Intelligent Document Recognition with Machine Learning

Stephen Charlton

Submitted in partial fulfilment of
the requirements of Edinburgh Napier University
for the Degree of
MSC Computing

In collaboration with Dialexy

School of Computing

August 2018

MSc dissertation check list

Milestones	Date of completion	Target deadline
Proposal	30/05/2018	Week 3
Initial report	16/07/2018	Week 7
Full draft of the dissertation	05/08/2018	2 weeks before final deadline

Learning outcome	The markers will assess	Pages[1]	Hours spent
Learning outcome 1 Conduct a literature search using an appropriate range of information sources and produce a critical review of the findings.	* Range of materials; list of references * The literature review/exposition/background information chapter	14-34	130 hours (for example, 200 hours)
Learning outcome 2 Demonstrate professional competence by sound project management and (a) by applying appropriate theoretical and practical computing concepts and techniques to a non-trivial problem, <u>or</u> (b) by undertaking an approved project of equivalent standard.	* Evidence of project management (Gantt chart, diary, etc.) * Depending on the topic: chapters on design, implementation, methods, experiments, results, etc.	35-54	300 hours (for example, 200 hours)
Learning outcome 3 Show a capacity for self-appraisal by analysing the strengths and weakness of the project outcomes with reference to the initial objectives, and to the work of others.	* Chapter on evaluation (assessing your outcomes against the project aims and objectives) * Discussion of your project's output compared to the work of others.	55-63	60 hours (for example, 100 hours)
Learning outcome 4 Provide evidence of the meeting learning outcomes 1-3 in the form of a dissertation which complies with the requirements of the School of Computing both in style and content.	* Is the dissertation well-written (academic writing style, grammatical), spell-checked, free of typos, neatly formatted. * Does the dissertation contain all relevant chapters, appendices, title and contents pages, etc. * Style and content of the dissertation.		24 (for example, 80 hours)
Learning outcome 5 Defend the work orally at a viva voce examination.	* Performance * Confirm authorship		1 hour

Have you previously uploaded your dissertation to Turnitin?	Yes
Has your supervisor seen a full draft of the dissertation before submission?	Yes/
Has your supervisor said that you are ready to submit the dissertation?	Yes

[1] Please note the page numbers where evidence of meeting the learning outcome can be found in your dissertation.

Authorship Declaration

I, Stephen Charlton, confirm that this dissertation and the work presented in it are my own achievement.

Where I have consulted the published work of others this is always clearly attributed;

Where I have quoted from the work of others the source is always given. With the exception of such quotations this dissertation is entirely my own work;

I have acknowledged all main sources of help;

If my research follows on from previous work or is part of a larger collaborative research project I have made clear exactly what was done by others and what I have contributed myself;

I have read and understand the penalties associated with Academic Misconduct.

I also confirm that I have obtained **informed consent** from all people I have involved in the work in this dissertation following the School's ethical guidelines

Signed: Stephen Charlton

Date: 11/08/2018

Matriculation no: 40331195

Data Protection Declaration

Under the 1998 Data Protection Act, The University cannot disclose your grade to an unauthorised person. However, other students benefit from studying dissertations that have their grades attached.

Please write your name below *one* of the options below to state your preference.

The University may make this dissertation, with indicative grade, available to others.

Stephen Charlton

The University may make this dissertation available to others, but the grade may not be disclosed.

The University may not make this dissertation available to others.

Abstract

This project will investigate methods of intelligent document recognition which can automatically classify documents as one of multiple predefined classes. The work is in collaboration with Edinburgh company Dialexy who use automatic document processing as part of their translation service. It will attempt to find a model which can successfully be implemented by the company on their translation platform.

Models using two different machine learning techniques are researched and tested to assess their performance and suitability for this task. One technique involves classifying documents through their image using a Convolutional Neural Network and the other classifies them through their text using Natural Language Processing Techniques.

The outcomes demonstrate that both techniques have the potential to perform well on Dialexy’s platform. It concludes with a suggestion for a Convolutional Neural Network model with the best potential to be successful based on the experiment results.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank Dimitra Gkatzia for her advice and support throughout this project and Neil Urquhart for his feedback and encouragement.

I would also like to thank Monica Camino Moreno and Blaise Potard at Dialexy for all of their help and for giving me the opportunity to work with them.

1 Introduction

1.1 The Context

The idea behind the project derives from a work placement at Dialexy who specialise in "the development of software solutions to improve and automate the certified translation process for certified translators and their clients" (Dialexy, n.d.). The company uses AI techniques in order to try to automate as much of the translation process as possible, making it both easier and quicker for a translator to complete a piece of work for a client.

This project falls within the field of 'Intelligent Document Recognition', where on receiving an image of a document; a model can accurately predict which of a set of previously seen templates it is most similar to. The document will be uploaded by a client in pdf format and due to the nature of the company; it could be in any number of languages.

If this can be successfully done, then the models would facilitate the translation process by providing the translator with the correct document structure to translate onto within the company's online translation interface.

Currently the company does not have much data accumulated but as the company grows, more users will upload their documents to the company's website for translation. These documents can be used to further optimize, train and develop the document recognition models. As many documents are similar, with enough examples in the database, the translation process could be facilitated by auto-predicting many sentence's translations. This would be done by comparing what documents and the words they contain have been translated into in the past. This in turn could pave the way for automatic translation in the future.

1.2 Aims and Objectives

In collaboration with the company Dialexy, the project is to use AI techniques to develop models which will be usable on the company's translation platform. One of the important features of the platform will be its ability to accurately and automatically classify a document's type, from a set of stored document templates.

The platform will use two main methods to attempt to classify the document; the first will be through its image using a pre-trained Convolutional Neural Network (Lecun, Bottou, Bengio & Haffner, 1998, p. 2284). The second will attempt to match the document through its text. The text will be read using an OCR tool that is fairly accurate although some documents may be handwritten which could cause problems. As some documents have similar textual content but very different layouts, the image recognition will have more emphasis throughout this project. Returning a project layout to the translator that matches the layout of the uploaded document is an important part of Dialexy's platform's selling point.

The project will predominantly involve adapting and training a Convolutional Neural Network (CNN) for the document image recognition. It will look into different network architectures, pre-processing techniques and training methods to try and discover how the platform's performance can be optimised. As Dialexy does not have a large amount of training data, some experiments will be carried out on the Big Tobacco dataset (Harley, Ufkes & Derpanis, 2015). It contains very similar document types to the ones the platform will be exposed to daily. The idea behind this is that if the models can perform well on this dataset, then it will also perform well in its business implementation when the company has more data.

During the training of the CNN, a potential way to get around the lack of data could be to create multiple 'artificial' copies of each document to be used in training. Therefore this project will look into whether training neural networks with artificial data is a serious option for businesses and if so, which ways they can go about doing it. It is important that the CNN model does not suffer from the overfitting usually caused by a small training sample.

There are a few other requirements that the CNN implemented on the company's platform must meet. Firstly, due to the nature of the business, the company's live system will occasionally be exposed to previously unseen document types. Therefore it would beneficial to the company if a way to accurately identify these unseen document classes could be found and implemented on the platform. Secondly, the customer’s uploaded documents will not always be perfect scans. It is very likely some will be of poor quality or upside down. Therefore it is important for the model to be trained to still recognise these documents. Lastly, speed will be an important aspect of the prediction process so that the translator is returned the document type promptly. It would therefore be beneficial to use as fewer models as possible while still being accurate.

The company is looking to implement a new model at the end of August, which can identify 10 main document classes and identify documents that do not belong to either of the main classes. This project will consider the many aspects and requirements of Dialexy’s current business situation and attempt to find the best suited model or model combination to be used on the company’s platform for this task.

1.3 Report Structure

This dissertation will be split into 6 Chapters which are as follows:

1. **The Introduction** introduces the reasons behind the project and outlines its objectives.
2. **The Literature Review** examines the literature within Image Processing and Text Classification. It also explains the terminologies and models used within this field of study.
3. **The Design and Implementation** chapter explains the datasets and models that were used in the project and how the experiments were set up.
4. **The Testing and Results** chapter outlines the different experiments that were performed and presents their results.
5. **Analysis and Discussion** will discuss each of the experiment’s results to see how well each model has performed.
6. **The Conclusion** will summarise the outcome of the project and will propose some further work that could be carried out in this area of study.

2 Literature Review

This chapter will present the background to Intelligent Document Recognition and Machine Learning (Chapter 2.1). It will discuss two different commonly used approaches, Text Classification Techniques (Chapter 2.2) and Convolutional Neural Networks (Chapter 2.3). It will then investigate Open Set Recognition (Chapter 2.4) and Augmented and Synthetic Data (Chapter 2.5) before concluding the findings in the Literature Review Conclusion (Chapter 2.6).

2.1 Intelligent Document Recognition

"Intelligent document recognition interprets content and patterns on documents to automatically classify paper and electronic documents into different document types" (Mancini, 2016, para. 1).

The ability to classify documents automatically can have a variety of different uses. They can be used within digital libraries to auto assign documents into categories (Cesarini, Lastri, Marinai & Soda, 2001, p. 1131) and are used within automated form processing systems in businesses (SoftWorks AI, n.d).

In this dissertation, the term document will refer to Chen & Blostein's (2004, p. 2), definition of "a single-page typeset document image. The document image may be produced from a scanner, a fax machine or by converting an electronic document into an image format".

This branch of study is part of the broader field of Machine Learning, which can be defined as "the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world" (Copeland, 2016, para. 10).

Machine learning can generally be broken up into further subcategories such as supervised learning, unsupervised learning and reinforcement learning (Ray, 2017, para. 3-5). The methods discussed in this piece of work will all fall within the machine learning field of supervised learning.

Supervised learning systems use training data that has both an input and a known desired output. An algorithm is used to ‘learn’ how to map the inputs into the desired outputs by adjusting its parameters. The training data is passed through the algorithm multiple times which is known as the training process (Ray, 2017, para. 4). If successful, the algorithm with its learned parameters would be able to give the desired output given a previously unseen input.

There are many examples of research into the Intelligent Document Recognition field in literature and the research can be split into two categories. One type of model relies on Text Classification to identify the document type through its content. This would involve extracting the image’s text using Optical Character Recognition tools and applying Natural Language Processing techniques to the content. The other type of model identifies the document type using image processing techniques to analyse its structure and to detect patterns and features (Afzal, Capobianco, Malik, Marinai, Breuel, Dengel & Liwicki, 2015, p. 1111).

Which of these two approaches is most suitable may depend on the type of documents; those which are textually very similar may be more distinguishable using an image processing technique and those with similar layouts e.g. different types of letter may perform better with a text classification.

2.2 Text Classification of Documents

Text classification aims to assign text documents into one or more predefined classes based on their word contents (Bai & Nie, 2004, p. 1). It is part of a wider field of study within Artificial Intelligence called Natural Language Processing (NLP), which explores how computers can understand, interpret and manipulate language (Sas, n.d., para. 1). NLP models have been successfully implemented in many areas of research such as sentiment analysis and information retrieval (Butnaru & Ionescu, 2017, p. 1).

2.2.1 Pre-Processing

Before an NLP model can be ran, the raw training text data is pre-processed so that it is easier for the model to learn from the data. Pre-processing can include any combination of the following depending on what may be more suitable to the study:

1. **Tokenization** “is the process of breaking a stream of text into words, phrases, symbols, or other meaningful elements called tokens” (Kannan & Gurusamy, 2014, p.

2). Splitting the text or ‘corpus’, into smaller pieces such as individual words allows for further processing tasks such as the creation of NGrams or the Bag of Words (Mayo, n.d., para. 5).

2. **NGrams** refers to the breaking up of the corpus into continuous sequences of n words, which are then used in the model. This could be unigrams (1 word), bigrams (2 words), trigrams (3 words) or any other desired number (Mayo, n.d., para. 16).
3. **Bag of Words** (BoW) is a representation of the corpus as a list of its words and word frequency counts for use in modelling. It discards any knowledge relating to the order the words appear in or any grammar (Brownlee, 2017, para. 11).
4. **‘Stop Words’** refer to frequently occurring words in a corpus such as “this” or “and”, which are often considered to not contribute to the meaning of the text. They are often removed so they do not form part of the model’s input (Kannan & Gurusamy, 2014, p. 3).
5. **Stemming** is the process of converting multiple variants of the same word into one representation, for example “presentation” and “presented” could be reduced into “present”, so they are considered the same word in the model (Kannan & Gurusamy, 2014, p. 4).

2.2.2 Text Classification Models

A large number of statistical classification and ML techniques have been used in Text Classification models in literature (Dumais, Platt, Heckerman & Sahami, 1998, p. 1), a few examples would be Decision Trees, Nearest Neighbour, Logistic Regression and Neural Networks (Li & Jain, 1998, p. 2). This work will examine two, The Naive Bayes Classifier and the Support Vector Machine (SVM).

The Multinomial Naive Bayes (NB) algorithm is one of the most commonly used classifiers which is “widely used in text classification for its simplicity and efficiency” (Bai & Nie, 2004, p. 1). Although it is simple and a probabilistic classifier rather than an optimization algorithm, it can be surprisingly powerful (Brownlee, 2016, para. 1).

NB starts off with the word frequencies for each class or BOW discussed above. To classify a sentence, a probability that it belongs to each of the classes is calculated and the one with the highest probability becomes the classification. For each word in the formula in Figure 1 is applied.

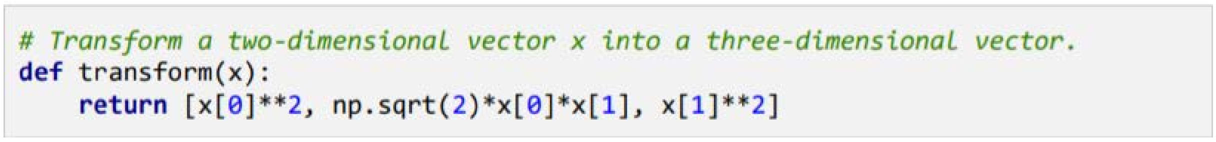
$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c)\right) + |V|}$$

Figure 1 The formula to calculate the Bayesian probability that a word belongs in a particular class. The count(wi,c) represents the word's appearance in that particular class. The (∑count(w, c)) are the total words in the corresponding class and the V is the total words in the entire corpus. Laplace smoothing has been added to the formula, so 1 is added to the numerator (Stanford University, n.d., p. 31).

To calculate the probability that a sentence pertains to a class, the number of times each word appears in that class is divided by the product of the total words in that class (including duplicates) and the number of unique words in the entire training set. The result of this calculation for every word in a sentence is multiplied together to get the overall classification score (Stecanella, 2017, para. 9-14). As some words will not be found in the training data, this would result in that word's calculation equalling 0. As this is going to be multiplied with every other word's score, then it is going to equal 0 overall. To avoid this 1 is always added to the word appearance, so that even words that do not appear in the training set have a numerator of 1 in the equation. This is known as 'Laplace Smoothing' which is one of many smoothing techniques which can be implemented in NLP (Stecanella, 2017, para. 20).

Support Vector Machine (SVM) is a supervised ML algorithm which can perform binary classifications. It does this by creating a feature space containing n dimensions where n is the number of features that exist in the training data input. The training features are then plotted as points inside the feature space and the model attempts to find a hyperplane that will divide the two classes, with all the points from each class on either side. SVM will attempt to place the hyperplane as far away as possible from the data points in each category (Stecanella, 2017, para. 5-6). The distance from the hyperplane to the data points which is to be maximised is calculated by computing the dot product between the data points and the hyperplane (Kowalczyk, 2014, para. 8). If the data is not linearly separable in the input dimensions, then it is necessary for the data to be projected into a higher dimensional space in order to be separated. This is done by applying a function to all the inputs. This could be done in the pre-processing stage, as per below in Figure 2, but the ‘Kernel’ features of SVMs mean that this is not necessary (Kowalczyk, 2017, p. 72-75).

Figure 2 An example of a 2D input vector being converted into 3D (Kowalczyk, 2017, p.72).



```
# Transform a two-dimensional vector x into a three-dimensional vector.
def transform(x):
    return [x[0]**2, np.sqrt(2)*x[0]*x[1], x[1]**2]
```

A Kernel is a function which returns the result of a dot product between 2 vectors performed in another dimensional space, without the need to transform the vector’s dimensions. This allows for the calculation of the best fitting hyperplane amongst the data points in a higher dimension, where the data could now be linearly separable. Various types of kernel exist, each one projecting the data points into a different space. Different kernels can perform very differently on the same dataset, so some domain knowledge is required to get the best performance out of a SVM (Kowalczyk, 2017, p. 75-77).

SVMs take an input of number vectors, so for text classification, the features must be converted into numbers. One common and simple way of doing this is by using word frequencies. Each word in the bag of word becomes a feature and the value of that feature is the words appearance. These features converted into vectors of word counts then become the inputs (Stecanella, 2017, para.16-17).

SVMs can only perform binary classifications, so in situations where there are more than two classes, they will need to follow either the 1 vs Rest or 1 vs All approach. In a 1 Vs Rest approach, a separate classifier is constructed for each class. So for n classes, there will be n classifiers. Each one performs a binary classification between one class and the rest of the class (n-1 classes). During testing, the classification will be the class in which the data point is furthest from the hyperplane in that classes binary classifier (Pal, 2008, p. 5). In the 1 Vs 1 approach, a binary classifier is constructed for each possible combination of 2 classes. During testing, the input is classified in each model and the final classification is the class with the largest number of classifications across all the models. In the case of a tie, one of the tied classes is usually selected randomly (Pal, 2008, p. 6).

2.2.3 Text Classification Models in Literature

In literature, both SVM and NB are commonly used for text classification in a number of different domains and have produced strong results. For example in Ko, Park & Seo (2004) four different classifiers were compared for accuracy when used on two large document datasets of 20,000 English and 10,331 Korean documents with 20 and 15 different classes respectively. They also assigned different weights to each feature in the document based on whether they should be more important or not. For example titles were given bigger weightings. The F score results found that SVM performed best in both datasets followed by Naïve Bayes. They also achieved best results for the datasets to date for each classification model on the datasets, indicating that some areas of a text can indeed be more important than others in the classification process (Ko et al, 2004, p. 4).

Li & Jain (1998) compared BN against 3 other classification algorithms, Nearest Neighbour, Decision Tree and a Subspace Model. They were tested on a news dataset of 814 documents belonging to 7 different classes, which is quite a small dataset. The training data was converted into unigrams and stop words were removed. Unlike Ko et al (2004), the text structure was not taken into account. The test data was split into 2 parts and each of the results were recorded separately. Naive Bayes was found to perform best in the first experiment and second best in the second experiment, losing out to the sub space model (Li & Jain, 1998, p. 6). The study concludes that NB works well within the news text domain as it outperformed 2 of the other 4 classifiers. It also indicates that NB can still be effective even with a smaller dataset (just over 100 items per class) (Li & Jain, 1998, p.7).

In Dumais et al (1998), SVM and BN were tested against 3 other algorithms on a large dataset of 12,902 texts spread across 118 classes. In the pre-processing stage, only the words with the highest counts in each category were kept. This number was 300 for SVM and 50 for NB. The experiment’s performance was measured by averaging out precision and recall. In the results SVM performed best with 92% and was also found to be faster to train than some of the other models such as decision trees. BN did not perform as well in this experiment scoring only 81.5% and being outperformed by all but one of the other models.

2.2.4 Difficulties Experienced With Text Classification Models

Although text classification has been successfully applied in the above mentioned studies, it does have some limitations. One of these would be that it is very difficult to capture the correct meaning of language from its key words. For example, the same word can have multiple meanings, making it difficult for the model to ‘understand’ a word’s meaning in the current context. In reverse, the same meaning can be expressed in multiple words, which makes different sentences have the same meaning. Spelling errors can also mean that two words which should be the same are classified differently (Li & Jain, 1998, p. 1). This is likely to be a problem when converting document images into text using OCR tools.

BOW methods may also discard some of the order and structure of the corpus. For example, in a document a word in a title or heading could perhaps be more significant than a word in small writing at the bottom as was argued in Ko et al (2004, p. 4). However in the BOW they are usually each given equal importance. The spatial arrangement of the words across a document is lost, although some word order is still preserved through larger NGrams (Brownlee, 2017, para. 13).

Datasets with smaller corpus, such as short sentences are harder to model as there is little information to be learned in them. This may apply to document classification where many documents contain very few words (Brownlee, 2017, para. 13).

2.3 Image Processing

2.3.1 Image Processing Techniques

Many types of documents have a distinct visual layout and structure which can be used to identify them (Harley et al, 2015, p. 1). This could be for instance, a photo in a passport scan. Several studies over the years have attempted to classify documents by detecting features within these layouts.

Earlier literature such as those surveyed in Chen & Blostein, (2007) focused on creating hand crafted features and searching for them within the document. These were classed by Chen & Blostein (2007, p. 6) as either ‘structural features’ such as text columns or ‘image features’ such as photographs and pictures. This was expensive to do as each domain of document classification had to have their own features created. These were originally hand crafted but later on extracted automatically from labelled samples. They were typically formatted as a binary image. The features were searched for in the documents allowing for certain amounts of noise and categorized into a document type using a classification algorithm. Types of algorithms were varied but included decision trees, string matching and K Nearest Neighbour (Chen & Blostein, 2007, p. 7).

These implementations were limited to certain classes of documents which change little across the class. To classify a more diverse range of documents with less constrained structures and no predefined features, a different type of model is required (Kang, Kumar, Peng, Li & Doermann, 2014, p. 1). One that can “learn different abstractions of structure hierarchy and spatial relationship among document elements” (Kang et al, 2014, p. 1).

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) have been successfully applied to the detection and recognition of objects in images since the early 2000s (LeCun, Bengio & Hinton, 2015, p. 439). Since their impressive performance in the 2012 ImageNet Competition, they are the most used approach for image recognition tasks, which has been facilitated by advances in software and hardware which allows faster processing and training (LeCun et al., 2015, p. 440).

The architecture of a CNN differs from that of a regular neural network (NN). The types of layer are different and typically consist of an input layer followed by any number of convolutional and pooling layers, ending with one or more fully connected layers. Instead of all the neurons being fully connected with each other, as found in the architecture of a regular NN, each neuron in a CNN only connects to a small region of the layer before it, with the exception of those in the fully connected layers (Stanford Vision and Learning Lab, n.d, para. 7). The inputs are 3D matrices of an image’s pixel values. The third dimension is the image depth which can either be one deep for greyscale or three deep for RGB coloured images (LeCun et al., 2015, p. 439).

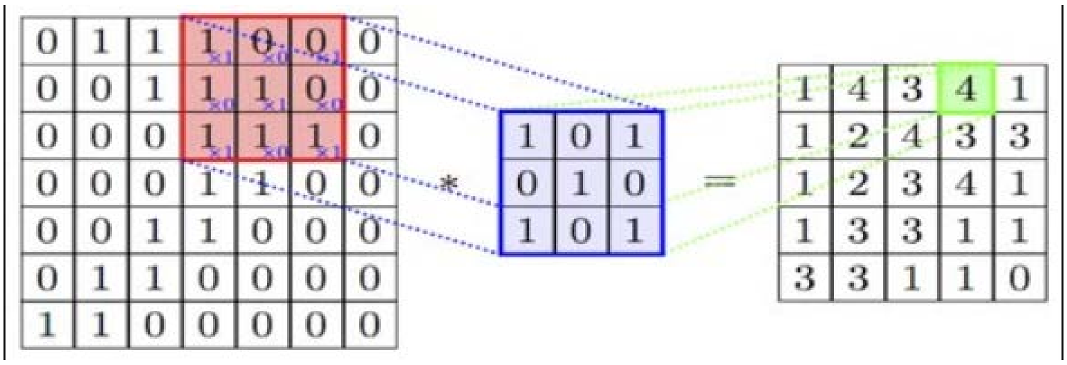


Figure 3 An example of a convolutional filter (middle) being passed over an input (left) and the sum of the products of the weights and inputs being combined into a feature map (right) (Chatterjee, 2017, para. 7).

The convolutional layers are filters of weights which are passed over the input getting the response for that filter at every spatial position in the input and producing a 2D output called a feature map. Each value on a feature map is the sum of each of the weights multiplied by the input. This becomes the sum of the dot product of the weight and multiple inputs if the input has a 3rd dimension. Each number on a feature map represents what has been detected within a certain region of the input. If the feature represented in the filter has been detected then this will result in a larger value in the feature map (Stanford Vision and Learning Lab, n.d, para. 11).

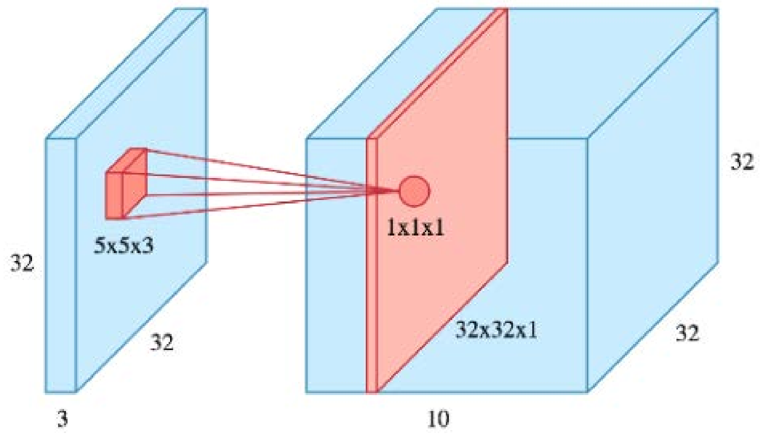


Figure 4 An example of a (5x5x3) filter being passed over a (32x32x3) input. Each distinct spatial position produces a single (1x1x1) neuron/value within the (32x32x1) feature map. This layer would have 10 filters, as the feature maps are 10 deep (Dertat, 2017, para. 13).

Multiple convolutional filters are run on each input, with each one aiming to detect a certain feature and producing its own feature map. This results in there being multiple feature maps in every layer. As every filter has its own set of weights which are passed over the entire input, it reduces the number of weights or parameters required across the whole architecture. This is known as parameter sharing (Stanford Vision and Learning Lab, n.d, para. 24).

The feature map values are passed through an activation function before being used as the input for the next layer. This is done to apply non-linearity to the architecture which has been computing linear operations during the convolutions. The most commonly used activation function in CNNs is ReLU which applies the function f(x) = max(0, x) to each element in the feature map. This changes any negative activations to 0 (Ujjwalkarn, 2016, para. 24-25).

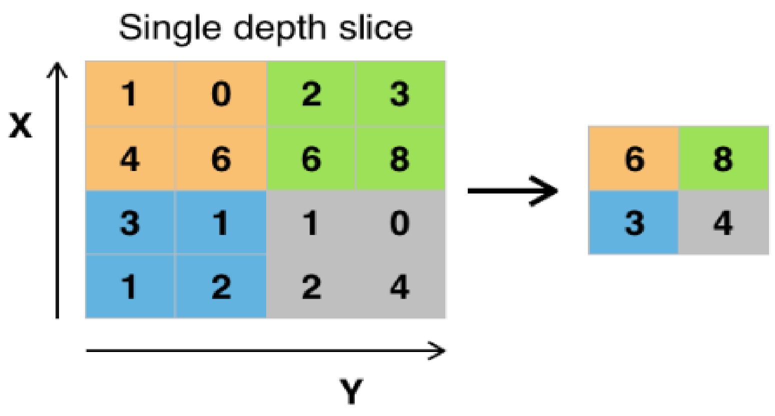


Figure 5 An example of max pooling being carried out on a feature map (Deshpande, 2016, para. 12)

Feature maps are sometimes reduced in dimensions by passing them through a pooling layer. These layers combine patches of multiple values into one value. This can be done in different ways such as averaging the values out or taking the maximum value. It is done as it reduces the number of parameters required in the network and can also prevent overfitting by providing an abstracted form of the representation, which allows there to be variation in the input. (Stanford Vision and Learning Lab, n.d, para. 41).

The convolutional and pooling layers are followed by a number of fully connected layers, which resemble those of a regular neural network. The final output from the convolutional layers are high level features which have been constructed from the lower level features searched for earlier on in the layers. Towards the start of the network these features will be edges and textures and as they travel up the network they will combine together to form motifs then parts and finally high level objects (LeCun et al., 2015, p. 439). The fully connected layers attempt to classify these high level features into the output classes the model is trying to detect. The probabilities of each class are passed through a softmax function which squashes each output value between 0 and 1, so that they sum up to 1 (Ujjwalkarn, 2016, para. 34-37).

Training a CNN will optimise the weights used in the filters and the fully connected layers of the network. This is done in the same way as common in regular NNs, by back propagating the result of the loss function and finding its gradient with respect to each weight to calculate whether they should be increased or decreased. (LeCun et al., 2015, p. 439). A popular loss function used in many experiments is categorical cross entropy, which performs better than mean squared error (Otter, 2017, para. 3). The formula can be seen below in Figure 1.

$$L_i = -\sum_j t_{i,j} \log(p_{i,j})$$

Figure 6 Formula for Categorical Cross Entropy. Where p are the predictions, t are the targets, i denotes the data point and j denotes the class (Theano, 2017, para. 12). This represents the summation of the logs of each of the outputs, multiplied by the desired target taken from the truth table (which will be 0 or 1). The non-target classes predictions end up being multiplied by 0 so they do not affect the result (McCaffrey, 2013).

There are a few techniques which are used in literature to attempt to improve the performance of CNNs. 'Regularization' techniques make slight changes to the model which allow it to generalize better and can prevent the model from overfitting (Jain, 2018, para. 4).

One such technique is Batch Normalisation which was first introduced in Ioffe & Szegedy (2015). It normalises the output of each of the hidden layer neurons. This is done by calculating the mean and standard deviation for all neurons per hidden layer throughout a training mini batch. These neurons are then normalised across the layer, so that their distribution is Standard Normal (0 Mean and 1 Standard Deviation). This is done for every layer, before the values are passed through the activation function (Kristiadis, 2016 para. 3-4). This brings the advantage of preventing neuron values from becoming either very high or low which can help prevent overfitting by adding some noise to each output (Doukkali, 2017, para. 4-6). It is also advantageous as it makes the model more robust to poor weight initialization (Stanford, n.d., para. 4).

Another commonly used regularization technique is Dropout, which works by 'dropping out' neurons randomly based on the percentage rate set. If a neuron has been 'dropped out' then its output is set to 0 during that training phase and will not be taken into account for backpropagation. This means that the CNN must find different paths through the network to arrive at the same outputs, instead of being heavily reliant on a small number of neurons. This lack of reliance on certain neurons is what stops the model from overfitting (Galeone, 2017, para. 1-5).

L2 regularization is one of the most common forms used in neural networks. It is used to reduce large weights, which forces the CNN to make use of more of its weights instead of relying on just a few large ones when making a prediction. This helps prevent the model from overfitting (Stanford, n.d., para. 25). L2 regularization is a penalty added to the loss function based on the sum of the squared weights in the entire model, multiplied by the L2 regularization rate set, which is then halved. The higher the weights are, the larger the increase to the value of the loss function. During backpropagation the calculation of the gradient for a weight will be directed more towards decreasing the weight’s value (McCaffrey, 2017, para. 4-8). The formula for L2 regularization can be seen below in Figure 7 where it has been implemented in an Ordinary Least Squares Cost Function.

$$E = \frac{1}{2} * \sum (t_k - o_k)^2 \; + \; \frac{\lambda}{2} * \sum {w_i}^2$$

Figure 7 The formula for a loss function with added L2 regularization (2nd half of the equation). The first half of the equation is the loss function Ordinary Least Squares but L2 can be applied to any loss function such as Categorical Cross Entropy. The lambda symbol represents the L2 regularization rate (McCaffrey, 2017, para. 6).

There exists an alternative means to apply L2 regularization without adapting the cost function. It is also possible to decrease each of the weights values by a certain L2 rate prior to performing backpropagation. Then the model can just perform backpropagation and update the weights as it would usually (McCaffrey, 2017, para. 11).

Momentum is a technique used to prevent models from becoming ‘stuck’ at local minima. This occurs due to the nature of the Gradient Descent algorithm preventing the parameters from shifting in a direction that would allow the function to climb out of the local minima. Instead it converges at that point. Momentum means that after backpropagation, the weights are updated by the learning rate multiplied by the gradient as per normal. However, the weight will also be updated by the momentum rate multiplied by the gradient from the previous cycle which will have been stored. This extra update keeps the parameters changing, even if the current calculation finds the function is at a local minima (McCaffrey, 2017, para. 5-8). Momentum also accelerates gradient descent, moving it towards the global minimum and minimising the function’s oscillations (movements that do not directly head towards the global minima) (Ruder, 2016, para. 19-20).

Stride and Padding are two parameters which need to be set when setting up a CNN's architecture. Stride controls how the filter passes over the input layer and refers to the 'jump' the filter makes vertically and horizontally every time it moves across the input. For example, if a stride is set to 1 then the filter passes over every possible combination of inputs. Increasing the stride will decrease the size of the output feature map making the architecture smaller (Deshpande, 2016, para. 1-3). Padding adds extra inputs around the edges of the original input, allowing the filter to pass over positions where part of it would be 'outside' of the input. The extra inputs are set to 0 as they do not contain any information; they merely allow the filter to pass over them. It has the advantage of preserving all of the input's information as any contained on the edges of the input will be kept (Deshpande, 2016, para. 4). An example of padding being applied to an image can be seen below in Figure 8.

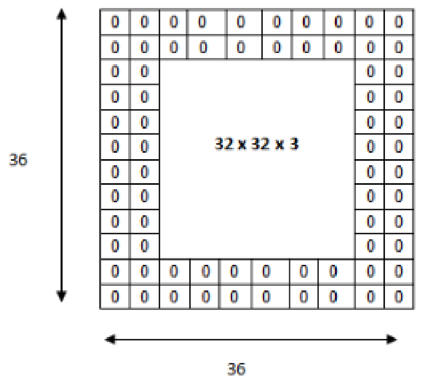


Figure 8 An example of a padding of 2 being applied to a 32 * 32 * 3 image, which makes the dimensions 36 * 36 * 3. The third dimension of the shape is not drawn. (Deshpande, 2016, para. 5).

2.3.3 Document Image Processing in Literature

There are many recent examples of Convolutional Neural Networks (CNNs) being used to classify Document Images. There are not many datasets in this field of study so most of the research uses two datasets from the same collection of documents relating to American Tobacco companies. The 'small tobacco' dataset contains 3482 documents spread unevenly across ten different classes (Lewis, Agam, Argamon, Frieder, Grossman, & J. Heard, 2006). The 'big tobacco' dataset contains 400,000 labelled images spread across 16 classes (Harley et al, 2015). The NIST tax form dataset (Garris, 1991) of 5590 tax form images spread over 20 classes is used in Kang et al (2014). These training data images are typically resized to ensure each input has the same dimensions. Although CNNs can function with rectangular inputs, the document image dimensions are squared in the discussed literature.

One of the first studies to use CNNs for document classification instead of relying on hand crafted features was Kang et al (2014). The images were resized to 150 by 150 pixels and the proposed model consisted of two convolutional layers of 20 and then 50 filters, each followed by a 4 x 4 pooling layer. After this came two fully connected layers and the softmax classifier. Drop out was used to prevent overfitting and ReLU used as the activation function (Kang et al, 2014, p. 3169-3170). It scored 65.37% accuracy on the Small Tobacco dataset (Lewis et al, 2006) and outperformed the experiments to date on that dataset (Kölsch, Afzal, Ebbecke & Liwicki, 2017, p. 2). It was also tested on the NIST dataset (Garris, 1991) but only using 1 training example per category and stopping the training after 50 epochs. This scored an average of 100% accuracy with the dataset split into 100 training and test splits. It is noted that these documents are very similar within the classes and very different across the classes (Kang et al, 2014, p. 3171). Overall this work showed that CNNs can perform well in document recognition as well as object recognition.

Afzal et al (2015) also proposed a CNN to classify the small tobacco dataset (Lewis et al, 2006). This model, along with Harley et al (2015), experimented with using transfer learning in Document Image Recognition. In transfer learning, instead of randomising the weights at the beginning of the training cycles, the weights are loaded from the optimized weights of a similar model trained on another dataset (Hulstaert, 2018, para. 4). For their experiments, Afzal et al (2015, p. 1113-1114) and Harley et al (2015) used the weights of AlexNet (Krizhevsky, Sutskever & Hinton, 2017) trained on the 2012 ImageNet challenge dataset (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg, & Fei-Fei, 2012) which is comprised of over a million images split into 1000 object categories (Harley et al, 2015). It is theorised that despite the differences in domains, that the low level features learned within this large dataset can improve the learning process of smaller datasets (Afzal et al, 2015, p. 1112).

The Afzal et al (2015, p. 1115) model scored 77.6% on the small tobacco dataset (Lewis et al, 2006), which is a significant improvement on Kang et al (2014). The model was however ‘deeper’ than Kang et al (2014), with five convolutional layers instead of three and received a larger input of 227 by 227 pixels. The input was also made 3D by passing the grayscale values into the three colour channels usually occupied by RGB. It is therefore not possible to attribute how much of the improvement was a result of the transfer learning.

The Harley et al (2015) CNN was tested on both the large tobacco dataset which they created and the small tobacco dataset (Lewis et al, 2006). The study examined whether certain regions of the document could be more important to the learning process, putting forward the idea that a CNN ran on the whole (holistic) document “may not take advantage of region specific information in document images” (Harley et al, 2015, p. 3). Five CNNs were ran, one for each of four different document regions and one on the holistic document. These 5 networks were also combined into another model which used these 5 individual results to make an overall ‘ensemble’ classification.

The study also extracted the document’s text using an OCR tool and used text based classification methods to categorize the documents. These included several Bag of Word’s (BOW) methods which use word frequency counts to train classifiers. This was done so that document image classification methods could be compared with text classification methods (Harley et al, 2015).

In the small tobacco dataset (Lewis et al, 2006) results the ensemble of the 5 CNNs performed the best at 79.9%, however the holistic was not far away at 75.6%. The holistic model was also run without transfer learning the initial weights and without them its accuracy fell to 63.4%. None of the regions perform better than the holistic CNN. The most accurate BOW model scored 68.7% (Harley et al, 2015, p. 6).

In the big tobacco (Harley et al, 2015) the accuracy of the CNNs largely increased and all the BOW models fell significantly in accuracy with the highest scoring 49.3%. The holistic CNN had the highest accuracy with 89.8% shortly followed by the ensemble with 89.3%. On a larger dataset the random integer holistic performed closer in accuracy to the transfer learning holistic with 87.8% (Harley et al, 2015, p. 5-6), indicating that transfer learning has a bigger effect on smaller datasets.

Unusually despite the same image size and model architecture, the small tobacco (Lewis et al, 2006) holistic CNN was 2% less accurate than Afzal et al (2015). This could perhaps be down to Harley et al (2015) reaching a local minimum convergence, Afzal et al (2015) using a 3D input or perhaps different weights being passed on in the transfer learning.

Kölsch et al (2017) improved on the transfer learning process by using AlexNet (Krizhevsky et al, 2017) weights in their model. It was then trained on 319,784 document images from the big tobacco dataset (Harley et al, 2015) before being tested on the small tobacco dataset (Lewis et al, 2015). This increased accuracy to 90.05% which was a significant improvement on one of their baseline results of 75.73% (Kölsch et al, 2017, p. 5) which only used the AlexNet (Krizhevsky et al, 2017) weights. This points to intra-domain transfer learning having a significant improvement on test results. The model architecture was the same as Harley et al (2015) & Afzal et al (2015) but the starting values were a 3D RGB input.

Das, Roy & Bhattacharya (2018) also used transfer learning on their model tested on the big tobacco dataset (Harley et al, 2015). The architecture matched the successful VGG16 model (Simonyan & Zisserman, 2014) comprising of 15 convolutional layers, which is at least 10 layers deeper than the other models discussed so far. The weights were also initialised with AlexNet (Krizhevsky et al, 2017) weights. On a holistic test the model scored 91.11% accuracy but then these weights were saved and used as the initializations for four separate CNNs, each one trained a different part of the document. The results of these four CNNs and the original holistic one were then used to 'ensemble' a prediction for each document using Stacked Generalization Scheme for the aggregation. This further increased the test accuracy to 92.21% (Das et al, 2018, p. 5). Even without the 'ensemble' of the 5 different CNNs, the results were higher than Harley et al (2015). This was likely down to the VGG16 architecture and its 3D inputs.

Despite all these positive results using transfer learning, Tensemeyer & Martinez (2017) are sceptical that weights from completely distinct domains can actually assist in the learning process. Their extensive study into what aspects of a CNN really affect document classification performance scored 90.8% without using any transfer learning. This was done using a 348 pixel 1D greyscale input and the research found that increasing the input size almost always increased the accuracy of the test, even with very small training sets.

Their tests on input show that Grayscale performs well, although a slight improvement can be made by combining it with SURF. Dropout and Batch Normalization were also shown to increase accuracy (Tensemeyer & Martinez, 2017, p. 2-5). With regard to the number of convolutional layers, as the depth increased, the performance fell, even with a very large training set. However it was most noticeable on smaller training sets and they argue that 2 convolutional layers perform better on training sets smaller than 32,000 documents (Tensemeyer & Martinez, 2017, p. 2-4).

2.3.4 Limitations of CNNs for Image Processing

CNNs have been shown to be powerful instruments in document classification; however they do have some drawbacks. Firstly, they require a lot of data to train. As shown in the datasets used in the discussed research so far, thousands of training samples are required. They also require a lot of time and processing power to run. In Kölsch et al (2017), the research was investigating a less time-consuming way of re-training a CNN so they could be used in real time situations. They succeeded in reducing the processing time but at the expense of a drop in accuracy.

Invariances in the document image such as rotation can also go undetected by a CNN. Images at different angles were shown to be very hard to predict correctly in Gong, Wang, Guo & Lazebnik (2014), where the CNN was trained on a standard dataset and then tested with flipped and rotated images from the test set. The higher the rotations of the images the less accurate the predictions were. Unusually, they almost reached the original accuracy levels once they were horizontal.

Like other machine learning models, overfitting can also be a problem. This occurs when the CNN learns to recognise the exact dataset instead of the classes (Stone, Cheryl, Pammer, Steele & Keller, 2015, para 2). There are ways to counter this, 'dropout' for example was shown to be effective in Tensemeyer & Martinez (2017) and was used in all the Document Image Classification research discussed. Data augmentation which is discussed in the below chapter can also be effective.

Deeper CNNs have often been shown to perform worse than shallower versions as shown in Tensemeyer & Martinez (2017) and (He, Zhang, Ren & Sun, 2015) who went on to propose a way to combat this. In their research they proposed a network architecture comprised of 'residual layers', a type of shortcut connection that skips one or more layers and the outputs are added in again further on in the layers. An example can be seen below in Figure 9. Using this method they won first place in the ILSVRC 2015 classification competition with a network architecture of 152 layers (He et al, 2016).

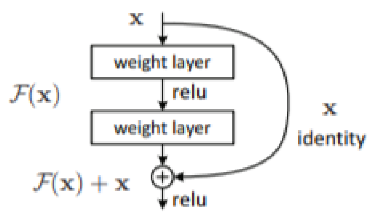

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 9 An example of a residual learning block, with a skip connection (also known as an Identity Mapping) being passed from a neuron to another neuron further on in the network (He et al, 2016, p. 2).

2.4 Open Set Recognition

“Traditional supervised learning makes the closed-world assumption that the classes [which] appeared in the test data must have appeared in training” (Shu, Xu & Liu, 2017, p. 1). This assumption that all possible categories are known during training is not always the case when deploying recognition systems in the ever changing real world (Bendale & Boult, 2015, p.1). This leads to the field of study known as Open Set Recognition, where “incomplete knowledge of the world is present at training time, and unknown classes can be submitted to an algorithm during testing” (Scheirer, Rocha, Sapkota & Boult, 2013, p. 1). For example, if there was an underwater camera that recognised species of fish and automatically categorised them based on their image. It must be able to tell when it has seen a new species of fish, instead of incorrectly classifying it as one of the categories the ML algorithm has been trained on (Scheirer, 2018, para. 2).

One approach to tackling this problem is to train the ML algorithm with an “other” output class for all non- seen classes to potentially fall into. It is however not possible to train an algorithm on every single unseen category, so there have been various attempts to tackle this problem a different way (Bendale & Boult, 2015, p.1).

Bendale & Boult (2015), developed a function labelled ‘Openmax’, which was proposed as an alternative to the softmax output function. It consisted of a computed threshold which predictions had to be greater than, in order to be classed as one of the known categories, if not they were rejected as an unseen class. As the softmax function only spreads the class probabilities over the known class labels, it is theorised that it would not be suitable to calculate the threshold on these values. Therefore the OpenMax function instead used the raw values of the neurons in the output layer of the CNN. To calculate the threshold, the output value for each of the correctly predicted training samples was recorded and the average calculated for each class. These were used to compute the threshold using a Weibull fitting method. Any test items with outputs lower than their corresponding thresholds was classed as an unseen class (Bendale & Boult, 2015, p.7).

The model was tested on a CNN trained on the ImageNet 2012 dataset with 1000 image categories (Russakovsky et al, 2012), however for the test data 50k images from this dataset were also combined with 15k images from 360 previously unseen categories from the

ImageNet 2010 competition (Russakovsky, Deng, Su, Krause, Satheesh, Ma, Huang, Karpathy, Khosla, Bernstein, Berg & Fei-Fei, 2015). For evaluation they compared the proposed Openmax classifier with the same threshold classifier which was instead used on the standard softmax output layer. It was also compared against a standard CNN without rejection capability. It found that Openmax's F-Score for the test was 4.3% more accurate than the Softmax classifier and 12.3% more accurate than the standard CNN (Bendale & Boult, 2015, p.7).

Shu et al (2017) proposed an open set recognition system called Deep Open Classification (DOC). It attempted to improve on this technique by also collecting the output of each correctly predicted training class to create a prediction threshold. However instead of using the raw output, this output was passed through the sigmoid activation function. The CNN model was also trained using a different loss function using a 1 Vs rest layer, where 1 was the desired output which is on the positive side of the loss function and the remaining incorrect outputs are on the negative other side (Shu et al, 2017, p. 3).

This loss function differs from Categorical Cross Entropy as it tries to both maximise the desired output and lower the incorrect outputs, compared to Categorical Cross Entropy which only computes the loss error on the correct label's prediction. It does not reward or penalise the predictions on the incorrect labels (McCaffrey, 2013).

For a prediction to be categorised as a class during testing, the sigmoid output had to be higher than the computed threshold for that predicted class. If not, it was classed as an unknown class. The threshold was calculated as 1 minus the standard deviation across all of the correctly predicted training outputs for the predicted class. A minimum threshold of 0.5 was also implemented as a separate experiment to see if it improved results, called DOC (t=0.5) (Shu et al, 2017, p. 3).

The CNN was different to Bendale & Boult's (2015) in that it was used for text classification with first layer embedding words into vectors (Shu et al, 2017, p. 2). The experiments were run on two datasets, one of Amazon reviews categorised by product and another of 20 news documents categorised by topic. They were set up in four different tests where 25%, 50%, 75% and 100% of classes had previously been seen. Evaluation was done by calculating F1 scores. They tested both the OpenMax classifier from Bendale & Boult's (2015), as well as the threshold they created, both with the minimum of 0.5 implemented and without (Shu et al, 2017, p. 4).

Overall both DOC setups performed significantly better than OpenMax on every experiment. As the number of seen classes decreased the higher both the DOC variations performed over OpenMax. Out of the two DOC classifiers, DOC performed better than DOC (t=0.5) in all but one of the experiments (Shu et al, 2017, p. 4).

2.5 Synthetic Data

Supervised learning is currently restrained by a lack of labelled training data and document image classification is no different. For example as we have already seen within Document Image Classification, there is only one existing large dataset and this was not created until 2015. Acquiring large amounts of labelled data is not always possible and other times it could be too expensive or time consuming for both companies and researchers alike. One potential way to overcome this is by making artificially constructed data just for the purpose of training machine learning algorithms, which has been done both in literature and in business.

Within small AI start-ups “spoofing training data is becoming a legitimate strategy to jumpstart projects when cash or real training data is short” (Simonite, 2018, para. 5) which can “level the playing field between start-ups and big companies” (Simonite, 2018, para. 7). German company Spil.ly trained the CNN for its camera augmented reality app using only artificial data created by techniques used to make video game graphics, thus saving them on having to pay for real videos (Simonite, 2018, para. 3).

Within literature, in Le, Baydin, Zinkov & Wood (2017) a captcha- breaking Neural Network was trained on purely synthetically generated data and achieved state of the art results. In Jaderberg, Simonyan,Vedaldi & Zisserman (2014) a convolutional neural network was trained to detect text in ‘natural scenes’, using only synthetically created data and outperformed all other studies on standard datasets.

2.6 Augmented Data

Other studies have augmented the data they already have by creating multiple variants of the pre labelled training examples. In Lecun, Bottou, Bengio & Haffner (1998) 60,000 image training samples were copied 9 times with transformations such as squeezing, rotation and scaling applied to them. This dropped the test loss rate from 0.95% to 0.8% on the same testing set.

Tensemeyer & Martinez (2017, p. 3) used 10 different types of transformations such as blur, crop and rotation to artificially enlarge their training set and see the effect they would have on their CNNs accuracy. The results were mixed, with some of the artificial data improving results and others lowering them. It must be noted that the test data was also augmented with the same synthetic transformations, therefore each test was in theory more difficult than the original test, which already had a very high accuracy of 88.30%. Therefore it could be argued that as none of the transformations except rotation really dropped the accuracy, then adding these transformations to the training set can help the network predict the classes of documents which have been scanned in poorly.

2.7 Literature Review Conclusion

Chapter 2 has explained and investigated recent approaches to Document Image Classification. Two text classification techniques Support Vector Machines and Naive Bayes were investigated and reviewed in Chapter 2.2 and were both found to have been successful in classifying documents through their text. In Chapter 2.3 CNNs were discussed in detail and developments in the Document Image Classification field such as Transfer Learning, Residual Layers, Batch Normalization and were found to have been effective in improving CNN performance in literature. Chapters 2.4 and 2.5 focused on looking into ways of solving current issues being experienced in the broader Image Processing field. It found evidence that augmented data could solve the lack of training data problem and that there had been some successes in reconfiguring a CNN to be able to detect previously unseen classes.

3 Design and Implementation

3.1 The Datasets

The big tobacco dataset (Harley et al, 2015) which has already been mentioned in the literature review will be the predominant dataset used for these tests. It is made up of 400,000 grayscale images spread over the 16 document classes contained in Table 1 below.

Table 1 The 16 classes of the Big Tobacco dataset.

0	Letter	8	File Folder
1	Form	9	News Article
2	Email	10	Budget
3	Handwritten	11	Invoice
4	Advertisement	12	Presentation
5	Scientific Report	13	Questionnaire
6	Scientific Publication	14	Resume
7	Specification	15	Memo

On an initial inspection, some of the differences in classes are almost indistinguishable to the human eye. It also appears to contain a few faulty documents that do not open. Others are poorly scanned or at skewed angles which could cause classification issues.

In order to improve Dialexy’s CNN, some experiments will be run on 10 of the classes which are most structurally similar to their data. For example, ‘handwritten’ could apply to any of their documents and ‘file folders’ are just a grayscale dark scans of the outside of a folder. These should not be hard to classify but as they do not resemble Dialexy’s documents, they will be initially left out due to a limit on the processing power available.

As the company wants to implement a version of the CNN that can detect 10 different types of document, this will give a good indication of how the model will perform in its business implementation. Therefore initial experiments have been carried out with the following ten classes Letter (0), Form (1), Email (2), Advertisement(4), Scientific Publication(6), News Article (9), Budget (10), Questionnaire (13), Resume (14), Memo (15).

The company’s data is made up of various personal and legal documents, such as passports, degrees and birth certificates. As it is a translation company, they are in a number of different languages, though the structure is similar for some of the documents of the same class across the different languages. However some documents vary enormously within the same class even if they are from the same country. This can range from different structures, colours, logos and having hand written sections. Some documents would even be very hard for a human to manually classify due to some containing very limited information regarding their purpose. This will all make it very difficult for the CNN to spot patterns amongst the classes and will provide a challenge in training.

As the texts are in different languages, this would need to be detected before running the text based classifier. This can be done by using a language detection API on the company’s server, so the language models will assume that the language for each test document has already been detected.

Due to the lack of training data, there are only really 3 classes (Birth Certificate, Degree Certificate & Passport) which can be trained and tested on for the CNN tests, as they each have over 100 documents. The other classes contain less than 50 documents in the class, which will not be enough to make a training or testing set of any significance for the CNN, however an attempt to be made to run the language models with these smaller classes to see how they perform.

3.2 The Models

3.2.1 Image pre-processing

The image pre-processing is carried out using Wand, which is a python wrapper for the ImageMagick software (ImageMagick, n.d.). The documents are firstly loaded into a Wand image object and converted to grayscale. The image is then resized to match the desired height and width before being converted into a Numpy array of pixel values. They are then normalised between 0 and 1 by dividing by 255 and reshaped back into the 2d matrix representation of the image. For any regional based experiments, the image is split and resized before being converted into the Numpy array.

If the experiment contains any synthetic data this also occurs at this stage by using the following 5 possible transformations on the image:

- **1. Rescale**: Crops the image from each of the 4 sides and resizes it back to its original size. The amount to crop is a percentage of the image's dimension. This percentage is calculated as 2 to the power of a randomised integer in the range -2 to 3.

- **2. Rotate**: Using the Wand 'Rotate' method, the image is randomised between every possible 2 degree jump between -4 degrees and +5 degrees, either side of 0, 90, 180 and 270 degrees. This represents the image on both of its sides and also upside down.

- **3. Contrast**: Contrast is randomly increased between 0 and 5%.

- **4. Brightness:** Brightness is lowered to either 80%, 90% or is kept as it is.

- **5. Noise**: Uses the Wand 'evaluate' method to apply Gaussian Noise across the image. This changes the pixel values across the image following a Gaussian distribution.

A combination of the five transformations above is randomised and applied to the synthetic copy of the image which is then used in the CNN.

3.2.2 The CNN Model

The CNNs are built in the Python library Lasagne (Lasagne, n.d.) which is "a lightweight library to build and train neural networks in Theano" (Lasagne, n.d.). Theano is a Python library that allows users to "define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently" (Theano, n.d.). It provides the tools required for the building of complex machine learning models. Theano provides features which facilitate the training process. It allows for faster processing by moving parts of the program to run on the computer's Graphics Card. It will also compile some Python Code into C++, to allow for faster and more stable computations. Theano also takes care of calculating the backpropagation of the model and will calculate the gradient for each of the model's parameters after each epoch (Theano, n.d.).

Although two different architectures will be tested, the CNN used in most of the tests will keep a similar structure. It will be comprised of the residual blocks with identity shortcuts used in He et al (2016).

Each residual block is made up of two convolutional layers and each of these layers is made up of 3x3 filters. The number of filters varies, some residual blocks have the same number of filters as there are output units in the previous layer, the others are set to have double the filters than there are outputs in the previous layer. If the number of filters is doubled, then the stride in the first residual layer is set to 2, if not the stride is set to 1. The stride on the second convolutional layer in the residual block is always 1. A stride of 1 will pass over every possible square of 9 input pixels or neurons but a stride of 2 will jump a neuron vertically and horizontally every time the filter switches position. This will reduce the dimensions of the feature map without the need for pooling. A padding of 1 unit is used around each input, which when combined with stride 1 results in the output size equalling the input size (Lasagne, n.d.).

The input dimensions for these tests will be 1 * x* x, where x is the height and width of the image, which will vary throughout the tests. The first dimension is set to 1 as the majority of the documents are black and white, so an RGB input would just add unnecessary complexity. On the live system the documents will also be converted to greyscale before the algorithm is run on them.

To explain the rest of the architecture an input image of 1 by 128 by 128 pixels will be used as an example. The architecture commences with 1 convolutional layer which passes 16 sets of 3 by 3 filters over the input at a stride of 1 so the 1 * 128 * 128 input becomes 16 feature maps of 128 by 128. Batch normalisation is also applied to this first layer. This is then followed by 4 stacks of 5 residual blocks. As there are two layers in a residual block then each stack contains a total of 10 layers. The first stack of residual layers has 16 filters so the output remains 16 * 128 * 128. The second stack of layers has 32 filters so the output dimensions change to 32 * 64 * 64. The third stack of layers has 64 filters so the dimensions change to 64 * 32 * 32. The fourth and last stack of layers has 128 filters so the dimensions change to 128 * 16 * 16. This is then followed by a mean pooling layer which feeds into a fully connected layer with an output neuron for each class. All the layers use the ReLU activation function on their respective neurons with the exception of the final fully connected layer which uses softmax to facilitate the final classification. The architecture of the CNN is shown below in Figure 10.

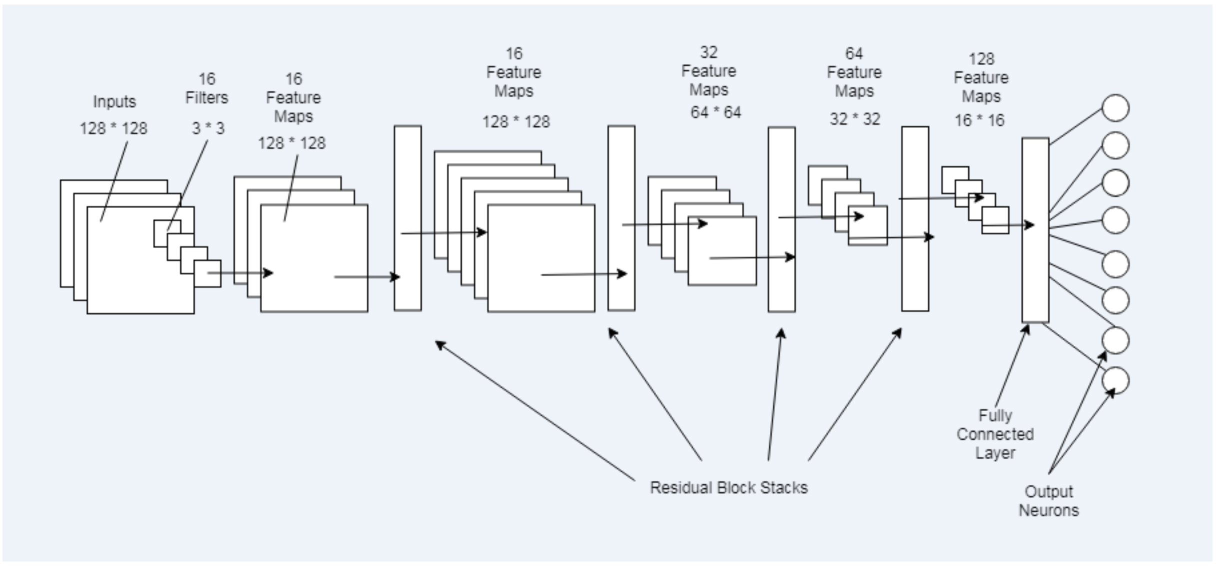


Figure 10 A diagram of the proposed CNN architecture used in the majority of experiments.

3.2.3 Training the CNN

The CNN is trained using categorical cross entropy as it is the loss function used in the majority of the literature discussed. Similarly to He et al (2016) and Ioffe & Szegedy (2015) dropout is not used as batch normalization will reduce the need for it in preventing overfitting. The model will be trained on the training data for 81 epochs; this will allow it to reach convergence each time, even with large datasets. The learning rate is set to 1% as in He et al (2016) and is automatically reduced to 0.1% on epoch 40 and 0.01% on epoch 80. Momentum is used and set at a rate of 90% and L2 regularization is set to 0.01%, which matches He et al (2016).

The training data will also be processed in batches, as it cannot all be sent to the CNN at one time. After each batch, the network will update all of its parameters based on the loss function. If the batch size is too small then the parameters will update too often as they will not have sampled enough training items. This can result in a lot of noise in the model which can lead to early convergence. The drawback of a large batch size can be that as the entire batch is sent to the processor at once, it requires a lot of memory. Due to the large amount of memory images require and the limited resources at Dialexy's disposal, some of the tests have had to be run using a smaller batch size than would be optimal, to avoid system crashes.
Evaluation will be carried out using accuracy as this seems to be the main metric in the literature reviewed such as Harley et al (2015) and Das et al (2018). It is important however,

that equal numbers of each training class are used, so that the CNN’s results are fair. Using a cross validation would give a better evaluation of the model being tested but due to the extremely long time it takes to train the CNN, it will not be feasible for these tests. Also due to the limited training documents that Dialexy have, the experiments using the company’s data will not use a validation set. This is because every possible document that exists will be needed for the training and test sets.

The CNN will be processed using a MSI NVIDIA GeForce GTX 970 (4096 MB) graphics card which will significantly increase processing speed over CPU processing.

3.3 Text Classification

Due to the nature of the text required to be classified, it should hopefully not be necessary to implement a complicated model in order to classify documents correctly from their texts. Most documents will be quite similar textually as many of the documents within the same class will be produced by the same institutions or businesses giving them similar key words and titles. Furthermore, it is not necessary for the model to ‘understand’ the meaning of the document in order to classify it; it should be able to assign the document automatically from words such as “passport”. Therefore for the text based classifiers this work will implement and compare Naïve Bayes, SVM and a custom made model. These models will be tested on both Dialexy’s data and the text from the tobacco dataset (Harley et al, 2015). It is likely that the results for the latter will be poor due to most of the documents being handwritten but they should give a good indication of which models are classifying more accurately. Both sets of documents were subject to the same pre-processing and will be evaluated using F score and accuracy to match the metrics used in the text classification literature.

3.3.1 Language Models Pre-Processing

An OCR tool hosted on Dialexy’s server was used to extract the text contents from the documents. The returned string was tokenized and kept as unigrams, as in Li & Jain (1998). The reasoning behind this is that the aim of the model is to pick out the key words a document contains. Knowing the relationships between the words is not so important for this type of classification. Stop words, numbers and all special characters were removed, again this was done to try and separate the key words from the other ‘noisey’ words, which would interfere with the model. Stemming is also applied as there are many plurals of the same word such as “birth” and “births”, which should have the same meaning.

3.3.2 Language Models

3 different models will be used for text classification and their results compared:

1. The first model implemented is multinomial Naïve Bayes, which will be implemented using the python library ‘naive-bayes-classifier’ (Muatik, n.d.).
2. The second model will be a Support Vector Machine which will be ran using sklearn’s SVM function. Each feature is converted into a vector of numbers which represents the word counts for each word in the training data. Only words which appear 15 times or more in the training data will be included in the vectors.
3. The third model will be a relatively simple custom made model which searches for keywords in the documents. Firstly for each document type, a bag of words with word counts is created from the training data. The unique words are then ordered by frequency with the most common appearing highest in the list. The top n number of words is then taken and used to create a scoring system, where the most common word is scored at n+1 and each word further down the word list scoring 1 less than the one before, until the final word, which is the nth most common, scores 1. The value for n will depend on the number of documents, as with more documents it should take a larger sample of the words to be able to differentiate between them and classify them. Therefore n is decided by taking the total number of unique words in each of the document classes and dividing it by the number of documents in the training data for that class. This is then averaged out across the classes. For each of the test items, every word was searched for within the most common word lists of each class. If one of these words was found, then its score was added to a cumulative total calculated for each of the classes. The classification of the test item was the class which had the maximum score out of all of the class’s cumulative word scores.

All three models will be trained and tested using documents that are in the same language only, as this represents the model’s live implementation where the language will be detected before the text analysis. The document classes will also be split into 75% training data and 25% test data.

4 Testing and Results

This section will outline each of the different experiments, which will be investigating the effect of a particular technique or parameter on the CNN. Each experiment will be made up of two or more separate tests and the results will be displayed in tables. The results will then be discussed in Section 5. This section will be split into a section for image processing tests (Section 4.1) and text classification tests (Section 4.2).

4.1 Document Image Processing Experiments and Results

4.1.1 Image Size Experiments

In this test the effects image size can have on the CNN are examined. The experiment was firstly carried out on the 10 document classes of the tobacco dataset. Other than the change in image size, all other variables were kept the same and the images were always square. The only exception to this was batch size, which had to be lowered from 10 to 5 for Test 4 with 256 pixels in order to avoid crashes. Exponentials of 2 are used for image size as they work well with the architecture as many of the layers halve the dimension as it passes through. The results are shown below in Table 2.

Table 2 Big Tobbaco Image Size Results.

Test	Image Size (Pixels)	Training Data	Test Data	Accuracy (%)
1	32	4500	750	71.73
2	64	4500	750	73.73
3	128	4500	750	81.87
4	256	4500	750	83.07

The same experiment was then carried out on Dialexy’s data and the results are shown below in Table 3. For this test all of the parameters except the image size were kept the same including the batch size.

Table 3 Dialexy Data Image Size Results.

Test	Image Size (pixels)	Original Training Items	Number of Augmented Copies	Total Training Data	Test Data	Accuracy (%)
1	32	90	10	900	60	43.33
2	64	90	10	900	60	75.00
3	128	90	10	900	60	80.00
4	256	90	10	900	60	78.33

Due to the extra difficulties caused with memory shortages and the time taken to run the CNN with 256 pixel images, the rest of the tests will predominantly use 128.

4.1.2 Different Architecture Experiments

In this experiment different CNN architectures are tested to see which performs better. All tests are run on 10 classes of the Big Tobacco dataset. Test 1 is run on the exact 34 layer residual architecture used in He et al (2016), using the Projection Shortcut (Option A in the paper). It is also run on a pixel size of 224 as to exactly match the experiment run in the paper. Test 2 is a Lasagne adaption of this architecture described in 3.3.2 with identity shortcuts (Option B in the paper). It has 42 layers and is run on image size of 256 pixels. The results are shown below in Table 4.

Table 4 Dialexy Data Transfer Learning Results.

Test	Architecture	Pixel Size	Training Data	Test Data	Accuracy (%)
1	34 Layer with Projection Shortcuts	224	4500	750	81.07
2	42 Layer Lasagne Implementation with Identity Mappings.	256	4500	750	83.07

4.1.3 Regional Experiments

This experiment was carried out in order to see whether allowing a CNN to concentrate on a particular region of the document only may improve performance, as each CNN can concentrate on only detecting features that appear in their respective regions. Also splitting each document into 4 x 128 by 128 images will give the CNN a chance to learn smaller features that it wouldn’t be able to detect otherwise, such as very small text.

For these experiments the document images were halved and then halved again, splitting them into 4 parts. In Test 2, the split documents were tested with four CNNs, where each one was trained on a certain region of each document. Once each network had converged the test images were split and the regions were sent to their respective networks and the score for each document section was recorded. Once each of the image's regions had been classified, the most frequently selected class was assigned as the document's prediction. If there was a tie between the predictions, then one of the tied classes was chosen at random. The accuracy of each individual region was also recorded in the test, to see if one part of a document is more distinguishable than the others.

This is compared against Test 3 where everything was exactly the same as Test 2, except that there was only one CNN instead of 4. The CNN was trained with each of the 4 regions of the training documents and each region of the test documents were tested on this CNN.
These two results are compared against Test 1 with no splits but with all other parameters the same. All 3 tests were run with 128 by 128 pixel sized documents on 10 classes of the big tobacco dataset (including each split which was resized to 128 by 128 pixels). The results are shown below in Table 5.

Table 5 Big Tobbaco Regional Results.

Test	Training Data	Test Data	Splits	CNNs	Accuracy(%)
1	4500	750	0	1	81.87
2	4500	750	4	4	79.47
3	4500	750	4	1	80.67

The regional test results for the experiment with 4 CNNs are shown below in Table 6. Each of the regional CNN's individual test accuracies were recorded from the test shown in Table 4 above.

Table 6 Split Document Results by Region

Document Region	Accuracy (%)
Top Left	77.20
Top Right	65.73
Bottom Left	72.13
Bottom Right	66.80

4.1.4 Transfer Learning Experiments

This experiment will investigate the effect transfer learning can have on the training of a CNN. In transfer learning, weights from another converged model are used as the initialised weights for the model in the current experiment.

The big tobacco dataset was run on the 3 classes letter (1), form (2) and resume (15), which are documents similar to the three that must be predicted accurately for the company.
750 items of training data were used for each class and the pixel size was set to 256. The model was then trained until it converged. The model's parameters were saved to be used as the initialising values of the parameters for the model using the company's data. These were used in Test 1 which was ran on 3 of Dialexy's documents (Birth Certificate, Degree Certificate & Passport) with 50 training items and 20 original test documents. 64 synthetic versions of each training item were created making the number of training items 3200. It is compared against Test 2 where the only difference is that the weights were initialised randomly instead of transferred from another model. The results are below in Table 7.

Table 7 Dialexy Data Transfer Learning Results.

Test	Weight Initialisation	Accuracy (%)
1	Random	86.67
2	Transfer Learning	88.33

4.1.5 Augmented Data Experiments

These experiments will investigate the effects that augmented and synthetic data can have on a model's accuracy. The first experiment investigates the different ways a small dataset can be used to train a model, using only 50 training items from each class from 10 classes in the tobacco dataset. This represents the problem Dialexy have with their lack of training data. The results are shown below in Table 8.

Test 1 was run with just the original unchanged 500 documents as the training data, to be used as a baseline. In Test 2, 9 copies of each training item was created by applying random versions of each of the five transformations listed in section 3.3.1 and applying them to each of the original training documents.

In Test 3 the original 500 documents were duplicated 9 times to enlarge the training set. They were exact copies with no augmentations implemented. Test 4 was identical to Test 2, except that 45 augmented copies were made instead of 9 and Test 5 was identical to Test 3 except that each document was put into the training set 45 times instead of 9. In the same way Tests 6 and 7 were identical to Tests 4 and 5 except that the copies were further increased to 60 each.

Table 8 Big Tobacco Augmented Data Results.

Test	Original Training	Test Items	Exact Copies	Augmented Copies	Total Training Items	Accuracy (%)
1	500	250	0	0	500	52.80
2	500	250	0	9	4500	54.40
3	500	250	9	0	4500	65.60
4	500	250	0	45	22500	66.80
5	500	250	45	0	22500	64.80
6	500	250	0	60	30000	68.80
7	500	250	60	0	30000	62.00

In the following experiment the CNN is tested on poor quality images and rotated documents which represent documents likely to be uploaded to the company’s platform by clients. So it is important that the CNN is trained to still be able to detect these documents.

Table 9 shows the results of an experiment which was carried out on the Big Tobacco dataset. For Test 1, the CNN was only trained on good quality and upright images. In Test 2 however, 4 augmented copies of the original image plus the original image were used for training. All 5 of augmentations described in section 3.3.1 were used.

Table 9 Big Tobacco Rotated Data Results

Test	Original Training Items	Augmented Copies	Total Training Items	Test Items	Accuracy (%)
1	4500	None	4500	15000	21.06
2	4500	4+ 1	22500	15000	70.53

4.1.6 Open Set Recognition Experiments

In these experiments the CNN's ability to detect documents in previously unseen classes is analysed. The first experiment is carried out on 3 classes of Dialexy's data (Birth Certificate, Degree Certificate & Passport), with the unseen class being made up of a random selection of documents that do not belong to any of the 3 classes. Test 1's CNN is trained on a random sample of these 'other' documents, so there are 4 classes being trained on in total. In Test 2, the CNN is trained and tested in the same way as the DOC classifier in Shu et al (2017). The CNN is only trained on 3 classes and then a threshold is implemented on the output layer of the network to detect the unknown class. As per the paper, DOC (t=0.5) which is explained in section 2.4 of this paper, is also used in Test 3. The results are shown below in Table 10.

Table 10 Dialexy Data Open Set Recognition Results.

Test	Method of Detecting Unseen Classes	Training Items	Test Items	Accuracy (%)
1	Train on 'other' class	11712	80	85.00
2	DOC (t=0.5)	8512	80	77.50
3	DOC	8512	80	73.75

In the following experiment, four classes from the Tobacco set email (2), letter (1), Scientific Publication (6) and Questionnaire (13) are used as four separate known classes. Four different tobacco classes are combined and randomly sampled to make an unseen class. For the 'other' class in Test 1, random documents from the remaining 8 unused classes from the Tobacco set were used for training. Tests 1 and 2 used the two versions of the DOC threshold explained in the last experiment. The results can be seen below in Table 11.

Table 11 Big Tobacco Open Set Recognition Results.

Test	Method of Detecting Unseen Classes	Training Items	Test Items	Accuracy (%)
1	Train on ‘other’ class	2400	300	73.00
2	DOC (t=0.5)	1920	300	68.66
3	DOC	1920	300	69.00

Confusion Matrices for Tests 1 and Test 2 are shown below in Table 12 (Test 1) and Table 13 (Test 2).

Table 12 Confusion Matrix for Test 1 from Table 11, training the CNN on the ‘Other’ class.

	Email	Letter	Scientific Publication	Questionnaire	Unseen Class
Email	58				2
Letter		43	1	4	12
Scientific Publication		2	50	3	5
Questionnaire	3	4	1	46	6
Unseen Class	5	18	9	6	22

Table 13 Confusion Matrix for Test 3 from Table 11, using the DOC Class threshold.

	Email	Letter	Scientific Publication	Questionnaire	Unseen Class
Email	54			1	5
Letter		40	1	4	15
Scientific Publication			46		14
Questionnaire	1	1		30	28
Unseen Class	3	9	8	3	37

In the following experiment the parameters saved from the trained models in Table 9 were loaded into a CNN which was then tested on 1052 images of flowers from the Flowers Recognition Set on Kaggle (Mamaev, 2018). All of these images should be of the unknown class, so the test was to investigate whether the models could identify them as unseen despite them being from a different domain to documents used to train the parameters. This experiment was run multiple times as it was fast due to the models already being trained. The exact same 3 methods of detecting unknown classes as the last experiment were tested. The results can be seen below in Table 14.

Table 14 Flower Recognition Set Results.

Test	Method of Detecting Unseen Classes	Training Items	Test Items	Test 1 Accuracy (%)	Test 2 Accuracy (%)	Test 3 Accuracy (%)	Average (%)
1	Train on 'other' class	0	1052	0	100	0	33.33
2	DOC threshold 0.5	0	1052	61.88	57.03	41.83	53.58
3	DOC	0	1052	88.33	44.67	90.40	74.46

4.2 Text Based methods

This section will present experiments and results from the language models. As the documents are each in different languages then two separate tests in English and Spanish will be carried out for the company's documents, as these are the only two document types with sufficient data to create a model. Unfortunately these classes had very small datasets. For each test the Training data and Test data was split in a way so that the number of testing documents for each class was equal. In each results table, the heading 'Minimum Docs' refers to how many instances of a class there has to be for the class to be included in the test.

For the tests on the Big Tobacco dataset, 10 classes were used Letter (0), Form (1), Email (2), Scientific Publication (6), Specification (7), News Article (9), Budget (10), Presentation (12), Questionnaire (13) and Resume (14). These were chosen as they were the documents that contained the most text. For each of the 10 classes, 160 documents were split into 75% training data (120) and 25% testing data (40). To evaluate the results, both the F1 Score and accuracy of each test was recorded.

4.2.1 Naive Bayes

The results for the Naive Bayes Model on Dialexy's documents can be found below in Table 15.

Table 15 Dialexy Naïve Bayes Results.

Test	Language	Minimum Docs	Total Classes	Training Docs	Testing Docs	F1 Score	Accuracy (%)
1	English	8	8	119	32	52.83	53.125
2	Spanish	8	6	158	24	81.48	79.17
3	English	10	5	100	25	64.84	68.00
4	Spanish	10	5	148	25	91.67	92.00
5	English	20	3	83	15	65.08	66.66
6	Spanish	20	4	141	20	89.58	90.00
	Average					74.25	74.83

Confusion Matrices for Test 3 (Table 16) and Test 4 (Table 17) can be found below.

Table 16 Confusion Matrix for Test 3

	Academic Transcript	Birth Certificate	Degree Certificate	No Impede Certificate	Payslip
Academic Transcript	4		1		
Birth Certificate		5			
Degree Certificate	4		1		
No Impede Certificate		3		2	
Payslip					5

Table 17 Confusion Matrix for Test 4.

	Academic Transcript	Birth Certificate	Criminal Record	Degree Certificate	Payment Receipt
Academic Transcript	3		2		
Birth Certificate		5			
Criminal Record			5		
Degree Certificate				5	
Payment Receipt					5

The results for the Naive Bayes Model on 10 classes of the Big Tobacco dataset can be found below in Table 18.

Table 18 Big Tobacco Naïve Bayes Results.

Test	Total Classes	Training Items	Testing Items	F1 Score	Accuracy (%)
1	10	1200	400	49.65	47.00

4.2.2 Support Vector Machine

The results for the SVM Model on Dialexy's documents can be found below in Table 19.

Table 19 Dialexy Data SVM Results.

Test	Language	Minimum Docs	Total Classes	Training Docs	Testing Docs	F1 Score	Accuracy (%)
1	English	8	8	119	32	81.90	81.25
2	Spanish	8	6	158	24	68.51	75.00
3	English	10	5	100	25	91.36	92.00
4	Spanish	10	5	148	25	79.70	84.00
5	English	20	3	83	15	93.27	93.33
6	Spanish	20	4	141	20	75.00	80.00
	Average					81.62	84.26

Confusion Matrices for Test 3 (Table 20) and Test 4 (Table 21) can be found below.

Table 20.Confusion Matrix for Test 3

	Academic Transcript	Birth Certificate	Degree Certificate	No Impede Certificate	Payslip
Academic Transcript	3		1		1
Birth Certificate		5			
Degree Certificate			5		
No Impede Certificate				5	
Payslip					5

Table 21 Confusion Matrix for Test 4

	Academic Transcript	Birth Certificate	Criminal Record	Degree Certificate	Payment Receipt
Academic Transcript	1		2	1	1
Birth Certificate		5			
Criminal Record			5		
Degree Certificate				5	
Payment Receipt					5

The results for the SVM Model on 10 classes of the Big Tobacco dataset can be found below in Table 22.

Table 22 Big Tobacco SVM Results.

Test	Total Classes	Training Items	Testing Items	F1 Score	Accuracy (%)
1	10	1200	400	58.74	59.25

4.2.3 Custom Model

The results for the Custom Model on Dialexy's documents can be found below in Table 23.

Table 23 Dialexy Data Custom Model Results

Test	Language	Minimum Docs	Total Classes	Training Docs	Testing Docs	F1 Score	Accuracy (%)
1	English	8	8	119	32	68.04	65.63
2	Spanish	8	6	158	24	90.65	83.33
3	English	10	5	100	25	71.43	72.00
4	Spanish	10	5	148	25	95.96	96.00
5	English	20	3	83	15	52.38	60.00
6	Spanish	20	4	141	20	94.95	95.00
	Average					78.90	78.66

Confusion Matrices for Test 3 (Table 24) and Test 4 (Table 25) can be found below.

Table 24 Confusion Matrix for Test 3.

	Academic Transcript	Birth Certificate	Degree Certificate	No Impede Certificate	Payslip
Academic Transcript	4			1	
Birth Certificate		5			
Degree Certificate	5				
No Impede Certificate				5	
Payslip					5

Table 25 Confusion Matrix for Test 3.

	Academic Transcript	Birth Certificate	Criminal Record	Degree Certificate	Payment Receipt
Academic Transcript	4			1	
Birth Certificate		5			
Criminal Record			5		
Degree Certificate				5	
Payment Receipt					5

The results for the Custom Model on 10 classes of the Big Tobacco dataset can be found below in Table 26.

Table 26 Big Tobacco Custom Model Results.

Test	Total Classes	Training Items	Testing Items	F1 Score	Accuracy (%)
1	10	1200	400	56.93	54.75

5 Analysis and Discussion

In this Chapter the results of the experiments will be analysed and discussed, with regard to their performance and potential usefulness in the company's implementation. It will first look at the CNN test results (Chapter 5.1) before examining the text classification results (Chapter 5.2). It will finally briefly summarise and compare the results (Chapter 5.3).

5.1.1 Analysis of Document Image Processing Results

In total 10 experiments were run on different variations of the CNN, investigating the effect a number of different techniques would have on a model's accuracy in order to try and identify which settings would work best for the company.

The first experiment in Section 4.1.1 investigated the effect the image size can have on the accuracy of the document. The first experiment's results clearly show that the larger the input dimensions of the image, the better the model can predict accurately. 32 by 32 pixel images achieved an accuracy of 71.73% over 10 classes, which is still fairly accurate considering this is only taking into consideration 1024 pixels. It is perhaps surprising that a CNN can still detect features with so little information being presented to it. The increase from 64 to 128 pixels yielded an increase in accuracy of 8.14% which is very high compared to the rest of the size increases which only resulted in accuracy rising by 2% or less. It is possible that a certain feature becomes detectable in those dimensions. In this experiment 256 by 256 images performed the best, so any extra noise created at this higher dimension is offset by the model being able to detect more features. The 256 pixel dimension experiments took a lot longer to run (averaging around 23 hours) compared to the other 3 (which each averaged 10 hours, 2.5 hours and 1 hour).

In the same experiment carried out on Dialexy's data, the 256 pixel images were 1.66% less accurate than the 128 pixel images which were the most accurate. It is possible that at this higher dimension and with fewer documents to train on, then the model is picking up a little more noise than it is at a lower dimension. On the opposite end of the scale, 32 pixel documents only scored 43.33% which is a big decrease from the equivalent test in the tobacco dataset. This is unusual as the 64 and 128 seemed to perform more or less the same (although they were a 3 class experiment instead of 10). It is possible that as Dialexy’s classes contain a more diverse range of documents then the CNN may need to be able to detect certain logos and titles which are not detectable at 32 pixels. Documents in the tobacco dataset however may be detectable through the document structure at this lower dimension.

In Section 4.1.2 two different architectures were explored to ensure that the customised model being used performs well against other known architectures. To evaluate this, the results of the model were compared against a custom built lasagne implementation of He et al’s (2016), 34 layer model with Projection Shortcuts (Test 1). In this experiment the deeper Lasagne 42 (Test 2) layer model performed marginally better with an accuracy of 83.07% compared to 81.07%. These were quite large tests with 750 test items, so an extra 2% accuracy is an extra 15 correctly predicted items. It is highly probable that the extra accuracy gained derives from the extra depth of the network provided by an extra 8 layers. The Test 1 model had 64 filters of 7 * 7 dimensions, passing over the input image compared to 16 filters of 3*3. It would perhaps be expected that having more filters at the start gives the model the ability to detect more features, as each will optimise to contain its own set of weights. However, it is possible that the smaller filters of 3*3 are combined together further along in the architecture to make features of larger dimensions. This smaller number of filters at the start also leads to another drawback of the 34 layer model in that it has 27,480,266 parameters compared to the 42 layer model’s 1,867,898. This makes the model a lot faster to train as there are less parameters to optimise. Therefore it makes sense to use the architecture from Test 2 in the rest of the experiments and there is a strong case for it to be used on the company’s platform. Especially as the model from Test 2 beat all other architectures it was compared against in He et al (2016).

The following experiment in Section 4.1.3 investigated the effect caused by splitting the documents into 4 regions and testing each region individually. All 3 tests produced similar results, within 2.4% of one another and neither of the models set up for the split documents performed better than the baseline which used the whole un-split document. This indicates that splitting documents in this manner does not provide any benefit to the prediction accuracy. Also Test 2 which trained a CNN separately on each region performed worse than the CNN in Test 3 where one CNN was trained and tested on all 4 regions. This is perhaps unexpected as it should be more likely that each of the separate CNNs in Test 2 could better ‘specialise’ on identifying the features each document class contains in a specific region.

However, it is perhaps possible that Test 3 was able to better detect the textures and patterns common across the document as a whole. This combined with having a training set 4 times the size as the CNNs in Test 2 may have made it perform better.

It is possible that the whole document test had a higher accuracy than the regional tests as during the splitting process some features such as headings or columns may end up spread across 2 separate documents, making them harder to detect. The whole document CNN however, is able to see the full layout of the document and may detect these features.

None of the 4 individual regions performed better independently than they did when their predictions were combined as seen in Table 6. Therefore, the extra information contained in the whole document is required for a CNN to make a more accurate prediction. The top left region of the document was the most accurate of the 4 and looking through the documents tested, this area contains a lot of unique high level features such as logos and addresses, which would explain this.

In Section 4.1.4 the experiment results demonstrated that transfer learning can have a positive effect on the model as the weights from another model did increase accuracy. However, due to the small amount of testing data it is hard to gauge how effective it can actually be. The increase of 1.66% corresponds to merely predicting one extra test item correctly. Nevertheless, the result does indicate that transfer learning is very much a viable alternative to random weight initialisation and as the transferred weights can be reloaded each time from a file, it does not cost any extra time to use transfer learning each time the model is re-trained in its business implementation.

In Section 4.1.5 the use of synthetic and augmented data was investigated. The results in the first experiment in Table 8 showed that artificially enlarging a dataset can increase accuracy, as each experiment using augmented data performed better than the baseline test which did not use any. Creating 9 augmented copies of each document increased accuracy by 1.6% and copying them 45 times increased it by 14%. However, using a document 9 times in the training set increased accuracy by 12.8%, which is significantly higher than the result with the 9 augmented copies. This is unusual as it would be expected that using the same unchanged item in the training set more than once would lead to overfitting. It is perhaps possible that the amount of noise being added through augmentations to prevent overfitting, actually prevents the model from learning to detect some of the features. However, when copies were increased to 45 or 60 the accuracy of the exact copies fell, so it does begin to over fit yet the accuracy with augmented copies continues to rise.

The other experiment with results in Table 9 demonstrated that a CNN that has been trained on upright images struggles to identify documents of the same class after they have been rotated. It only managed to predict the class of 21.06% of documents correctly. However CNNs can be made more resistant to rotated images by training them on a training set that contains rotations. When this was done accuracy increased to 70.53% which is a very big improvement. Nevertheless, it seems that a CNN which is going to be exposed to rotated images such as Dialexy's will not predict as accurately as one exposed to upright images only. This is theorised because the CNN with identical parameters used in Chapter 4.1.1 still had a larger accuracy of 81.87% across the same 10 classes. It is possible that this drop in performance is due to the large number of features a CNN must learn if it has to learn many instances of the same feature but at different angles.

The experiments in Section 4.1.6 attempted to discover the best means to adapt a CNN so that it would successfully detect images that did not belong to any class. On both datasets being tested on unseen documents, the results shown in Tables 10 and 11 demonstrate that training an 'other' class increased the CNN's performance more than implementing the DOC threshold. In Experiment 1 the 'Other' class was 7.5% more accurate than the best DOC performing threshold and in Experiment 2 it was 4% more accurate. In the confusion matrix in Table 12, it can be seen that the for the 'Other' class test on the big tobacco dataset, a large number of the incorrect predictions (38) were caused by the unseen class being falsely classified as seen classes. However, as shown in the DOC predictor confusion matrix for the Big Tobacco experiment shown in Table 13, 62 incorrect predictions were caused by seen classes being falsely identified as the unseen class. It is possible that the 'Other' class may be better suited for the company as it correctly predicted 197 of the 'seen' documents, compared to the DOC class which only correctly identified 170. Maximising the accuracy of the known documents but allowing some unseen documents to go through incorrectly as a known category could be more beneficial to the company. That way they can maximise the correct predictions of the main types of document that customers will upload.

In the final experiment in Section 4.1.6 when the ‘other’ class was exposed to images from a completely different domain, it only seemed to direct all of them into one of the classes at random which resulted in two scores of 0% and a score of 100%. As there are no similarities at all between flower photographs and documents, it seems that the CNN did not detect any features at all, so it was not able to ‘understand’ the link between the ‘other’ class and the pictures of flowers as would be desired. On the other hand although results deviated a lot, DOC performed well here averaging 74.46% when no minimum threshold is implemented. Therefore DOC has the advantage over the ‘other’ class that it does need to have been exposed to documents from a particular domain, in order to be able to detect non-classes. This could be a successful way of detecting images from unseen classes that are from completely different domains.

5.1.2 Analysis of Text Classification Results

In total 3 different models were applied to document text classification of the company’s documents and the Big Tobacco dataset to see how they compared. These models were Naive Bayes, SVM and a custom model which searched for the main key words.

With regard to the Big Tobacco dataset, the best performing model was SVM with an accuracy of 59.25% and an F1 Score of 56.93. As this was over 10 classes and on a dataset that was created for visual classification instead of textual, this indicates that SVM can be a powerful classifier of document types through their text. The Custom Model performed only marginally worse with an accuracy 54.75% and an F1 Score of 56.93. For such a simple model, these results are also promising and it indicates that you can classify many documents by only using their most common words (7 in this case), it is not necessary to use them all.
NB was the worst performer only scoring an accuracy of 47% and an F1 Score of 49.65. This would perhaps be expected from quite a simple classifier, however the literature indicated that NB can perform well, but it was 12.25% less accurate on these documents than SVM. It would perhaps have benefited from only being run on words that occurred more than a certain number of times, instead of the whole dataset (minus stop words).

In the experiments for the company’s data, 3 different tests were run for 2 different languages for each model. This was to get an indication of which models can perform well with differing levels of training data.

The overall average results for each model across the 6 experiments deviated a lot less than their results for the text classification of the Tobacco set's documents. Of the 3 models, SVM again performed the best, averaging an accuracy of 84.26% and an F1 Score of 81.62. However despite these results it performed worse than the other 2 models in each of the Spanish experiments although it made up for this by outperforming them in English. Examining the confusion matrices in Tables 16, 20 and 24, SVM mostly managed to differentiate between Academic Transcripts and Degree Certificates in English, which were classes the other 2 models confused in every experiment. These documents are very similar textually but SVM was mostly able to separate them with a hyperplane using high dimensional word count vectors.

The Custom Model was the second best performer with an average accuracy of 78.66% and F1 Score of 78.90 which was better than NBs average accuracy of 74.83% and F1 Score of 74.25. Across the experiments the Custom Model performed better than NB in all but 1 experiment and was the best performer in Spanish overall. Its simplicity seemed to allow it to detect the Spanish document classes easily, perhaps the most common words in each document make the class easy to identify but when more words are taken into consideration, the word frequencies become less distinguishable between classes. Both NB and the Custom Model made classification errors in English as some of the documents were very similar and a more complex model is required to be able to distinguish between them.

5.2 Summary of Results

Both models generally performed well considering the circumstances regarding limitations on the amount of training data and processing power. For the CNN transfer learning, augmented data and increased image size all seemed to positively affect the models performance and indicate that they would be good choices when creating a final model for the company. Training the model using rotated images is also important if the CNN will be exposed to them on the platform. The best method to identify unseen classes however was less clear and a decision will have to be made by the company as to which method to implement. Whichever implementation is chosen, it is likely that the accuracy is going to suffer as a result. For the text classification, SVM was the best performing classification method and its results (particularly in English) were very promising.

It was originally planned for the Text and Image models performances to be compared but it does not seem that this would be fair because of the documents each one was exposed to. The document image experiments with company data had documents from many different countries within the same class, each with their own completely distinct structure. The text classification models on the other hand had a less difficult task as the documents had already been split down into their respective languages. Within each language the documents are sometimes different visually, depending on the year of issue for example, but the text does not change all that much.

It is possible to deduce that the CNN performed better than the Text Model on the Big Tobacco dataset. This is based on comparing the best results from the augmented data experiments (68.80% accuracy on 50 Documents of each class) with the SVM model (59.25% accuracy on 160 Documents of each class). This dataset however was created for the image processing domain and there were some documents which contained very few words.

In terms of business implementation, with the OCR tool and a language detector, a word model could perform well for the business but much more training data is required as some languages have too little training data to even create a model. Creating synthetic data for word models is also not really an option. If a model were to be used, it would be SVM as this performed the best, however a much bigger investigation into why it performs less well in Spanish would be needed. It is very probable that this could be improved.

With this in mind, the best choice of a classification model for the company to use at the moment would be the CNN. Although it had a ‘tougher’ job than the text classifier, the results from the experiments were promising. The tests carried out on the Big Tobacco dataset indicated that the architecture is very accurate at predicting document classes when there is sufficient training data. Without this extra training data it is of course a lot more difficult, however there are some techniques that should allow for the CNN to be useable until more training data can be sourced.

6 Conclusion

6.1 Areas for Further Work

The results of the experiments did give a good insight into ways that the company's CNN can be optimised in its business implementation and also showed that text classification was a strong alternative. However, there are two things which were not resolved by the experiments which would merit further investigation as they would be important to improving this project or any others which are similar.

The first and suggestion would be to further investigate Open Set Recognition as both the methods used in experiments performed worse than expected. Even on the best performing experiment, 25.54% of flower images were classified as a document class. Considering how visually distinct a flower image is from a document, it is unusual that there were so many misclassifications. Therefore further investigation into a way of identifying previously unseen classes within image processing would be useful for many projects where models will be exposed to unseen classes.

The other suggestion would be to see what extent augmented data can be used to train Machine Learning models. As augmented copies of the dataset were increased, accuracy continued to increase, until limitations on processing power meant carrying out further experiments would take too long. It would have been interesting to see at what point creating more augmented copies of the same image, would decrease accuracy instead due to overfitting. It must be noted however that these successful results were probably partly due to the similarities that exist between document classes. It is unlikely that creating so many augmented copies would work with images in a more varied domain. Also the effect on performance would depend on which augmentations are used as its possible some could decrease accuracy.

6.2 Conclusion

This projects aim was to investigate models and machine learning techniques which Dialexy could potentially use to classify documents on their platform. The results indicated that there is definitely potential for a model to be created which will perform successfully despite the lack of training data. Both text classification and CNN models displayed results that indicated that they would perform well in the company's implementation, but the CNN seemed a more robust choice as it does not require language detection beforehand and performance can be boosted through augmented copies. For the final 11 class model (10 classes + 1 unseen) that the company is looking to implement in August 2018, there are certainly some strong recommendations from the results of the experiments. The model should be a CNN that uses an image input size of either 128 or 256 pixel dimensions and the input should not be split into regions. Transfer Learning may increase accuracy and should perform no worse than randomised initial weights. Augmented copies should be used while training data is limited as they can boost performance. Also, rotated copies in the training data are necessary to make the model better able to classify rotated images. For open set recognition, having an 'other' class seemed to be the best option, as the uploaded files will all be documents and it was the best performer in the unseen document class experiments. This is likely to cause a drop in accuracy for all of the 'seen' classes but the results indicate this is the best option of the methods tested.

Despite this downside, if this is all implemented then this should give the company a model which can perform successfully on their platform and should open the door to being able to implement some intelligent document processing methods.

7 References

Afzal, M., Capobianco, S., Malik, M., Marinai, S., Breuel, T., Dengel, A., & Liwicki, M. (2015). Deepdocclassifier: Document classification with deep convolutional neural network. *Document Analysis and Recognition*, pp.1111-1115.

Bai, J., & Nie, J. (2004). Using language models for text classification.

Bendale, A., & Boult, T. (2015). Towards Open Set Deep Networks.

Brownlee, J. (2016, April 11). Naive Bayes for Machine Learning. Retrieved July 14, 2018 from https://machinelearningmastery.com/naive-bayes-for-machine-learning/.

Brownlee, J. (2017, October 9). A Gentle Introduction to the Bag-of-Words Model. Retrieved 12 July from https://machinelearningmastery.com/gentle-introduction-bag-words-model/.

Butnaru, A., & Ionescu, R. (2017). From Image to Text Classification: A Novel Approach based on Clustering Word Embeddings. *Procedia Computer Science*, Vol.112, pp.1783-1792.

Cesarini, F., Lastri, M., Marinai, S., & Soda, G. (2001) Encoding of modified X–Y trees for document classification. *Proceedings of the 6th International Conference on Document Analysis and Recognition*. pp. 1131– 1136.

Chatterjee, S. (2017, December 20). Different Kinds of Convolutional Filters. Retrieved June 6, 2018 from https://www.saama.com/blog/different-kinds-convolutional-filters/.

Chen, N., & Blostein, D. (2007). A survey of document image classification: problem statement, classifier architecture and performance evaluation. *International Journal of Document Analysis and Recognition* (IJDAR), Vol.10(1), pp.1-16.

Copeland, M. (2016, July 29). What’s the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?. Retrieved June 30, 2018 from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/.

Das, A., Roy, S., & Bhattacharya, U. (2018). Document Image Classification with Intra Domain Transfer Learning.

Deshpande, A. (2016, July 29). A Beginner’s Guide To Understanding Convolutional Neural Networks Part 2. Retrieved June 6, 2018 from https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/.

Dertat, A. (2017, November 8). Applied Learning – Part 4: Convolutional Neural Networks. Retrieved June 6, 2018 from https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2.

Dialexy. (n.d). DIALEXY. Retrieved May 29, 2018 from http://www.thisiscodebase.com/dialexy/.

Doukkali, F. (2017, October 20). Batch normalization in Neural Networks. Retrieved 19 July from https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c.

Dumais, S., Platt, J., Heckerman, D., & Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. *Proceedings of the seventh international conference on Information and knowledge management.* pp. 148-155.

Galeone, P. (2017, January 10). Analysis of Dropout. Retrieved 19 July from https://pgaleone.eu/deep-learning/regularization/2017/01/10/anaysis-of-dropout/.

Garris, M. (1991). NIST Structured Forms Reference Set of Binary Images. Retrieved June 12, 2018 from https://www.nist.gov/srd/nist-special-database-2.

Gong, Y., Wang, L., Guo, R., & Lazebnik, S. (2014). Multi-scale Orderless Pooling of Deep Convolutional Activation Features.

Harley, A., Ufkes, A., & Derpanis, K. (2015). Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *The IEEE Conference on ComputerVision and Pattern Recognition (CVPR).*

Hulstaert, L. (2018, January 19). Transfer Learning: Leverage Insights from Big Data. Retrieved June 30, 2018 from https://www.datacamp.com/community/tutorials/transfer-learning.

ImageMagick. (n.d.). Imagemagick. Retrieved 12 July from https://www.imagemagick.org/script/index.php.

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition.

Jain, S. (2018, April 19). An Overview of Regularization Techniques in Deep Learning (with Python code). Retrieved 25 July from https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/.

Kang, L., Kumar, J., Peng, Y., Li, Y., & Doermann, D. (2014). Convolutional Neural Networks for Document Image Classification. *22nd International Conference on Pattern Recognition*, pp.3168-3172.

Kannan, S., & Gurusamy, V. (2014). Preprocessing Techniques for Text Mining.

Kölsch, A., Afzal, M., Ebbecke, M., & Liwicki, M. (2017). Real-Time Document Image Classification using Deep CNN and Extreme Learning Machines.

Ko, Y., Park, J., & Seo, J. (2004). Improving text categorization using the importance of sentences. *Information Processing & Management*, Vol. 40(1), pp. 65-79.
34. Rei, M. http://www.marekrei.com/blog/theano-tutorial/.

Kowalczyk, A. (2014). SVM - Understanding the math - Part 2. Retrieved 25 July from https://www.svm-tutorial.com/2014/11/svm-understanding-math-part-2/.

Kowalczyk, A. (2017). *Support Vector Machines Succinctly*. Retrieved from https://www.syncfusion.com/ebooks/support_vector_machines_succinctly.

Kristiadis, A. (2016, July 4). Implementing BatchNorm in Neural Net. Retrieved 25 July from https://wiseodd.github.io/techblog/2016/07/04/batchnorm/.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, Vol.60(6), pp.84-90.

Lasagne. (n.d.). Convolutional Layers. Retrieved 12 July from http://lasagne.readthedocs.io/en/latest/modules/layers/conv.html

Le, T., Baydin, A., Zinkov, R., & Wood, F. (2017). Using Synthetic Data to Train Neural Networks is Model-Based Reasoning.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 439–440.
Lecun, Y., Bottou, L., Bengio, Y & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, Vol.86(11), pp.2278-2324.

Lewis, D., Agam, D., Argamon, S., Frieder, O., Grossman, D., & Heard, J. “Building a test collection for complex document information processing,” vol. 2006, 2006, pp. 665–666.

Li, Y., & Jain, A. (1998). Classification of text documents. *Proceedings. Fourteenth International Conference on Pattern Recognition.* pp. 1295-1297.

Mancini, J. (2016, July 11). What The Heck is Intelligent Document Recognition. Retrieved June 6, 2018 from http://info.aiim.org/digital-landfill/what-the-heck-is-intelligent-document-recognition.

Mayo, M. (n.d.). Natural Language Processing Key Terms, Explained. Retrieved July 14, 2018 from https://www.kdnuggets.com/2017/02/natural-language-processing-key-terms-explained.html

McCaffrey, J. (2017, June 29). Implementing Neural Network L2 Regularization. Retrieved 25 July from https://jamesmccaffrey.wordpress.com/2017/06/29/implementing-neural-network-l2-regularization/.

Muatik. (n.d.). Naive-Bayes-Classifier. Retrieved 12 July from https://github.com/muatik/naive-bayes-classifier

McCaffrey, J. (2013, 5 November). Why You Should Use Cross-Entropy Error Instead Of Classification Error Or Mean Squared Error For Neural Network Classifier Training.

Retrieved 19 July from https://jamesmccaffrey.wordpress.com/2013/11/05/why-you-should-use-cross-entropy-error-instead-of-classification-error-or-mean-squared-error-for-neural-network-classifier-training/.

McCaffrey, J. (2017, June 6). Neural Network Momentum. Retrieved 19 July from https://jamesmccaffrey.wordpress.com/2017/06/06/neural-network-momentum/.

Mamaev, A. (2018). Flowers Recognition. Retrieved 19 July from https://www.kaggle.com/alxmamaev/flowers-recognition/version/2.

Otter, C. (2017, September 16). Loss Functions In Deep Learning. Retrieved 19 July from http://yeephycho.github.io/2017/09/16/Loss-Functions-In-Deep-Learning/.

Pal, M. (2008). Multiclass Approaches for Support Vector Machine Based Land Cover Classification.

Ray, S. (2017, September 9). Essentials of Machine Learning Algorithms (with Python and R Codes). Retrieved June 30, 2018 from https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/.

Ruder, S. (2016, January 16). An overview of gradient descent optimization algorithms. Retrieved 19 July from http://ruder.io/optimizing-gradient-descent/index.html#momentum.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg., A & Fei-Fei, L. (2012). ImageNet Large Scale Visual Recognition 2012.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg., A & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition 2015. *International Journal of Computer Vision (IJCV)*, pp. 1–42.

SAS. (n.d.). what-is-natural-language-processing-nlp. Retrieved 12 July from https://www.sas.com/en_gb/insights/analytics/what-is-natural-language-processing-nlp.html.

Scheirer, W. (2018). Open Set Recognition. Retrieved 12 July from https://www.wjscheirer.com/projects/openset-recognition/.

Scheirer, W., Rocha, A., Sapkota, A., & Boult, E. (2013). Toward Open Set Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence.* Vol.35(7), pp.1757-1772.

Shu, L., Xu, H., & Liu, B. (2017). DOC: Deep Open Classification of Text Documents.

Simonite, T. (2018, April 25). Some Startups Use Fake Data to Train AI. Retrieved June 16, 2018 from https://www.wired.com/story/some-startups-use-fake-data-to-train-ai/.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

SoftWorks AI. (n.d.). Automated Forms Processing. Retrieved June 6, 2018 from https://www.softworksai.com/our-solutions/automated-forms-processing.

Stanford Vision and Learning Lab. (n.d.). Setting up the data and the model. Retrieved 25 July from http://cs231n.github.io/neural-networks-2/#reg.

Stanford University. (n.d.). Text Classification and Naïve Bayes [PowerPoint slides]. Retrieved from https://web.stanford.edu/~jurafsky/slp3/slides/7_NB.pdf.

Stanford Vision and Learning Lab. (n.d.). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved June 6, 2018 from http://cs231n.github.io/convolutional-networks/.

Stecanella, B. (2017, May 25). A practical explanation of a Naive Bayes classifier. Retrieved 19 July from https://monkeylearn.com/blog/practical-explanation-naive-bayes-classifier/.

Stecanella, B. (2017, June 22). An introduction to Support Vector Machines (SVM). Retrieved 25 July from https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/.

Stone, B., Cheryl, S., Pammer, C., Steele, C., & Keller D. (2015, September 3). The Danger of Overfitting Regression Models. Retrieved June 30, 2018 from http://blog.minitab.com/blog/adventures-in-statistics-2/the-danger-of-overfitting-regression-models.

Tensmeyer, C., & Martinez, T. (2017). Analysis of Convolutional Neural Networks for Document Image Classification.

Theano. (n.d.). Welcome. Retrieved 12 July from http://deeplearning.net/software/theano.

Theano. (2017, November 21). Ops for neural networks. Retrieved 12 July from http://deeplearning.net/software/theano/library/tensor/nnet/nnet.html#theano.tensor.nnet.nnet.categorical_crossentropy.

Ujjwalkarn. (2016, August 11). An intuitive Explanation of Convolutional Neural Networks. Retrieved June 6, 2018 from https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

8 Appendices

8.1 Project Proposal

EDINBURGH NAPIER UNIVERSITY SCHOOL OF COMPUTING
MSc RESEARCH PROPOSAL

The process of completing and reviewing the contents of this form is intended ensure that the proposed project is viable. It is also intended to increase the chances of a good pass. Much of the material produced while completing this form may be reused in the dissertation itself.

1. **Student details**

First name	Stephen
Last (family) name	Charlton
Napier matriculation number	40331195

2. **Details of your programme of study**

MSc Programme title	Computing
Year that you started your diploma modules	2017
Month that you started your diploma modules	September
Mode of study of diploma modules	Full-time
Date that you completed/will complete your diploma modules at Napier	2018

3. **Project outline details**

Please suggest a title for your proposed project. If you have worked with a supervisor on this proposal, please provide the name. You are strongly advised to work with a member of staff when putting your proposal together.

Title of the proposed project	Developing Models for Automated Document Classification
Is your project appropriate to your programme of study?	Yes
Name of supervisor	Dimitra Gkatzia
I do not have a member of staff lined up to supervise my work	

4. **Brief description of the research area - background**

Please do not describe your project in this section. Instead, provide background information in the box below on the broad research area in which your project sits. You should write in narrative (not bullet points). The academic/theoretical basis of your description of the research area should be evident through the use of citations and references. Your description should be between half and one page in length.

The idea behind the proposal derives from a work placement at Dialexy who specialise in “the development of software solutions to improve and automate the

certified translation process for certified translators and their clients" (Dialexy, n.d.).
The project falls within the field of 'Intelligent Document Recognition', where on receiving an image of a document a model can accurately predict which of a set of previously seen templates, it is most similar to.
As well as correctly identifying the document type, it would also be of interest to Dialexy for a means to detect which of all the documents it already contains within its database is the most 'textually' similar, as this may be able to help automate the translation process.

If all of this could be successfully done, then the models would facilitate the translation process by providing the translator with the correct document structure to translate onto and could even lead to the auto-prediction of some sentence translations.

As the company grows, more users will upload their documents to the company's website for translation. These documents can be used to further optimize, train and develop the document recognition models.

There are examples of research into the Intelligent Document Recognition field in literature and they can be roughly split into two categories. Models which identify the document type through the text it contains using Optical Character Recognition tools and Natural Language Processing techniques. The second identifies the document type using image processing techniques.

For image processing techniques, convolutional neural networks have been successfully applied to the detection and recognition of objects in images since the early 2000s (LeCun, Bengio & Hinton, 2015, p. 440). Since their impressive performance in the 2012 ImageNet Competition, they are the most used approach for image recognition tasks, which has been facilitated by advances in software and hardware which allows faster processing and training (LeCun et al., 2015, p. 440).
In Harley, Ufkes and Derpanis (2015), a CNN is successfully trained to classify documents through their images which performed better than state of the art 'bag of word' methods. The study also trained and tested five different neural networks each focusing on specific regions of the documents and one more focusing on the whole document. The results indicated that on a large dataset, classifying the image based on the whole image seemed to perform better than focusing on a particular region (Harley et al., 2015).

For matching document texts in literature, string matching techniques have been used to compare the text in documents. For example in Lopresti (1999), four different adaptions of the approximate string matching algorithm were used to successfully detect duplicate documents in a database.

5. **<u>Project outline for the work that you propose to complete</u>**
Please complete the project outline in the box below. You should use the emboldened text as a framework. Your project outline should be between half and one page in length.

In collaboration with the company Dialexy, the project is to develop two models. The aim of the first is to accurately and automatically classify a document's type, from a set of stored document templates. This would be split into two parts; it will firstly attempt to classify the document's image using a pre-trained convolutional neural network for image processing. If the highest classification probability is not above a certain threshold, then the model will attempt to classify it in a different way. This will be done through the document text, which will be read using an OCR tool. This part will comprise of two parts, the first will attempt to correct any errors in the OCR's output, which may have been caused by hand written documents etc. The second part will attempt to match the document by using a model which detects key words in the text which will attempt to classify it as a certain document category.
The project will involve adapting and training Dialexy's convolutional neural network for the document recognition. Neural Networks can be difficult to train and encounter problems such as vanishing gradients and overfitting amongst many more. This is further complicated as Dialexy does not have a large amount of training data. The aim of the project would be to improve on the model and training process in order to increase its accuracy in its business implementation.
This will be done by attempting to train the network using the data Dialexy currently have. It is possible to increase training data by 'mining' for some document types which can be found online through Google's 'Search by Image' functionality. Also, as most documents have the same structure, it is possible to remove or add text to different training documents. This will both give us the ability to create new 'synthetic' data and also test whether 'noise' such as names and dates have a negative effect on the models accuracy.
Synthetic data has been used to train and optimize Machine Learning Models in literature. For example in Jaderberg, Simonyan, Vedaldi and Zisserman (2014), a convolutional neural network was trained to detect text in 'natural scenes', using only synthetically created data.
The model will also be evaluated using the RVL-CDIP collection (Harley et al., 2015), which is a dataset of documents pre-labelled with their categories. This will be relatively similar to the documents Dialexy will have when their database but has many more documents, so can be used to test the accuracy of the model being implemented. It may also give indications of improvements that can be made to train the business model with its current levels of training data. The RVL-CDIP collection (Harley et al., 2015) is made up of 15 different types of document such as letters, emails, resume's and invoices. Each of these types is structurally and textually quite distinct which would allow the models to detect the same distinctive patterns they should find in Dialexy's data.

The second model will also use the OCR tool to detect the document type through its text so that the platform can identify the textually 'most similar' document to the one that has been fed in from all the documents stored in its database (instead of just identifying the document type). This will involve implementing an advanced string matching technique, which will very quickly be able to compare lots of document's texts, as there could potentially be thousands.
It will be evaluated using both the company's training data and also the RVL-CDIP collection, where both accuracy and speed will be important to its performance in the

business scenario.

The project will involve writing in python and using its various libraries. The neural network will be written using Lasagne which uses Theano.

6. **References**

Please supply details of all the material that you have referenced in sections 4 and 5 above. You should include at least three references, and these should be to high quality sources such as refereed journal and conference papers, standards or white papers. Please ensure that you use a standardised referencing style for the presentation of your references, e.g. APA, as outlined in the yellow booklet available from the School of Computing office and http://www.soc.napier.ac.uk/~cs104/mscdiss/moodlemirror/d2/2005_hall_referencing.pdf .

1. Dialexy. (n.d). DIALEXY. Retrieved May 29, 2018 from http://www.thisiscodebase.com/dialexy/.
2. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 439–440.
3. Harley, A., Ufkes, A., & Derpanis, K. (2015). Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval.
4. Lopresti, D. (1999). String techniques for detecting duplicates in document databases. *International Journal on Document Analysis and Recognition,* 2(4), 186-199.
5. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A. (2014). Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition.

7. **Ethics**

If your research involves other people, privacy or controversial research there may be ethical issues to consider (please see the information on the module website). If the answer below is YES then you need to complete a research Ethics and Governance Approval form, available on the website: http://www.ethics.napier.ac.uk .

Does this project have any ethical or governance issues related to working with, studying or observing other people? (YES/NO)	NO

8. **Confidentiality**

If your research is being done in conjunction with an outside firm or organisation, there may be issues of confidentiality or intellectual property.

Does this project have any issues of confidentiality or intellectual property? (YES/NO)	YES

9. **Supervision timescale**

Please indicate the mode of supervision that you are anticipating. If you expect to be away from the university during the supervision period and may need remote supervision please indicate.

Weekly meetings over 1 trimester	✔

Meetings every other week over 2 trimesters	
Other (what?)	

10. **Submitting your proposal**
    1. Please save this file using your surname, e.g. macdonald_proposal.docx, and e-mail it to your supervisor, who will discuss it with you and suggest possible improvements.
    2. When your supervisor is content with your proposal, email it to your internal examiner, who will provide feedback and possibly suggestions for improving your idea.
    3. Discuss your feedback from the internal examiner with your supervisor and if necessary make final changes to your proposal.
    4. Upload the final version to the dissertation learning space on Moodle.
    5. When you produce your dissertation, add your finalised proposal as an appendix.