The clusters with content most interesting from the point of view of generating an index for the document are manually assigned a label. A partial transcription of the document collection by assigning the cluster labels to all word images contained in a cluster. This in turn, creates a partial index for the collection and allows a retrieval of text portions that contain the manually assigned labels. Hence, word spotting can be viewed as a retrieval task which is accomplished in the approach presented in [MHRC96] through clustering. In a typical retrieval task in the context of word spotting, one word from the collection is provided at a time as a query. This query made to a word spotting system generates a retrieval list from the given collection of words and every word image in this retrieval list is ranked according to the query. Eventually, each word in the collection is used as a query. The performance in word spotting is evaluated based on the relevance of the words in the retrieval list to the query and their order determined by the ranking. Ideally, a retrieval list should contain all the words from the given set matching the query and the best matches should have the top ranks. This is distinguished from a handwriting recognition task which aims at recognizing words in a handwritten dataset. The robustness of a recognizer lies in its ability to map a word from its image to its string representation and vice versa. For example, the task accomplished in [SF16] is a word spotting task, whereas that in [AF15] is a recognition task. In fact, word spotting aims to efficiently access textual information without processing and recognizing all contents of the document. It is especially suitable for handwritten documents, which text recognizers find especially challenging.

Word spotting can be achieved in a segmentation-based or a segmentation-free scenario. A segmentation-free environment considers entire documents as inputs. The document is not segmented into lines or words before being used as input for a segmentation-free word spotting system. On the contrary, a model used for word spotting in the segmentation-based environment takes single words as input. Several query representations have been introduced for word spotting. Among these, the two approaches associated with this work are Query-by-Example Query-by-String. In the Query-by-Example, the query is a word image and the relevance of the retrieved images is based on the visual similarity of the two word images. However, this approach has some limitations. In practical applications, an instance of the key-word must be identified by the user. If this word used as query does not appear frequently in a large collection, it is a tedious task for the user to manually extract a desired query image. Additionally, if the word does not appear in the collection, it may already be the solution for this task [ARTL13]. This limitation may be overcome by the Query-by-String approach. The string representation of a word is provided by the user as a query in this approach. A retrieval list is then generated from all the relevant word images in the collection. The earlier approaches involving Query-by-String focused on creating character templates from a typed string to synthetically generate an example of the query. Despite these approaches being user-friendly, they pose a challenge that the textual representation must be mapped on to the image representation of the word. Hence, one of the several approaches in word spotting aims to find a common subspace in which the image of a word and a string representing the word can be represented by the same point. The queries are then used to rank all images in the dataset and the retrieval list, thus obtained, is evaluated for its precision.

Several methods have been used for word spotting in handwritten documents. An early approach in this field used XOR-maps and Euclidean Mapping in order to spot key words in binarized handwritten images [MHR96]. The later approaches focus on word spotting in historic documents using sequential models such as Dynamic Time Warping [RM07] and Hidden Markov Models (HMMs) [RSP09]. These are unsupervised approaches, i.e. they do not require annotated training data. Since word spotting is based on image matching, other methods from computer vision also find an application. For example, an unsupervised approach which combines HMMs with the Bag-of-Feature approach is described in [RVF12]. In this approach, the document images are represented by local features. SIFT features extracted from densely sampled patches in the document image are used for this purpose. Next, a query is initialized and trained on the sequence of Bag-of-Features representations obtained from the bounding box for the query word image. Finally, the model is decoded on the entire

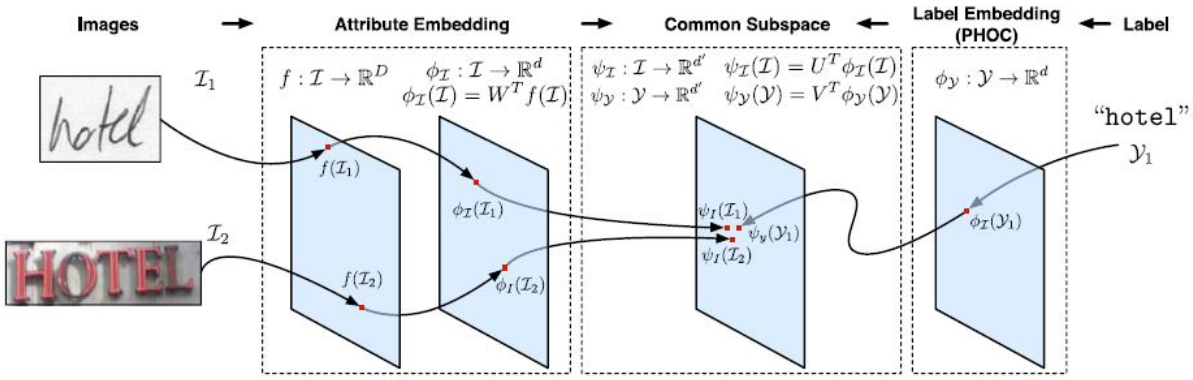


Figure 3.0.1: This image taken from [AGFV14] visualizes their approach where the image and text representation of a word is mapped through their respective representations on to a common subspace.

document collection. For the final word retrieval, the patches are ranked with respect to their scores and the performance of the model is evaluated for this retrieval list.

The unsupervised approaches in word spotting are, however, outperformed by supervised approaches at the cost of having to annotate data for training. One of the most prominent approaches in recent times is introduced in [AGFV14]. In this segmentation-based approach, word spotting is achieved by images and labels are embedded on to a common subspace. The word images are first encoded into feature vectors, which are then used along with label embeddings to learn Support Vector Machine-based attribute models. These are referred to as AttributeSVMs. The approach is visualized in Fig.3.0.1. First, the images are embedded in the attribute space where each dimension encodes how likely it is that a spatial region of a word contains a character, whereas the labels are represented with a binary vector called *Pyramidal Histogram of Characters* or PHOC. The PHOC representation introduced in [AGFV14] encodes the presence of a particular character in a spatial region of a string and is a key element used in this thesis. It will be explained further in detail in the following chapter. Both the embeddings have the same dimensionality. However, they are not perfectly comparable since the PHOC vector is binary and the attribute representation for the images is not. Therefore, they are then projected onto a common learned subspace where the representations are comparable and word strings as well as images which are relevant to each other are brought together.

However, the authors of [AGFV14] state that there is no need to use Fisher vectors or SVMs in particular, and that any encoding method which transforms the input image into its attribute representation could be used to replace them. The approach defined

in [SF16] confirms this statement by using a CNN to transform input images into their PHOC representation directly. The CNN called the PHOCNet which is introduced in this work is specifically designed for word spotting and has achieved state-of-the-art performance.

The PHOCNet is the primary tool used word spotting in this thesis. Further information about the PHOC representation, the PHOCNet and the other related academic contributions which have inspired this thesis most prominently are discussed in detail in the following chapter.

4 RELATED WORK

In this chapter, the established methods in word spotting and handwriting recognition are discussed, each of which have, in part, inspired the development of the approach presented in this work. The use of synthetic data for training a CNN-based model is the basis of the approach presented in this work. Hence, section 4.1 explains some of the approaches in word spotting which use Convolutional Neural Networks and discusses the features and merits of these methods. Section 4.2 discusses some of the approaches which use computer generated data to train a model in the field of document analysis.

4.1 APPROACHES USING CNNS

CNNs have been empirically shown to achieve exceptional results in several fields of computer vision such as image recognition, video analysis etc. Since word spotting is an image matching problem, CNNs find an application in this field. In fact, CNN-based methods have achieved state-of-the-art performance in word spotting. Some of these methods are discussed in this section. The choice of using the CNN as a model for word spotting in this thesis has been inspired by these related methods.

4.1.1 *PHOCNet*

The PHOCNet is a network specifically designed for word spotting. It has been shown to achieve a high performance using limited amounts of training data. It takes images of words as input and generates a PHOC representation of the words. Here, the approach using the PHOCNet is discussed which has achieved state-of-the-art performance in word spotting.

PHOC Representation

The PHOCNet maps images of words on to their PHOC representations. PHOC or Pyramidal Histogram of Characters is a binary representation of words introduced in [AGFV14]. An example of a three-level PHOC representation of the string *beyond* has been shown in Fig.4.1.1. It encodes whether a particular character is present in

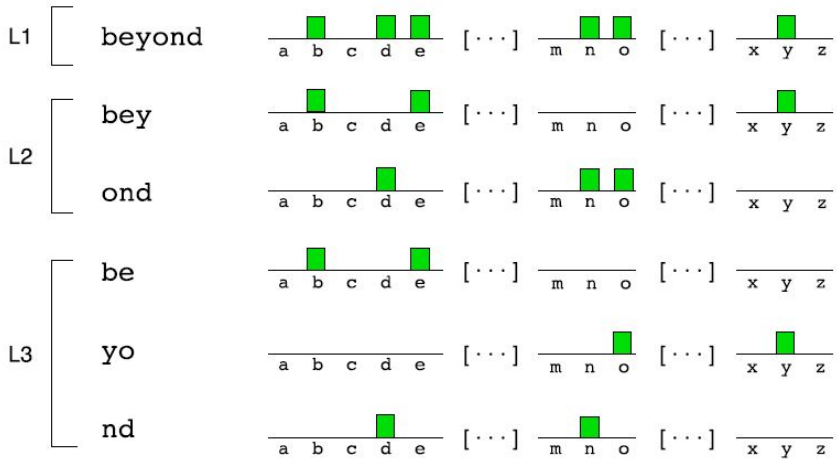


Figure 4.1.1: A 3-level PHOC representation of the word 'beyond'. L1 represents the first level where the complete word is considered, whereas L2, and L3 represent levels 2 and 3 levels at which the word is split into 2 and 3 parts respectively. A binary vector indicating the presence of a character in that split of the word is generated for each part in each level. These vectors are concatenated to obtain the PHOC representation. This image has been taken from [AGFV14].

a particular region of the word. Each of these characters is referred to as unigram. A frequently occurring combination of two such unigrams, which is called a bigram, can also be included in the PHOC representation. An attribute vector, or *Histogram Of Characters*, is developed for a word representing the presence of such attributes in the word from amongst a pre-defined set of attributes. This is done for multiple *levels* by dividing the word into an increasing number of splits at each level. For example, at level 2, the word string is split into 2 parts, i.e. a six-character word is divided into 2 parts containing 3 characters each as shown in Fig.4.1.1. An attribute vector indicating the presence of the characters within the half-word is developed for each half. Assume that only lower case letters of the Latin alphabet are used as unigrams to generate a PHOC vector in this example. The attribute vector generated for each split of the word at each level thus contains 26 *bins*, each representing a unigram. As shown in Fig.4.1.1, at level 2 (L2), the first half contains the letters *b,e,y*. From this, a vector with 26 elements is obtained. The values of all bins are zeroes except the 3 bins representing the letters *b,e,y* which have the value 1. Another 26 element vector is similarly generated for the next split of the word containing the letters *o,n,d*. This is repeated for a desired number of levels, i.e. the word may be split into 3 parts at level 3, 4 parts at level 4 and so on. These attribute vectors are concatenated to give the PHOC representation of the word. Learning the character attributes independently allows straightforward out-of-vocabulary word spotting and recognition. Out-of-vocabulary word spotting has formerly been accomplished using unsupervised techniques such as Dynamic Time Warping [RM07], Hidden Markov Models (HMM) [RSP09], and Bag-of-Feature HMMs [RRF13, RF15].

In [SF16], for the datasets in Latin script the authors use all 26 letters of the alphabet in the lower case, numbers 0 to 9, as well as 50 commonly occuring bigrams in levels 2,3,4,5 to generate the PHOC representation.

In order to gain a verbatim understanding, it is desirable for the string and image representations of the same word to be represented by the same point in a common subspace. In other words, the string and image representations of a word should result in the same PHOC representation. Hence, character attributes must be learned from the image of a word in order to assign a PHOC representation to it. Light is shed over the characteristics of the CNN which has been used to learn these attributes.

PHOCNet Architecture

The architecture of the PHOCNet is similar to that presented in [SZ14]. It has been designed to learn abstract features and to prevent overfitting. Similar to [SZ14], the PHOCNet contains an increasing number of filters from the lower to the higher layers

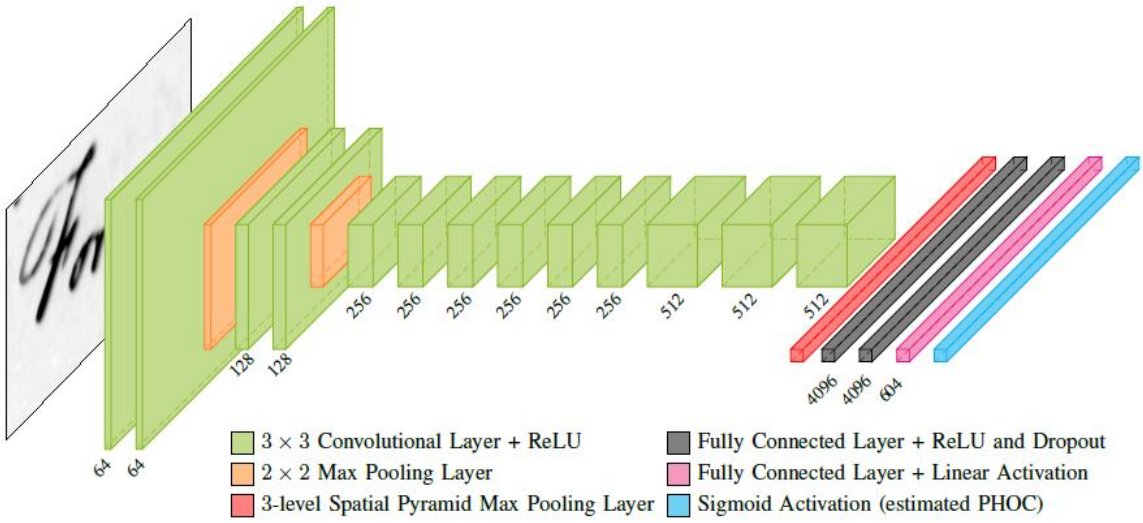


Figure 4.1.2: The figure shows the architecture of the PHOCNet. This image is taken from [SF16].

because of which the network learns fewer features for small receptive fields and a larger number of abstract features at the higher level. The PHOCNet also uses only 3×3 convolutions in the convolutional part of the network since this has shown better results as compared to larger convolutions [SZ14]. The smaller receptive fields impose a regularization on filter kernels, thus preventing overfitting. An important feature of the PHOCNet is the Spatial Pyramidal Pooling or SPP Layer. First introduced in [HZRS14], it enables the network to process inputs of varying size while still producing ouputs of a constant size, which is necessary for training the network. The output of the SPP layer with max-pooling consists of the maxima taken from local spatial bins of the input feature map. The entire feature map is divided into an increasing number of bins in subsequent steps for a fixed number of steps. Since the bins have sizes proportional to the input feature map, the number of bins, and hence the size of the output of the SPP layer, is fixed despite a varying size of input images. The structure of the PHOCNet is shown in Fig.4.1.2.

Training labels for CNNs in recognition tasks are, usually, 1-hot vectors signifying the target class of the input. However, multiple elements in a PHOC vector can be 1. Hence, the PHOCNet uses the sigmoid activation function instead of a softmax function [SF16]. Each output of the last fully connected layer of the PHOCNet represents an attribute in the PHOC. The sigmoid activation function is connected to each of these

outputs to give a pseudo probability of the presence of that attribute in that PHOC. In Fig.4.1.3, $\hat{a}_i$ is the pseudo-probability for the presence of attribute i in the given word.

The experiments conducted using the PHOCNet in [SF16] show that a CNN does not necessarily require large amounts of training data. The experiment on the George Washington dataset with 3645 training images outperforms other methods such as AttributeSVM-based word spotting[AGFV14]. Additionally, the same set of parameters can be used in all experiments. Hence, PHOCNet is robust with respect to parametrization. Moreover, a high performance of PHOCNet on a modern handwritten dataset written by multiple authors shows its robustness for recognition in a multi-writer dataset. The results for the PHOCNet experiments will be discussed in 6 as a benchmark for the work presented in this thesis.

4.1.2 *Triplet-CNN*

The triplet-CNN is introduced in [WB16] as a model for word spotting in a segmentation-based scenario. In this approach, word images and the text representation of the words are embedded in a common embedding space. The approach and the results achieved in word spotting using the triplet-CNN are discussed below.

Obtaining Embeddings using Triplet-CNN

For obtaining the image representation, the triplet-CNN, which is a set of three identical CNNs sharing weights, is used. The input to the triplet-CNN is a set of three images consisting of two of the same class (positive images) and one of a different class (negative image). Each of these images is propagated through one of the CNNs in the triplet-CNN each. The output of each of the CNNs is a descriptor for each of the images. The distance between the images is calculated using these descriptors. Using the SoftPN loss [BJTM16], the similarity that is smallest between the two positive images and the negative image is selectively back propagated as well as the distance between the two positive images. Thus a model is learned on triplets of images. One of the three CNNs is then used to extract a feature vector for images. In order to learn the embedding from this descriptor, a two-layer fully connected neural network is used. The embedding generated from the text representation of the word is used as the target output to train the neural network to map the image feature vector on to the embedding space. The hand-crafted representations PHOC and DCToW (Direct Cosine Transform of Words) [WB16] are used to generated words embeddings from text representations. Additionally, LSTMChar-Large model from [KJSR16] is used to learn a language model from separate text data in order to obtain two conceptually

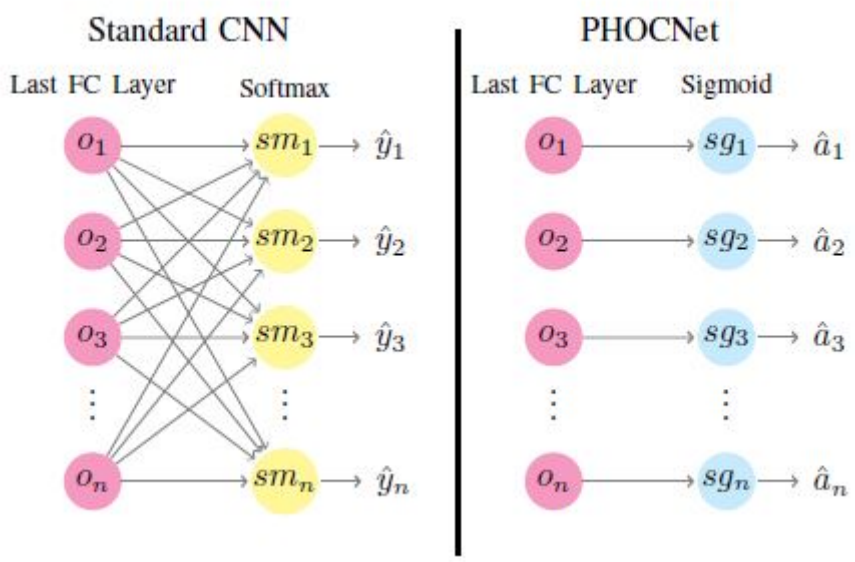


Figure 4.1.3: Difference between a softmax layer and sigmoid layer. The image is taken from [SF16].

different word representations. These embeddings are called the n-gram and semantic embedding respectively. The n-gram embedding groups the verbally similar words together whereas the semantic embedding groups the words based on their semantic similarity or relevance. The performance is evaluated and a comparison is drawn for all these text representations. This approach is visualized in Fig.4.1.4.

Results and Discussion

The results obtained by triplet-CNN are competitive to that obtained using the PHOCNet. In fact, the triplet-CNN outperforms the PHOCNet using atleast one embedding in experiments conducted for each of the datasets in QbE and QbS scenario.

One of the most interesting features of this approach using the semantic embedding is that the model is capable of retrieving words of semantic relevance to a query. For example, for a query *two* from a historic dataset, the top word classes retrieved include *two*, *twelve*, *twenty*, *ten*, as well as words signifying fuzzy quantities *several* and *some*. The authors state that word spotting using semantic embeddings may be desirable when the user does not have a definitive idea about what must be searched while browsing a collection of documents. The authors also state that using the spatial

pyramidal pooling layer, such as that used in the PHOCNet, would be a possible improvement for their model since it allows the input to the CNN to be of variable size.

To sum up, this is one of the several established methods which confirms that using a CNN as a model yields a high performance in word spotting. Besides, the approach is directly comparable to that using the PHOCNet since both perform word spotting in the segmentation-based scenario using a CNN. Hence, this work also provides a benchmark for the performance of the PHOCNet.

4.2 APPROACHES USING SYNTHETIC TRAINING DATA

This section discusses the methods which use synthetically generated data in order to training a model for text recognition. The approaches discussed here inspire the idea that it is possible to generate training data cheaply, thus limiting the need for the tedious task of annotating handwritten data and reducing manual effort.

4.2.1 *Training a Recognizer without Handwritten Data*

The focus of the work presented in [AF15] is on recognition of handwritten text without using handwritten training data. The authors investigate four main approaches including, computer generated text in different typefaces as training data, unsupervised adaptation, and using recognition hypothesis on the test sets as training data. The idea behind this work as well as the conclusions drawn point us toward developing a method to recognize handwritten text with minimal manual annotation effort.

Recognition in Arabic Script

The authors define four approaches, each building on the success of the previous. First, text is generated synthetically with several different font typefaces. One classifier is trained for recognition of text in each typeface. This sheds light over how the classifiers trained on computer generated text perform depending on the visual complexity of the typefaces.

In the next approach, a recognizer is trained for all font typefaces. This helps to determine how the recognizer trained on a dataset using diverse fonts performs in comparison to the classifiers trained on text generated using single typeface.

Next, unsupervised HMM adaptation techniques are used to enhance the performance of the recognizer. These techniques use the new data they see during recognition

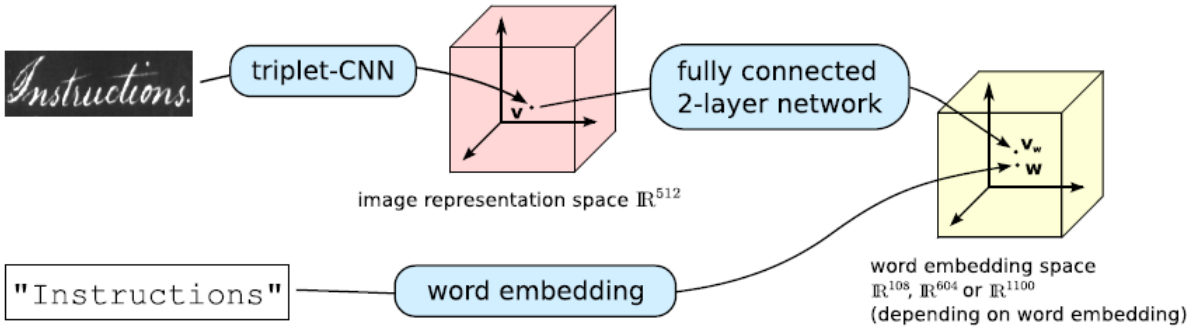


Figure 4.1.4: This image visualizes the approach in word spotting using the triplet-CNN. The image embedding is obtained by propagating the word image through one of the CNNs from the triplet-CNN and then by passing the feature descriptor obtained at the output of the CNN through a 2-layer neural network. The word embedding is either hand-crafted using the PHOC representation [AGFV14], the DCToW [WB16] or with the trained LSTMChar-Large model introduced in [KJSR16]. This image is taken from [WB16].

in order to re-calibrate the trained parameters. The model parameters related to the data part i.e. the mixture means and variances, are adapted, whereas the model length and the transition probabilities are the same. The task of adaptation is to update to the new model parameters by fine tuning the original model parameters such that the likelihood of the new adaptation data is maximized. Maximum Likelihood Linear Regression (MLLR), a common technique used for HMM adaptation, estimates linear transformations for the mixture means and variances in order to adjust them such that they better fit the adaptation data.

Finally, a recognition hypothesis is generated for the handwritten test set using the system developed through the first three approaches. This hypothesis is then used to train the recognizer. When the recognizer achieves a reasonably satisfactory performance on the test data, using the test set to bootstrap the system is hypothesized to be an effective approach. The authors state that in this regard, the first three approaches can be seen as initialization steps for the recognizer. This step may also be repeated iteratively in order to improve the model performance on the condition that the approach leads to a reasonable improvement [AF15] in the first iteration.

The recognizer model used in this work is the Arabic sub-character model based HMM recognizer as presented in [ARFM13, AFM14]. The recognizer is a continuous HMM system built using HTK tools [YEG+02]. It uses 97 sub-character HMMs to model all the characters and the variations in character shapes. A sub-character model

based HMM system is used since the authors of [AF15] claim that it is more robust and effective, especially under constrained training environments.

Results and Discussion

One of the aims of this thesis is to develop a method to reduce the manual annotation effort that is taken to prepare training data for CNNs. The work [AF15] does not only show that data can be generated synthetically, but also comments on the nature of a dataset that can yield the best results. For example, in the first approach, the authors train a classifier for text in each of eight font typefaces. Since both, the machine printed and the handwritten text in Arabic script are cursive, the machine printed text resembles the handwritten text, although with a smaller degree of variability. The results show that irrespective of the complexity of the typeface in terms of visual appearance, the classifiers yield a low percentage Word Recognition Rate (WRR), the maximum being 26.92%.

However, in the second approach of training one classifier for text in multiple typefaces, the performance of the model improves greatly. It achieves a WRR score of 61.35%. This indicates that a recognizer trained using diverse dataset with a large variance in writing performs better as compared to a recognizer trained using any single typeface.

The third approach, where the parameters of the model trained with text in multiple typefaces are recalibrated using HMM adaptaion, achieves a WRR score of 70.47%. Finally, the recognizer iteratively finetuned with handwritten test data leads to the best performance among all four approaches with a WRR score of 90.23% after 5 iterations. This shows that a recognizer trained on computer generated data is a good initialization for the model which can then be reinforced well using additional knowledge.

To sum up, this work inspires the idea that training data can be generated synthetically. Using such a dataset may lead to greatly reduced the manual effort needed for annotation of training data. Moreover, the conclusion drawn by [AF15] that a better performance can be achieved using a diverse training dataset with multiple typefaces instead of that with a single typeface provides an insight into the nature of a suitable synthetic training dataset. A dataset in the Latin script fulfilling these requirements has been introduced by [KJ16]. The characteristics of this dataset called HW-SYNTH will be discussed in the further chapters.

4.2.2 *Word Spotting in Latin Script*

Early approaches using synthetic data in document image analysis aimed at increasing the training data size of already existing datasets in order to improve classification performance in handwriting recognition tasks [VB03, VB04] or supplying datasets for new script types for which obtaining document images and annotations is difficult [MP01].

One of the prominent approaches which involves training of a CNN using synthetic data in the Latin script is defined in [KJ16]. The main aim of this work is to predict a similarity between two handwritten documents written by different individuals. A similarity score is computed for a pair of document images by identifying patterns of text re-usages between them irrespective of variations in word ordering, word morphology, layout and paraphrasing of the text. The primary tool used in this work is a CNN called the HW-Net defined by the authors. The HW-Net is used to extract a feature representation from word images. To account for the lack of handwritten data for training the CNN, the authors generate a synthetic dataset in Latin script called the HW-SYNTH. This dataset is an important aspect of the approach presented in this thesis, and hence will be described in detail in chapter 6 along with the other datasets used. In the approach described in [KJ16], this training data is normalized by imitating the process of stemming in the visual domain by labeling the training data in terms of the corresponding root words given by the Porter stemmer [Por80]. The root of a word is obtained from a word string by stripping off common suffixes from the word. The HW-Net is trained using this normalized data which then learns the visual representation of word images and their invariance to inflectional suffixes. This network is then finetuned over the training partition of the handwritten dataset over which it is to be evaluated. This model is used for word spotting in both the documents and it retrieves semantically similar words (normalized words). The similarity scores are calculated for the documents in terms of number of matching words as well as the location of words in the document images. The authors demonstrate the use of this approach in detecting plagiarism in handwritten assignments.

The most significant feature of this work with respect to this thesis is the synthetic dataset generated by the authors. Moreover, this work also shows that this synthetic dataset can be used for transfer learning for word spotting in handwritten documents to yield a high performance. In fact, the performance in word spotting using the fine-tuned HW-Net surpasses the state-of-the-art performance of the SVM-based approach shown in [AGFV14]. Hence, the method and results of the intermediate stage of word spotting presented in [KJ16] play an important role in determining the methodology to be used in this thesis.

The established methods described above help shape the idea from which the approach presented in this work stems. Some of the concepts from these established approaches described in this chapter are brought together in order to develop a method to reduce the amount of handwritten annotated data required for training a supervised system for word spotting. This methodology is further explained in the next chapter.

5 METHOD

The previous chapters explained the various concepts and tools which have inspired this approach for word spotting. This chapter gives an introduction to the proposed methodology in word spotting based on the related work. The interpretation of weak supervision in the context of word spotting is defined in section 5.1. Additionally, it defines a two-phase proposed method for word spotting which aims at limiting the annotated training data to a minimum and consequently reducing the manual effort taken to prepare that data. Section 5.2 gives a detailed account of the regularization techniques used in this method in order to avoid overfitting of the learned model on the training data.

5.1 LEARNING WITH WEAK SUPERVISION

In this approach, a system is introduced which retains the ability of the supervised techniques to adapt to visual variation in data while keeping the amount of training data low, and hence reducing the annotation effort. For this purpose, a CNN is trained using weak supervision. In order to explain weak supervision in the context of word spotting, the concepts of unsupervised and supervised learning are briefly recapitulated. Unsupervised learning in the context of machine learning implies that the learning of features does not require any annotated training labels. Generating a Bag-of-Features representation [RATL15, ARTL13, AGFV14] is an example of unsupervised learning of features. On the other hand, with supervised learning, a model is trained with the help of labeled training data which typically requires human effort. For example, a partition containing approximately 75% of words in a dataset is manually annotated and used to train a convolutional neural network for word spotting, as shown in [SF16]. In the context of word spotting, weak supervision may be defined as training with minimal manual annotation effort. The learning, in this case, requires labeled training data. However, the amount of data to be annotated and the effort for computation is much lower than what is used for supervised learning. As a result, the human effort and time taken in order to prepare the training data is, also, reduced to a large extent.

In the approach presented here, a CNN is trained with weak supervision, i.e., using very little training data while retaining a high performance. This eliminates one of the crucial disadvantages of using CNNs: having to prepare a large amount of training data. The following section explains the salient features of a CNN and why that makes it the preferred choice of model for the proposed approach.

5.1.1 *Necessity for CNN*

Training a CNN is a supervised approach. Supervised approaches have an advantage over unsupervised approaches since they are able to differentiate between the types of variances in the given dataset. This leads to the question: what are the different types of differences? And why is it advantageous to be able to differentiate between them? Variations in a dataset can be classified into two types, namely, inter-class and intra-class variation. Intra-class variation, in the context of word spotting, may include the difference in the visual appearance of one word written by multiple authors because of the difference in their writing styles and the writing materials they use. Moreover, intra-class variation may also include the visual difference that exists between two instances of one word written by the same author because of human error. However, inter-class variation is the difference between the visual appearance of the words, or classes, in the dataset. Unsupervised techniques are unable to differentiate between these two types of variations of a dataset. On the other hand, supervised techniques are able to identify different classes because of the annotated training labels. For example, if two word images representing the same word have different visual appearances, the training labels enable the system to identify the images as belonging to the same word class. On the other hand, if two word images representing different words are similar in their visual appearance, the training labels separate these two images into their respective word classes. Hence, they are inherently designed to distinguish between inter-class and intra-class variation of the given dataset.

All supervised techniques are thus capable of adapting to a large intra-class variation or differentiating between classes despite a small inter-class variation in the dataset. However, amongst the supervised techniques, CNNs have been empirically shown to achieve a high result. Additionally, as discussed in the previous chapters, the presented work uses transfer learning in order to achieve training with weak supervision. Transfer learning with CNNs is achieved in a straightforward manner since the parameters of the network retain the knowledge from the source domain and can be directly transferred for use in the target task or for finetuning in the target domain. Due to these advantages that a CNN poses over other approaches, it is used in this work.

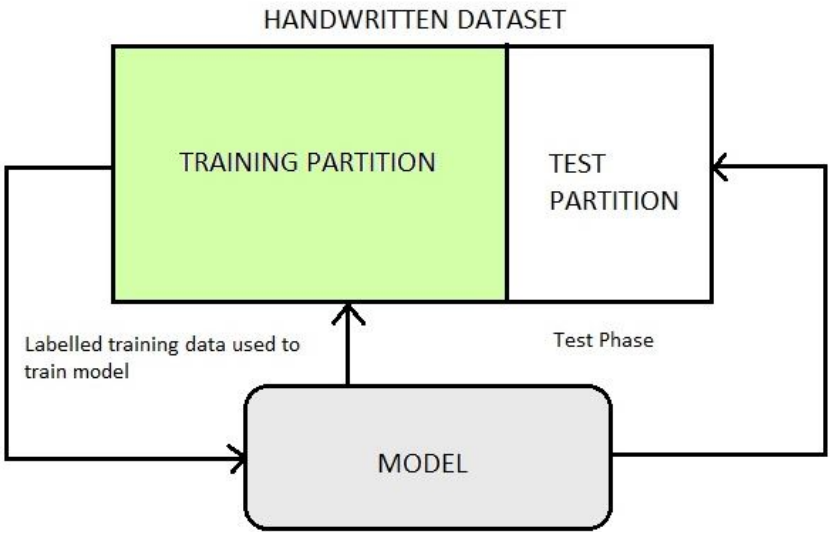


Figure 5.1.1: Abstracting of the method used by [SF16]

5.1.2 *Transfer Learning using Synthetic Data*

The aim of the thesis is to minimize the amount of training data required to train a CNN for word spotting. The first step toward achieving this is discussed here. For this purpose, a CNN is trained, specifically the PHOCNet introduced in [SF16]. As discussed in chapter 4, the PHOCNet is the first system for word spotting which can be trained end-to-end. In [SF16], the PHOCNet has been trained on the training partition of a dataset and the performance of the trained model is evaluated using the test partition of the same dataset. This can be visualized in an abstract manner in Fig.5.1.1.

In the approach presented here, the PHOCNet is initially trained with synthetically generated data. For this, a dataset of synthetically generated word images called HW-SYNTH is used. It has been first introduced in [KJ16]. This diverse dataset consists of several thousands of word images, which are expected to represent the variation in visual appearance seen in handwritten documents. Yet, no manual annotation of words is required. One may argue that this cannot be considered an unsupervised approach in the context of word spotting, since the words are annotated, albeit synthetically. On the other hand, the word images are generated synthetically from defined word classes. This means that if the generation of images is considered a part of the learning process, there is no manual supervision involved once the word classes have been

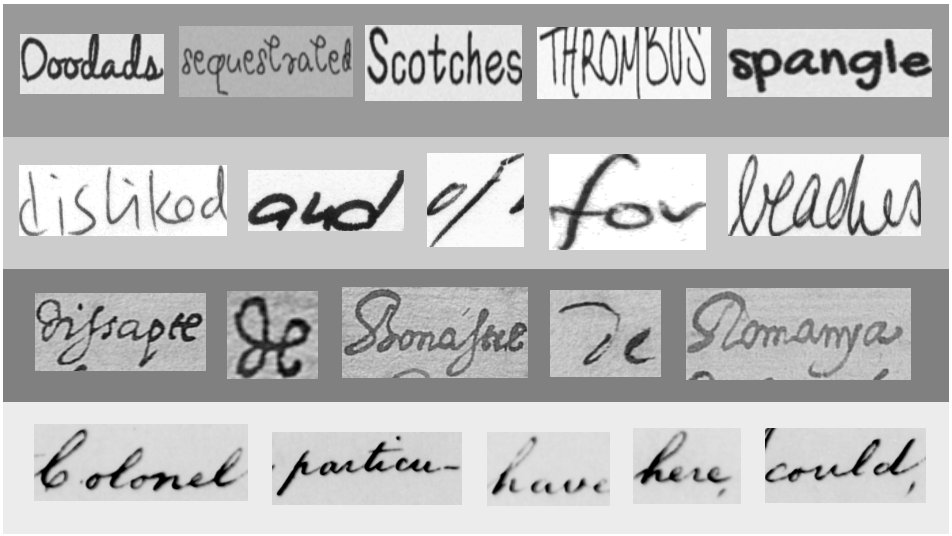


Figure 5.1.2: Top to bottom: Randomly sampled word images from the datasets HW-SYNTH, IAM, Esposalles and George Washington respectively. This image shows the differences in visual appearance of the contents of the four datasets.

chosen. In this work, this approach is defined as weak supervision. The question that arises is: How well does the model trained only on the synthetic data adapt to the real-world handwritten data?

The performance of this trained model is evaluated using handwritten datasets. This makes up the first phase of the approach and can be visualized in Fig.5.1.3. According to [PY10], given a source domain and task, transfer learning aims to improve the learning of a target task in the target domain. The authors also define a domain to have a feature space and a probability distribution. One can assume that there is a difference between the distributions of the data in synthetic and handwritten datasets in terms of the words used as well as the visual appearance of the word images. The difference in the visual appearances of the contents of all four datasets is shown through randomly extracted word images from each of the datasets in Fig.5.1.2. Based on these assumptions, the synthetic dataset is defined to be of a different domain as compared to each of the handwritten datasets. The synthetic data is said to be of the source domain whereas the handwritten data is of the target domain. Since the synthetic data in the source domain is used to enhance the target task of mapping handwritten word images on to a PHOC representation, this approach can be defined as transfer learning. The source and target tasks have the same predictive function, i.e. the PHOCNet and the same feature space, i.e. PHOC vector space. This means that

the source and target tasks are the same. Hence, specifically, this approach is defined as Transductive transfer learning.

This network model is capable of recognizing handwritten words to some extent, but it can be expected to face limitations due to the difference in the visual appearance between the synthetically generated and handwritten word images.

5.1.3 *Finetuning with Handwritten Data*

In order to improve the performance of the model trained only on the synthetic data, the approach of finetuning is considered. A major advantage of finetuning is that the amount of in-domain training data used to finetune a pre-trained network is much smaller as compared to what would, otherwise, be required to train the network from scratch. In order to introduce the characteristic visual appearance of a handwritten dataset to this model, it is further trained using a small subset of the training partition of that dataset. This makes up the second phase of the presented approach and can be visualized in Fig.5.1.4. Since this approach uses very few manually annotated training labels, the supervision is weaker as compared to that using the entire training partition.

The evaluation of this finetuned model using the corresponding test partition is expected to show a large improvement in the performance of the network. The hypothesis is that the training of the network parameters using synthetic data is a good initialization for the parameters to be finetuned over real-world data. Hence, it follows that the network parameters approach an optimum within half the number of iterations using fewer training data samples as compared to the first phase of training. Consequently, the manual effort taken in order to annotate the training data is reduced to a fraction of the training partition of the handwritten dataset while retaining the high performance. Additionally, due to the low number of iterations required to finetune the network, the time required for attaining a high performing word spotting model is, also, greatly reduced once the model pre-trained on the synthetic data is made available.

5.2 REGULARIZATION

The complexity of the PHOCNet makes it prone to overfitting. Hence, the following techniques are used for regularization.

Dropout [SHK+14] in fully connected layers acts as a regularizer. The activations of a layer with dropout are randomly set to 0. Any neuron following such a layer cannot

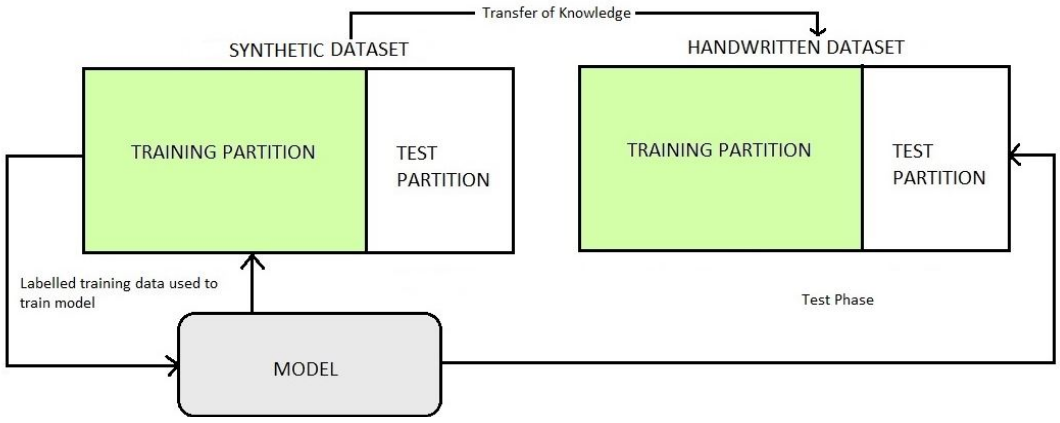


Figure 5.1.3: Training the PHOCNet on synthetic data and testing on handwritten data.

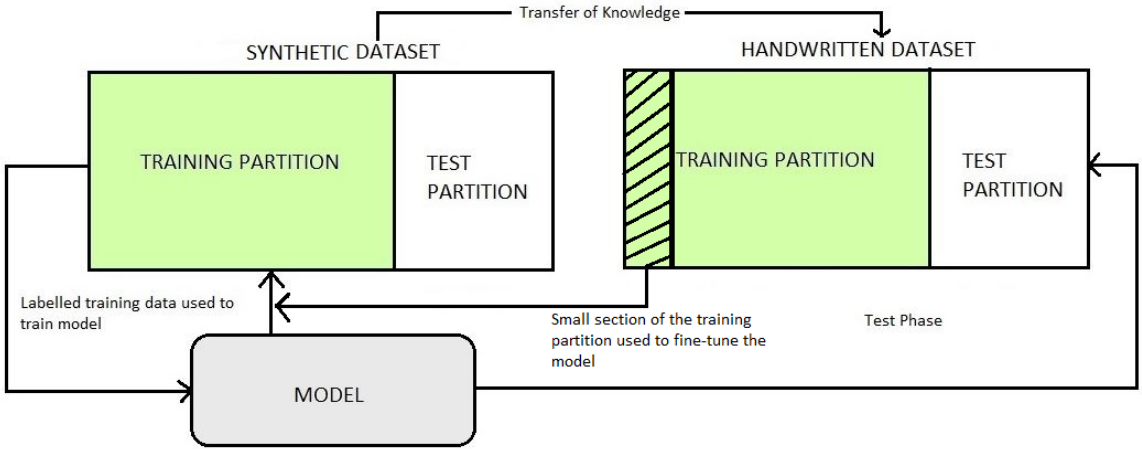


Figure 5.1.4: Model trained on synthetic data is finetuned using a few examples from the handwritten dataset. The performance is evaluated using the test partition of the handwritten dataset.

rely on neurons in the previous layer to be active for a given input. Similar to the approach in [SF16], a dropout of 0.5 is applied to all fully connected layers except the last.

The set of training images used for finetuning the PHOCNet is also augmented. The method of augmentation presented here serves two purposes. It not only imposes regularization on the network, but also generates additional training images from the existing images without extra annotation effort. the expected number of images in each class of the augmented training set is set to the maximum number of images in a class in the original training set. The images in all the other classes are then augmented to match that number. This concept can be visualized in Fig.6.4.5. For augmentation, random images are sampled from each class and an affine transformation is applied to each image, similar to [SF16]. The relative coordinates (0.5, 0.3), (0.3, 0.6) and (0.6, 0.6) are selected from the image to be transformed and each coordinate value is multiplied with a random number drawn with uniform distribution within the range [0.8, 1.1]. This method of augmentation is shown to achieve good results in [SF16]. Thus, with this method of augmentation, each class in the augmented training set contains as many images as the class with the maximum number of images in the original training set. In some early experiments, it was observed that finetuning the CNN with an augmented training set consistently shows better results during the evaluation phase as compared to training without augmentation.

The next chapter discusses the experimental set-up and to implement the method described here. This helps to quantify and verify the success of this method. Additionally, experiments have also been set up to determine the use of certain experimental parameters over others.

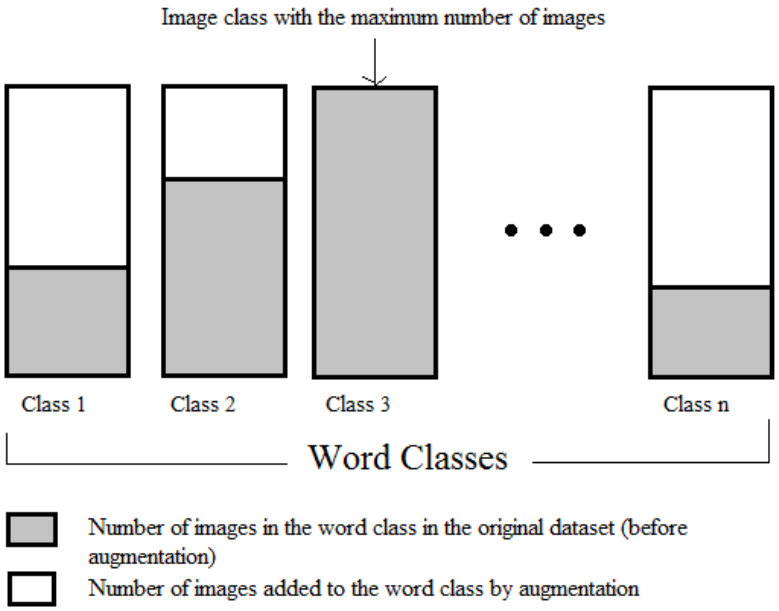


Figure 5.2.1: Augmentation of a dataset. The total number of images in a word class of the augmented dataset is indicated by the complete (vertical) length of the box representing the word class.

6 EXPERIMENTAL EVALUATION

The previous chapter describes the proposed method for word spotting which limits the amount of manual effort that is taken to prepare training data. A further description of the experimental set-up is provided in the current chapter. Additionally, the conclusions drawn from these experiments provide a brief insight into the behaviour of the PHOCNet and justify the parameters of the proposed experimental set-up.

The section 6.1 gives a description of the datasets on which the experiments have been conducted. This includes popularly used handwritten datasets as well as a synthetically generated dataset which is a crucial component of the presented approach. Section 6.2 gives a description of the word spotting protocol. Section 6.3 provides details about the training set-up of the experiments, whereas section 6.4 elaborates on the experiments conducted and lists the results. Finally, the results from the presented method are discussed in detail in section 6.5.

6.1 DATASETS

HW-SYNTH is a synthetically generated dataset introduced in [KJ16] consisting of 1 million images. The annotations in the dataset consist of all 26 letters of the Latin alphabet and the numerals 0 to 9. It has been created using a set of 10,000 words from the Hunspell dictionary such that no word in the set is repeated. Each word from this set is randomly sampled using 100 out of a collection of 750 publicly available fonts. These images are rendered by varying the inter character space, the stroke width, and the mean foreground and background pixel distributions. Subsequently, the images are smoothed using Gaussian filtering. In order to learn a case sensitive model, every word class is rendered using all letters in the lower case, only the first letter capitalized, and all letters capitalized. The data partition used for training consists of 750,000 images. No manual effort is needed for annotating the word images since each word image is generated from a pre-defined word string.

The first handwritten dataset is the **George Washington dataset**[1] which consists of 4860 words. They are a part of a 20-page document of correspondence between George Washington and his associates. Since there are no official available training

1 http://ciir.cs.umass.edu/downloads/old/data_sets.html

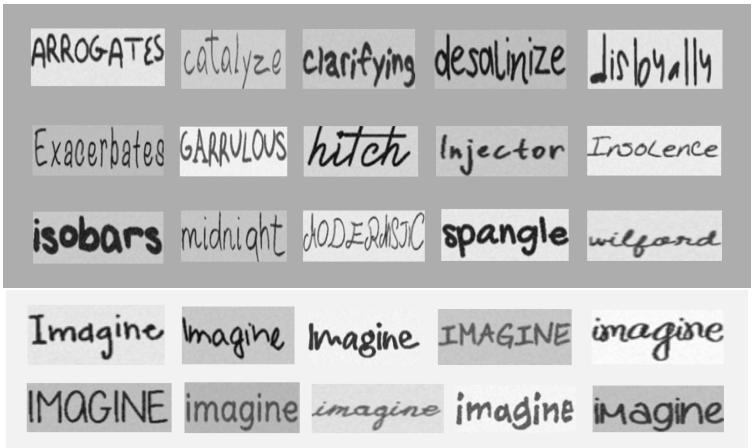


Figure 6.1.1: Top: Examples of random words in the HW-SYNTH dataset. Bottom: Examples of a single word from the HW-SYNTH dataset showing a variation in fonts and image rendering.

and test partitions for this dataset, the four fold cross validation method presented in [AGFV14][2] is used. The training and test partitions in all four folds contain 3645 and 1215 images respectively. As the documents in the George Washington dataset are obtained from the letterbook 2, which is not an original, but a later re-copied volume, it can be assumed that the dataset has been produced by a single writer.

The second handwritten dataset is the historic **Esposalles database** [RFS+13][3]. It is a multi-writer dataset consisting of 32,052 training images and 13,048 test images. The database is an ancient marriage license register written between 1451 and 1905. This is the only database used which consists of words in the Spanish language as opposed to other datasets which are in English.

The third handwritten dataset is the **IAM Handwritten Database**[4] written by 657 writers. It is a modern dataset which consists of 115,320 words. Apart from the difference in time periods in which the contents of modern and historic datasets are written, there exist a few key differences which set modern datasets apart from historic datasets. For example, historic datasets require a large manual effort as compared to modern datasets as well as expertise in historic literature for annotating the text. Additionally, historic documents show a difference in the writing style and visual appearance of words as compared to modern datasets. The data in IAM has been

2 Partitions available at https://github.com/almazan/watts/tree/master/data
3 http://dag.cvc.uab.es/
4 http://www.iam.unibe.ch/fki/databases/iam-handwriting-database

generated solely for the purpose of academic experimentation by having several people copy some provided articles in a specified manner [RMR06]. Therefore, deriving word-by-word annotations for such a dataset is easier as compared to historic datasets. Moreover, it is also easier to expand such a modern dataset by adding text from more authors. For the purpose of training, the official partition for writer independent text recognition is used. This partition provides 60,453 training words. Due to the large intra-class variance within the dataset, it is a particularly challenging dataset for word spotting.

6.2 WORD SPOTTING PROTOCOL

The experiments conducted on the above mentioned datasets follow a standard word spotting protocol. The details of this protocol are explained in this section.

The PHOCNet is evaluated for QbE and QbS in the segmentation-based case following the protocol in [AGFV14]. In the first phase, the PHOCNet is trained on the HW-SYNTH dataset for 80,000 iterations. For QbE, each image in the test partitions of the handwritten datasets is used as a query. The remaining images in the test partition are ranked using the cosine metric. This is achieved by passing all images in the test partition through the PHOCNet and obtaining their PHOC representations at the output of the network. One image from the test partition is used as a query at a time. Then the PHOC vectors for all the remaining images (non-query images) in the test partition are compared to the PHOC vector for the query image using the cosine metric to determine the similarity of each of the non-query images to the query image. The images are then ranked based on the cosine distance such that those most similar to the query image are assigned the top ranks. This process is repeated for all images in the test partition, i.e., all test images are used as a query once. However, words which appear only once in the test set are not used as queries, since there would be no relevant retrieved image for that word. A retrieved image is said to be relevant for the given query if they belong to the same word class. For QbS, all unique transcriptions for words in the test set are extracted and their PHOC representations are used as queries to rank all images on the test set.

The evaluation metric used for this approach is the mean Average Precision (mAP). The Average Precision (AP) is a metric for the correctness of a retrieval list for a single query. Mathematically, it is defined as the area under the precision-recall curve associated with that query. The mean Average Precision is the mean of the AP scores over several queries. An ideal retrieval list has an AP score of 1 if all the items relevant

to the query are included in the list and if all the relevant items are assigned the top-most ranks. The AP score can be calculated using the following equation:

$$AP = \frac{\sum_{i=1}^{N} P(i) \times R(i)}{m} \tag{6.2.1}$$

where $P(i)$ is the precision at position i in the list, $R(i)$ is the relevance indicator and m is the number of all relevant items included in the retrieval list. $R(i)$ takes binary values, i.e. 1 if the item is relevant to the query and 0 otherwise. The precision at each position is calculated as the fraction of the number of relevant items counted till that position over the index of the position in the list. Then the precision is multiplied by the relevance indicator associated with the item at that position. All these products are averaged over the total number of relevant items in the retrieval list to obtain an AP score for the retrieval list.

For example, consider the three cases of retrieval lists obtained in Fig.6.2.1. The black retrieved items are relevant to the query whereas the white items are not relevant. In this case, it is assumed that all relevant items are included in the retrieval list. In the first case shown in Fig.6.2.1a, the relevant items are assigned the top ranks. Hence, they have the highest AP score of 1 calculated as follows:

$$AP = \left[(\frac{1}{1} \times 1) + (\frac{2}{2} \times 1) + (\frac{3}{3} \times 1) + (\frac{3}{4} \times 0) + (\frac{3}{5} \times 0) \right] /3 = 1 \tag{6.2.2}$$

In the second case shown in Fig.6.2.1b, all the relevant items are assigned the last ranks. This is the worst case scenario amongst the given examples. Hence, the AP score calculated for this retrieval list is the least among all three cases. It is given by:

$$AP = \left[(\frac{0}{1} \times 0) + (\frac{0}{2} \times 0) + (\frac{1}{3} \times 1) + (\frac{2}{4} \times 1) + (\frac{3}{5} \times 1) \right] /3 = 0.477 \tag{6.2.3}$$

The AP score is similarly calculated for the third case shown in Fig.6.2.1c as follows:

$$AP = \left[(\frac{1}{1} \times 1) + (\frac{1}{2} \times 0) + (\frac{2}{3} \times 1) + (\frac{2}{4} \times 0) + (\frac{3}{5} \times 1) \right] /3 = 0.75 \tag{6.2.4}$$

Although one of the relevant items retrieved in this case is assigned the top rank, it is not an ideal retrieval list since the items not relevant to the query are ranked higher than atleast one relevant item. The retrieval list may be assigned the lowest AP score of 0 if no relevant items are retrieved at all.

(a) All relevant items are assigned the top ranks.

(b) All relevant items are assigned bottom ranks

(c) Relevant items are assigned ranks 1,3,5

Figure 6.2.1: The Average Precision score calculated for 3 retrieval lists of 5 items each. The black items are the relevant items whereas the white items are not relevant with respect to a query. Assume that in this case all the items relevant to the query have been retrieved. The items are arranged in order of their ranks with the left-most item being assigned the top rank.

6.3 TRAINING SET-UP

The initialization of parameters of a network plays a crucial role in learning the model. The biases of the PHOCNet are initialized to zero whereas the weights are randomly sampled from a normal distribution with mean 0 and variance $\frac{2}{n}$ [GB10]. Here, n is the number of parameters in the layer to which the given parameter belongs. PHOCNet is trained using stochastic gradient descent with a batch size 10. The first phase of training with the HW-SYNTH dataset is run for 80,000 iterations. The initial learning rate is set to 10^{-4} and is divided by 10 after 70,000 iterations. The momentum is 0.9 and the weight decay is $5{\cdot}10^{-5}$. The second phase of training uses the momentum vector from the previous training phase, and a constant learning rate of 10^{-5}. The experiments are carried out using a single Nvidia GeForce GTX 1080 GPU.

The word images in the synthetic dataset have a fixed size of 48×128 pixels, whereas those in handwritten datasets have varying, and often larger sizes (depending an the resolution they were scanned with). In order to introduce a suitable variation in image size to the model trained on purely synthetic data, the images in the synthetic dataset are scaled by a random factor within the interval $[1,2)$ before providing it as input to the PHOCNet.

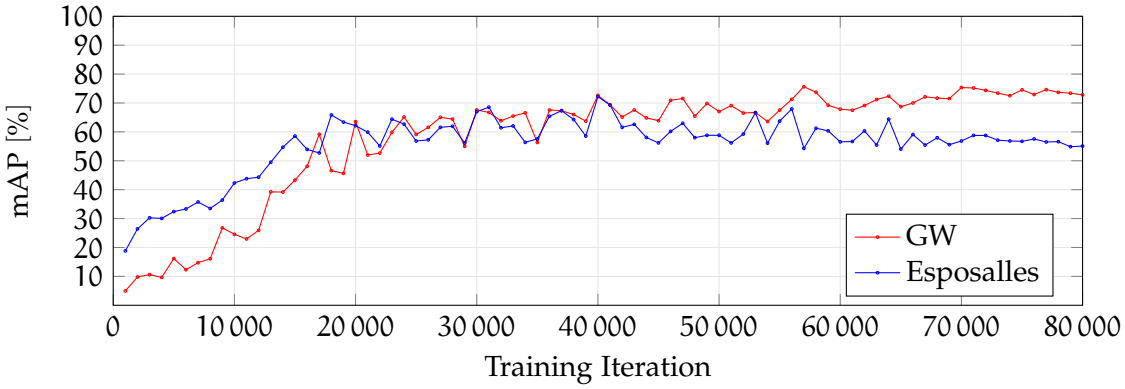


Figure 6.3.1: Mean Average Precision values for evaluating the performance of the PHOCNet trained only on the IAM dataset on the two historic datasets. The results are obtained for the QbE scenario.

6.4 EXPERIMENTS AND RESULTS

The experiments conducted in this work serve two broad purposes. Firstly, the first three experiments are conducted in order to gain a deeper understanding of the effects of transfer learning using a CNN. The first experiment shows the initial approach toward understanding transfer learning by transferring knowledge gained from a modern dataset on to a historic dataset. The next two experiments show the performance of the PHOCNet evaluated on handwritten datasets, when the network is trained using only the synthetic data and when this trained model is finetuned using the handwritten data. The other experiments are conducted in order to justify the use of some parameters of the experimental set-up presented in this work.

6.4.1 *Transferring Knowledge from Modern to Historic Data*

As discussed in section 6.1, it easier to generate and annotate a dataset such as the IAM handwritten database than it is to annotate a historic dataset. Hence, in an initial approach to understand the effects of transfer learning, an experiment is set up to train the PHOCNet for 80,000 iterations using the entire training partition of the IAM dataset and the performance is evaluated in the QbE scenario using both historic datasets, George Washington and Esposalles. The mAP scores are recorded for every 1000 iterations. The result is visualized in Fig.6.3.1. At 80,000 iterations,

the performance evaluated on the George Washington dataset yields a mAP score of 72.82% and that on the Esposalles dataset yields 55.09%.

It can be inferred from the results that the network performance steadily improves over the course of the training. That is, the PHOCNet steadily learns to map the word images from the historic datasets on to their PHOC representations by being trained with using a modern dataset. Based on the assumption that all three datasets have different distributions in terms of visual appearance, words used etc. they are said to belong to different domains. The difference in the visual appearance of the contents of the datasets alone is visualized in Fig.5.1.2. Hence, in the context of this experiment, IAM dataset belongs to the source domain, whereas each of the historic datasets belongs to a different target domain. Here, the task is to map the word images on to their PHOC representations. Since the source domain is used to improve the task in both the target domains, this process is defined as transfer learning in both the cases. Although the performance evaluated at 80,000 iterations is not comparable to those achieved by the established approaches in word spotting, this experiment shows that knowledge can be transferred from one domain to another to some extent for the task of word spotting.

Although a dataset such as IAM is annotated with less effort as compared to historic datasets, it still costs a considerable amount of manual effort to generate the data explicitly for academic experimentation. Hence, the next experiment investigates the effects of transfer learning in word spotting without using handwritten data.

6.4.2 *Training with Synthetic Data*

In order to achieve a high performance in word spotting using the PHOCNet, a large partition of a dataset must be used for training. This means that a large partition of the dataset needs to be manually annotated. This is a tedious task, especially for historic datasets, as discussed in section 6.1. A possible solution to reduce the amount of handwritten training data required is to train it using data which is easier to obtain and annotate as compared to the real-world data and then test the performance for the handwritten data. But, the question arises whether the network trained with one dataset can perform well when evaluated using another dataset. To answer this question, the PHOCNet is trained for 80,000 iterations using the synthetic dataset HW-SYNTH and evaluated using the test partitions of each of the handwritten datasets. As explained in chapter 5, this process is defined as transfer learning. The mAP scores for evaluation of the PHOCNet performance recorded every 1000 iterations is shown in Fig.6.4.1.

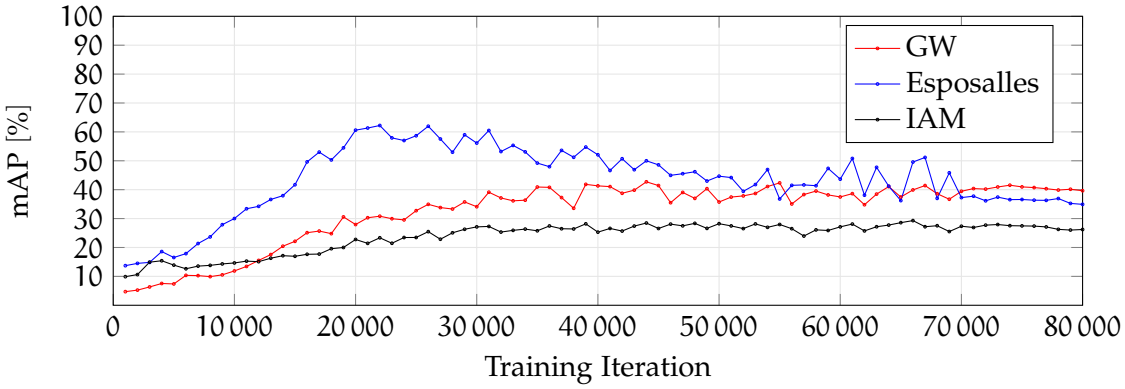


Figure 6.4.1: Mean Average Precision values for evaluating a PHOCNet trained on purely synthetic data on the three respective datasets. The results shown in this figure are obtained for the QbE scenario.

The mAP scores show a slight improvement of the network performance over the course of training. However, experiment reveals that the learned model faces limitations and it does not yield satisfactory results in word spotting. This is expected to have occurred because of the considerable difference in the visual appearance of the word images of the synthetic dataset as compared to those in each of the handwritten datasets. Hence, it is considered worth investigating whether the performance of the network improves when handwritten data is introduced to it. This leads to the next experiment involving finetuning of the network.

6.4.3 *Finetuning with Handwritten Data*

In this experiment, the PHOCNet trained on HW-SYNTH is finetuned on a subset of the training partition of a handwritten dataset. Subsets consisting of an absolute number of images are used to finetune the model for 40,000 iterations. In this work, subsets containing 100, 250, 500, and 1000 randomly drawn images have been used for each dataset. This provides an insight into how the number of images used to finetune the network affect its performance with an exponential granularity. The results of these experiments have been shown in Table 6.4.1. However, Fig.6.4.3 shows the results for the same experiments for the first 1000 iterations, since the largest improvement in mAP scores is made within the first 1000 iterations. Hence, the rapid improvement in performance of the network evaluated every 10 iterations can be well visualized in Fig. 6.4.3. Moreover, the bottom part of Table 6.4.1 shows the results for established

methods in word spotting in order to provide a benchmark for the performance of the method presented in this work.

The effect of the number of images in the subset used for finetuning the network on the performance of the network is visualized for QbE and QbS scenarios in Fig.6.4.2a and Fig.6.4.2b respectively. The graphs show a sharp increase in performance of the network when the subset size is increased from 100 to 250. However, the performance shows only a small improvement when the size of the subset is increased from 250 to 500, and an even smaller improvement from 500 to 1000.

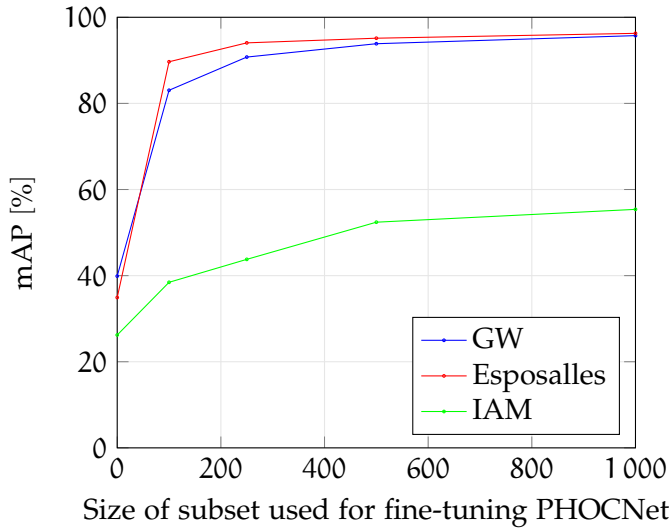
Note that the PHOC representations generated as target outputs from the labels of these subsets consist of the unigrams extracted from the words in the synthetic dataset. Hence, the PHOC representation for words in the handwritten documents may ignore characters which do not occur in the synthetic dataset. These subsets used to finetune the network are augmented in order to impose a regularization as well as to increase the number of images according to that specified in section 5.2.

This experiment proves that the PHOCNet pre-trained using synthetic data yields a competitive result in word spotting when it is finetuned with a small amount of real-world data. However, in order to confirm that the synthetic data plays an important role in achieving this performance, it must be shown that the PHOCNet trained using only the subsets used to finetune the pre-trained network would not achieve comparable results. Therefore, the significance of the synthetic data in this approach is investigated through the next experiment.

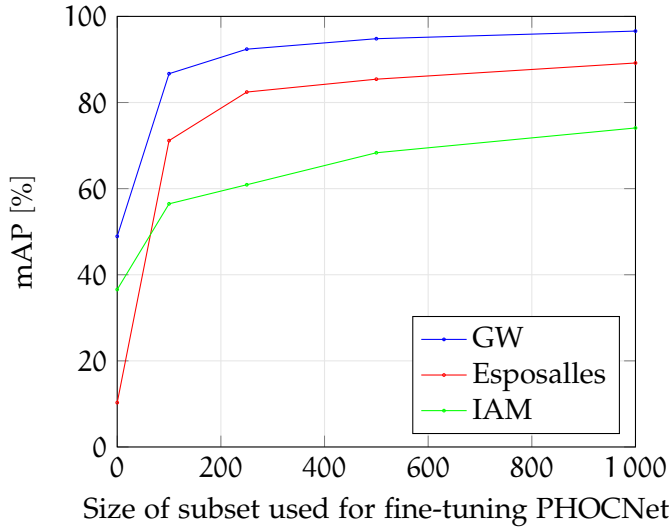
6.4.4 *Effect of Synthetic Data*

In order to find the effect that pre-training using the synthetic data has on the performance of the PHOCNet after finetuning, an experiment is conducted eliminating the use of the synthetic data. The three subsets containing 250 word images from each of the three handwritten datasets are used to train the PHOCNet from scratch using the set-up defined in section 6.3 for 80,000 iterations. It is expected that the conclusions drawn from this experiment with the subset of 250 word images also hold true for the subsets containing 100, 500, and 1000 word images. The results of this training are compared to those achieved using finetuning over pre-training with synthetic data in Fig.6.4.4.

It is seen that the results achieved by finetuning using the subset of 250 word images are considerably higher as compared to those obtained by training the PHOCNet using only that subset for all three datasets. This confirms that pre-training the PHOCNet with synthetic data makes a substantial contribution to the network performance.



(a) The effect of the size of the subset used for finetuning the network on its mAP score at 40,000 iterations for QbE scenario.



(b) The effect of the size of the subset used for finetuning the network on its mAP score at 40,000 iterations for QbS scenario.

Table 6.4.1: Results for experiments with different amounts of training images in mAP [%]

Method	Training Subset Size	GW		IAM		Esposalles	
		QbE	QbS	QbE	QbS	QbE	QbS
proposed	0	39.89	48.92	26.21	36.57	34.92	10.30
proposed	100	83.05	86.69	38.45	56.47	89.67	71.15
proposed	250	90.76	92.39	43.78	60.90	94.06	82.43
proposed	500	93.86	94.82	52.41	68.33	95.14	85.42
proposed	1000	95.74	96.59	55.39	74.09	96.27	89.18
AttributeSVM [AGFV14]	complete	93.04	91.29	55.73	73.72	–	–
PHOCNet [SF16]	complete	96.71	92.64	72.51	82.97	97.24	93.29
Deep Features [KDJ16]	complete	94.41	92.84	84.24	91.58	–	–
Triplet-CNN [WB16]	complete	98.00	93.69	81.58	89.49	–	–

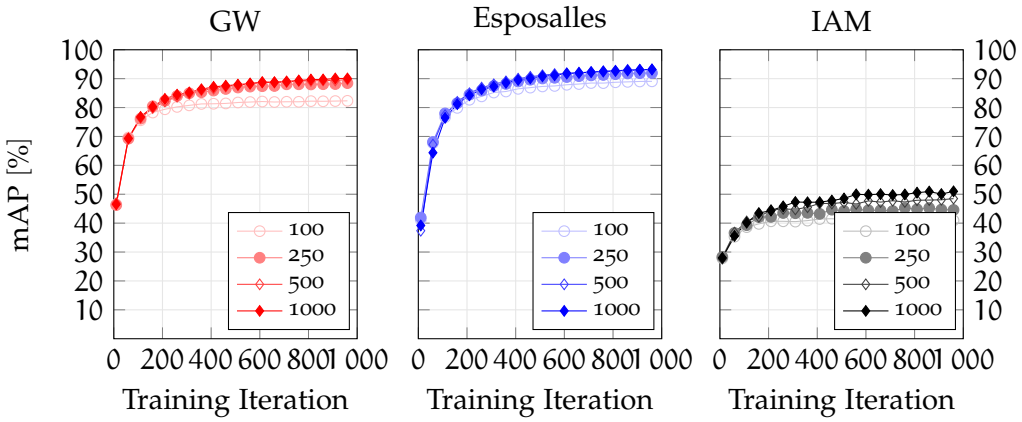


Figure 6.4.3: Mean Average Precision values for the finetuning experiments on the three handwritten datasets. The numbers in the legend indicate how many images were selected for finetuning from the training images. The results shown in this figure are obtained for the QbE scenario.

Another factor which is expected to contribute to the high performance of the finetuned PHOCNet is the augmentation of the subsets. The following experiment is conducted in order to to investigate the effects of augmentation.

6.4.5 *Effect of Augmentation*

The augmentation of the subsets used for finetuning the network requires computational effort. One may question whether the merit of augmentation outweighs this additional computational effort. Hence, an experiment is set up to show the how the PHOCNet would perform on being finetuned with a non-augmented subset. Non-augmented subsets containing 250 randomly sampled words are taken for each of the three handwritten datasets. These subsets are further augmented as specified in section 5.2 to give three augmented subsets. The PHOCNet pre-trained using synthetic data is finetuned for 40,000 iterations using each of the non-augmented and the augmented subsets. The mAP scores are obtained for each of the six episodes of finetuning the pre-train network. The results for all three datasets using an augmented and a non-augmented subset each are compared in Fig.6.4.5. It clearly shows an improvement in results when the subset used for finetuning is augmented. In fact the performance is shown to improve by up to approximately 5%. The performance is evaluated for the QbE scenario.

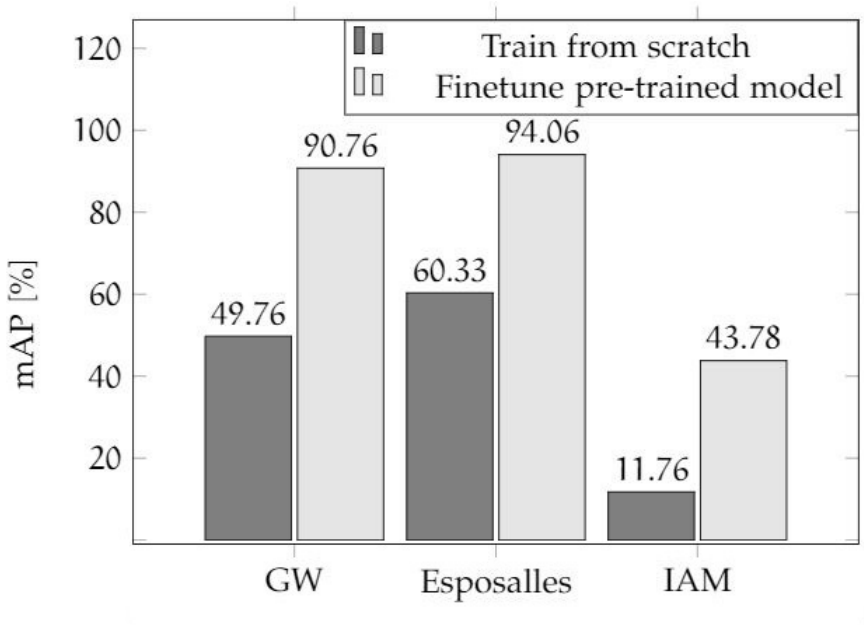


Figure 6.4.4: This graph compares the results obtained by training the PHOCNet from scratch for 80,000 iterations on a subset of 250 images of the handwritten dataset to that by finetuning the pre-trained PHOCNet for 40,000 iterations using the same subset for all 3 handwritten datasets. The figure clearly shows a higher performance when the subset is used to finetune the pre-trained model than when it is used to train the PHOCNet by itself. The results are obtained for the QbE scenario.

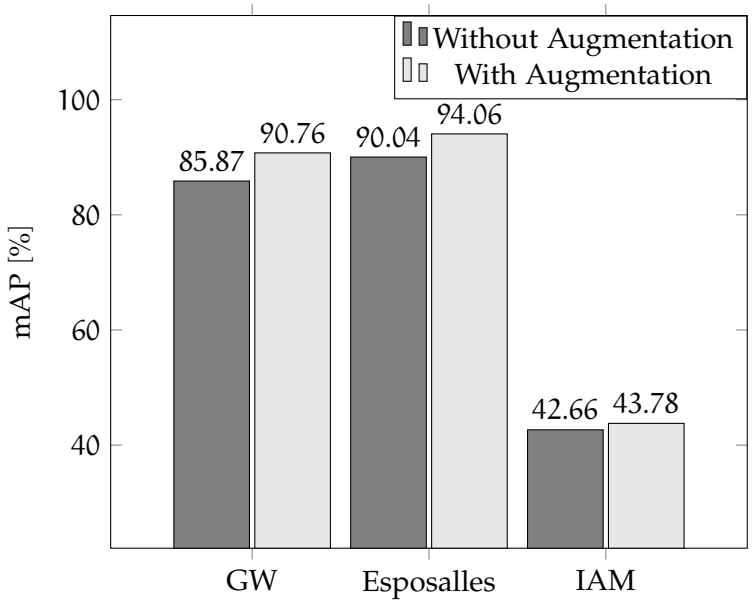


Figure 6.4.5: The figure shows results in terms of percentage mAP scores evaluated for the QbE scenario when the PHOCNet pre-trained with the synthetic dataset is finetuned using a subset of 250 randomly drawn images from the handwritten dataset. The results for finetuning using augmented and non-augmented subsets are compared.

It is expected that the conclusions drawn from this experiment with the subset of 250 word images also hold true for the subsets containing 100, 500, and 1000 word images. This shows that using augmentation consistently results in a better performance of the network. Hence, the subsets used for finetuning the PHOCNet are augmented in this work.

6.4.6 *Bray-Curtis Measure vs. Cosine Metric*

The PHOCNet used in this approach has been taken from [SF16]. Hence, the method presented here adopts most of the other parameters of the experimental set-up stated in the same related work which are expected to achieve an optimal performance for experiments using the PHOCNet. However, it can be expected that changing a few of these parameters may marginally improve the performance. One of the key differences is that the approach presented here uses the cosine metric as opposed to the Bray-Curtis measure used by [SF16]. The hypothesis is that the cosine metric is a preferred measure for this approach since Bray-Curtis measure is designed for the comparison of multivariate distributions or histograms, whereas the cosine metric is a commonly used measure for quantifying the similarity of orientation of two vectors such as the PHOC vectors in this case. Note that none of these are formal distance metrics since they do not satisfy the triangle inequality. Mathematically, for any distance metric, the triangle inequality states that the sum of distances computed from any arbitrary point A to an arbitrary point B and from the point A to an arbitrary point C must be greater than or equal to the distance computed between the points B and C. Nevertheless, the Bray-Curtis measure and the cosine metric provide a measure of dissimilarity between two data samples. An experiment is set up to empirically determine which metric yields the better results. The PHOCNet is trained on the training partition of all three handwritten datasets for 80,000 iterations and the performance is evaluated using the test partition of the corresponding dataset using both, the cosine metric as well as the Bray-Curtis measure, to obtain the mAP score. The cosine metric is shown to yield the same or a better mAP score in each experiment as compared to that obtained using the Bray-Curtis measure. This is visualized in Fig.6.4.6 for Query-by-Example and Fig.6.4.7 for Query-by-String. Hence, the proposed set-up uses the cosine metric to rank images during a query.

Additionally, the approach by [SF16] uses 50 commonly occurring bigrams in the PHOC representation which are not used in the PHOC generation for the approach presented here. Pre-experiments show that the overall set-up proposed in this work

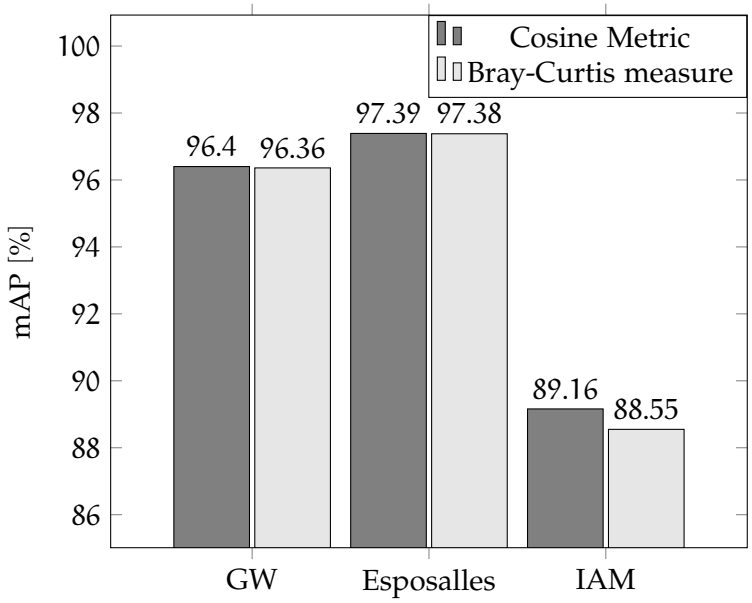


Figure 6.4.6: Cosine Metric vs Bray-Curtis measure. The results shown are obtained for Query-by-Example for all 3 handwritten datasets.

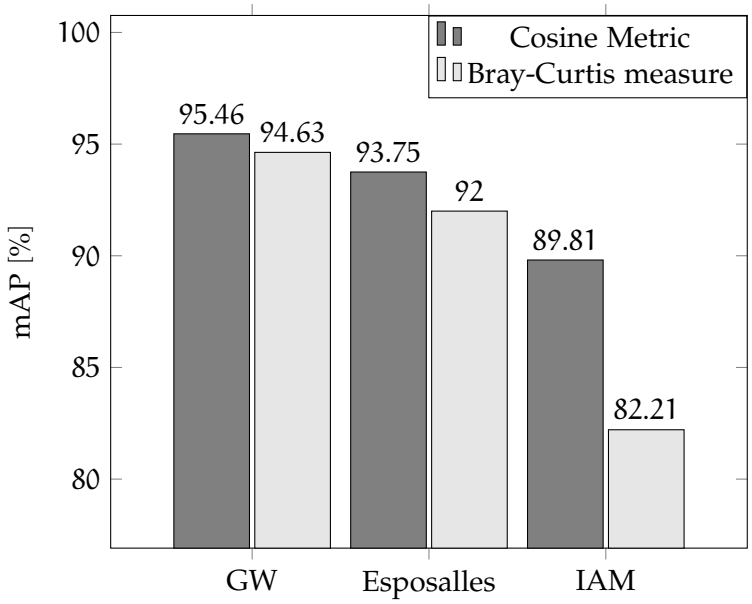


Figure 6.4.7: Cosine Metric vs Bray-Curtis measure. The results shown are obtained for Query-by-String for all 3 handwritten datasets.

shows a marginal improvement in performance as compared to that by [SF16] for a similar training protocol.

6.5 DISCUSSION

Several inferences can be drawn from the experiments. Firstly, from the results in Table 6.4.1, one can observe that training the PHOCNet only on the HW-SYNTH data set (i.e. synthetic data) does not allow the CNN to perform better than common unsupervised techniques such as [SF15] which uses local descriptors in a Bag-of-Features approach. However, after finetuning the model, its performance for the historic datasets surpasses that of the aforementioned unsupervised technique using merely 100 images from the handwritten dataset.

Secondly, the training set size is reduced by 86% for GW and almost 98% for IAM and still achieve results competitive to those of an AttributeSVM [AGFV14]. Moreover, when a subset of 14% of training images from IAM is used for finetuning, the results obtained are greatly improved. The mAP score is 65.81% for QbE and 86.11% for QbS.

The low amount of annotation effort that the method presented here demands can be further put into perspective. The ICFHR2016 Handwritten Keyword Spotting Competition [PZG+16] states that, for practical applications of keyword spotting, annotating 40 pages of handwritten documents is not a significantly tedious task, and that even annotating 154 pages is affordable. With the proposed set-up, this number drastically reduces further. For example, each page of the George Washington dataset consists of approximately 250 words on an average. This implies that, merely 4 pages must be annotated in order to obtain high performance.

Another significant feature of this method is the low training time. Although some time is required to train the model on the synthetic data, this phase of training does not need to be repeated for performing the task of word spotting in every handwritten dataset. In fact, once the model pre-trained on the synthetic data is made available, the training time to be considered for any practical application would be only that needed for finetuning the model on the desired handwritten dataset. The time required to finetune the pre-trained model for 1000 iterations ranges from approximately 7 to 11 minutes, depending on the dataset and the machine used for computation. From Figure 6.4.3, one can observe that the largest leap in the mAP scores is made within the first 1000 iterations. The next iterations bring about an improvement of approximately 2-3% in the mAP scores. Under ideal computing conditions, the training time can be considered to increase linearly with the number of iterations.

During the finetuning phase using a subset of 1000 images, 79-87% of the final mAP score (at 40,000 iterations) is reached within the first 200 iterations for all 3 datasets. That is, for a batch size of 10, the network needs to merely observe 2000 images in order to achieve 79-87% of the final mAP score. From this, it can be interpreted that the network is capable of learning the visual appearance of the word images in a dataset as a whole as opposed to individual words, since the network performance makes a leap without having observed more than a fraction of the available training words especially for the IAM and Esposalles datasets. This may explain the lower performance for the IAM as compared to George Washington and Esposalles datasets. IAM consists of more visual diversity in writing styles as compared to those in the other handwritten datasets, hence making it more difficult to learn the model.

7 CONCLUSION

One of the primary challenges of word spotting in handwritten documents is that it is difficult for a word spotting system to adapt to the diversity in visual appearance of handwritten words. Supervised approaches are best equipped to overcome this challenge. In a supervised approach such as training a CNN, the system requires training data which spans this diversity. This means that the amount of handwritten training data must be large enough to take into account the variation in the visual appearance of the text. However, annotating a large amount of training data is a time consuming and tedious process. Hence, it is desirable to explore techniques which allow a system to perform well despite a decrease in the amount of data used for training it. In this work, transfer learning is introduced as an approach to reduce the amount of handwritten data used to train the PHOCNet for word spotting.

The PHOCNet, is trained to map word images on to their PHOC vector representation. The PHOC representation is extracted from the text or string representation of a word and it encodes the presence of a character within a spatial region of the word. The PHOC vector space is thus a common subspace on to which an image as well as the text representation of a word can be mapped. Hence, this approach enables word spotting in the QbE as well as the QbS scenario. The network is first trained using synthetically generated data, namely the HW-SYNTH dataset. The contents of this dataset are designed to emulate the variation in handwriting. The performance of the PHOCNet thus trained is evaluated using handwritten data. That is, knowledge from the source domain of the synthetic data is transferred to the target domain of handwritten data. However, one can observe that the PHOCNet faces limitations because of the difference in the visual appearance of the contents of the synthetic and handwritten datasets. Hence, the visual characteristics of handwriting are introduced to the PHOCNet by finetuning the network pre-trained over the synthetic data using handwritten data. Finetuning is carried out using augmented subsets of 100, 250, 500, and 1000 images randomly sampled from the training partitions of each of the three handwritten datasets. This shows how the performance is affected by the number of images used for finetuning with an exponential granularity. As expected, the performance of the network improves with the increase in size of the subset used for finetuning. In fact, the results obtained using 1000 images for finetuning surpass that obtained by training AttributeSVMs [AGFV14]. Although this is achieved by finetun-

ing over the largest subset of handwritten data, it is still a fraction of the training data used by established supervised approaches in word spotting. Consequently, the manual effort for annotating the training data is reduced to a large extent. Additionally, once the PHOCNet model pre-trained on the synthetic data is made available, the effective training time for practical applications is only that required to finetune the network using a subset of the desired handwritten dataset. Thus, the training time is also reduced to a fraction of that required for other supervised approaches such as [AGFV14] and [SF16].

To sum up, the approaches of transfer learning and finetuning allow the PHOCNet to retain a high performance in word spotting in handwritten documents while greatly reducing the amount of handwritten data required to train the network. As a result, this approach also reduces the manual effort and time required to prepare the training data and to train the network.

BIBLIOGRAPHY

[AF15] Ahmad, Irfan ; Fink, Gernot A.: Training an Arabic handwriting recognizer without a handwritten training data set. In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, aug 2015. – ISBN 978–1–4799–1805–8, S. 476–480

[AFM14] Ahmad, Irfan ; Fink, Gernot A. ; Mahmoud, Sabri A.: Improvements in Sub-character HMM Model Based Arabic Text Recognition. In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. Crete : IEEE, sep 2014. – ISBN 978–1–4799–4334–0, S. 537–542

[AGFV14] Almazán, Jon ; Gordo, Albert ; Fornés, Alicia ; Valveny, Ernest: Word Spotting and Recognition with Embedded Attributes. In: *Pattern Analysis and Machine Intelligence* 36 (2014), Nr. 12, S. 2552–2566

[ARFM13] Ahmad, Irfan ; Rothacker, Leonard ; Fink, Gernot A. ; Mahmoud, Sabri A.: Novel Sub-character HMM Models for Arabic Text Recognition. In: *2013 12th International Conference on Document Analysis and Recognition*. Washington DC : IEEE, aug 2013. – ISBN 978–0–7695–4999–6, S. 658–662

[ARTL13] Aldavert, David ; Rusinol, Marcal ; Toledo, Ricardo ; Llados, Josep: Integrating Visual and Textual Cues for Query-by-String Word Spotting. In: *International Conference on Document Analysis and Recognition*, 2013. – ISBN 978–0–7695–4999–6, S. 511–515

[BJTM16] Balntas, Vassileios ; Johns, Edward ; Tang, Lilian ; Mikolajczyk, Krystian: PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. In: *arXiv* abs/1601.0 (2016)

[BMP06] Blitzer, John ; McDonald, Ryan ; Pereira, Fernando: Domain adaptation with structural correspondence learning. In: *Proceedings of the 2006 conference on empirical methods in natural language processing* Association for Computational Linguistics, 2006, S. 120–128

[DR08] De Raedt, Luc: *Logical and relational learning*. Springer Science & Business Media, 2008

[DYXY08] Dai, Wenyuan ; Yang, Qiang ; Xue, Gui-Rong ; Yu, Yong: Self-taught clustering. In: *Proceedings of the 25th international conference on Machine learning* ACM, 2008, S. 200–207

[GB10] Glorot, Xavier ; Bengio, Yoshua: Understanding the Difficulty of Training Deep Feedforward Neural Networks. In: *AISTATS* 9 (2010), S. 249–256

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[HZRS14] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: *European Conference on Computer Vision* (2014), S. 346–361

[KDJ16] Krishnan, Praveen ; Dutta, Kartik ; Jawahar, C.V.: Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text. In: *International Conference on Frontiers in Handwriting Recognition*, 2016, S. 289–294

[KJ16] Krishnan, Praveen ; Jawahar, C.V.: Matching Handwritten Document Images. In: *European Conference on Computer Vision*, 2016

[KJSR16] Kim, Yoon ; Jernite, Yacine ; Sontag, David ; Rush, Alexander M.: Character-aware neural language models. In: *Thirtieth AAAI Conference on Artificial Intelligence*, 2016

[MHR96] Manmatha, R ; Han, Chengfeng ; Riseman, E.M.: Word Spotting: A New Approach to Indexing Handwriting. In: *Computer Vision and Pattern Recognition* (1996), S. 1–29. http://dx.doi.org/10.1109/CVPR.1996.517139. – DOI 10.1109/CVPR.1996.517139. – ISBN 0–8186–7258–7

[MHRC96] Manmatha, R ; Han, Chengfeng ; Riseman, Edward M. ; Croft, W B.: Indexing handwriting using word matching. In: *Proceedings of the first ACM international conference on Digital libraries* ACM, 1996, S. 151–159

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

[MP69] Minsky, Marvin ; Papert, Seymour: Perceptrons. (1969)

[MP01] Märgner, V. ; Pechwitz, M.: Synthetic Data for Arabic OCR System Development. In: *International Conference on Document Analysis and Recognition*, 2001. – ISBN 0769512631, S. 1159–1163

[Mun14] Munoz, Andres: Machine Learning and Optimization. In: *URL: https://www. cims. nyu. edu/˜ munoz/files/ml_optimization. pdf [accessed 2016-03-02][WebCite Cache ID 6fiLfZvnG]* (2014)

[Nie15] Nielsen, Michael A.: Neural networks and deep learning. In: *URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.11. 2014)* (2015)

[PMB13] Pascanu, Razvan ; Mikolov, Tomas ; Bengio, Yoshua: On the Difficulty of Training Recurrent Neural Networks. In: *International Conference on Machine Learning*, 2013. – ISSN 1045–9227, 1310–1318

[Por80] Porter, Martin F.: An algorithm for suffix stripping. In: *Program* 14 (1980), Nr. 3, S. 130–137

[PY10] Pan, Sinno J. ; Yang, Qiang: A survey on transfer learning. In: *IEEE Transactions on knowledge and data engineering* 22 (2010), Nr. 10, S. 1345–1359

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

[RATL15] Rusiñol, Marçal ; Aldavert, David ; Toledo, Ricardo ; Lladós, Josep: Efficient segmentation-free keyword spotting in historical document collections. In: *Pattern Recognition* 48 (2015), Nr. 2, S. 545–555. http://dx.doi.org/10.1016/j.patcog.2014.08.021. – DOI 10.1016/j.patcog.2014.08.021. – ISSN 00313203

[RF15] Rothacker, Leonard ; Fink, Gernot A.: Segmentation-free Query-by-String Word Spotting with Bag-of-Features HMMs. In: *International Conference on Document Analysis and Recognition*, 2015. – ISBN 978–1–4799–1805–8, S. 661–665

[RFS+13] Romero, Verónica ; Fornés, Alicia ; Serrano, Nicolás ; Sánchez, Joan A. ; Toselli, Alejandro H. ; Frinken, Volkmar ; Vidal, Enrique ; Lladós, Josep: The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition. In: *Pattern Recognition* 46 (2013), Nr. 6, S.

1658–1669. http://dx.doi.org/10.1016/j.patcog.2012.11.024. – DOI 10.1016/j.patcog.2012.11.024. – ISSN 00313203

[RM07] Rath, Tony M. ; Manmatha, R.: Word Spotting for Historical Documents. In: *International Journal on Document Analysis and Recognition* 9 (2007), S. 139–152. http://dx.doi.org/10.1007/s10032-006-0027-8. – DOI 10.1007/s10032–006–0027–8. – ISBN 0–7695–1960–1

[RMR06] Rothfeder, Jamie ; Manmatha, R ; Rath, Toni M.: Aligning transcripts to automatically segmented handwritten manuscripts. In: *International Workshop on Document Analysis Systems* Springer, 2006, S. 84–95

[Ros58] Rosenblatt, Frank: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological review* 65 (1958), Nr. 6, S. 386

[RRF13] Rothacker, Leonard ; Rusinol, Marcal ; Fink, Gernot A.: Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents. In: *International Conference on Document Analysis and Recognition*, 2013. – ISBN 978–0–7695–4999–6, S. 1305–1309

[RSP09] Rodríguez-Serrano, José A. ; Perronnin, Florent: Handwritten word-spotting using hidden Markov models and universal vocabularies. In: *Pattern Recognition* 42 (2009), Nr. 9, S. 2106–2116. http://dx.doi.org/10.1016/j.patcog.2009.02.005. – DOI 10.1016/j.patcog.2009.02.005. – ISSN 00313203

[RVF12] Rothacker, Leonard ; Vajda, Szilard ; Fink, Gernot A.: Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script. In: *International Conference on Frontiers in Handwriting Recognition*, 2012

[SF15] Sudholt, Sebastian ; Fink, Gernot A.: A Modified Isomap Approach to Manifold Learning in Word Spotting. In: *German Conference on Pattern Recognition*, 2015, S. 529–539

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

[SHK+14] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout : A Simple Way to Prevent Neural

Networks from Overfitting. In: *Journal of Machine Learning Research (JMLR)* 15 (2014), S. 1929–1958. – ISBN 1532–4435

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *arXiv* (2014), S. 1–13

[VB03] Varga, Tamás ; Bunke, Horst: Generation of Synthetic Training Data for an HMM-based Handwriting Recognition System. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR* Bd. 2003-Janua, 2003. – ISBN 0769519601, S. 618–622

[VB04] Varga, Tamás ; Bunke, Horst: Comparing Natural and Synthetic Training Data for Off-line Cursive Handwriting Recognition. In: *International Workshop on Frontiers in Handwriting Recognition*, 2004. – ISBN 0769521878, S. 221–225

[WB16] Wilkinson, Tomas ; Brun, Anders: Semantic and Verbatim Word Spotting using Deep Neural Networks. In: *International Conference on Frontiers in Handwriting Recognition*, 2016, S. 307–312

[WH86] Williams, DRGHR ; Hinton, GE: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Nr. 6088, S. 533–538

[WSZ08] Wang, Zheng ; Song, Yangqiu ; Zhang, Changshui: Transferred dimensionality reduction. In: *Machine learning and knowledge discovery in databases* (2008), S. 550–565

[YEG+02] Young, Steve ; Evermann, Gunnar ; Gales, Mark ; Hain, Thomas ; Kershaw, Dan ; Liu, Xunying ; Moore, Gareth ; Odell, Julian ; Ollason, Dave ; Povey, Dan u. a.: The HTK book. In: *Cambridge university engineering department* 3 (2002), S. 175