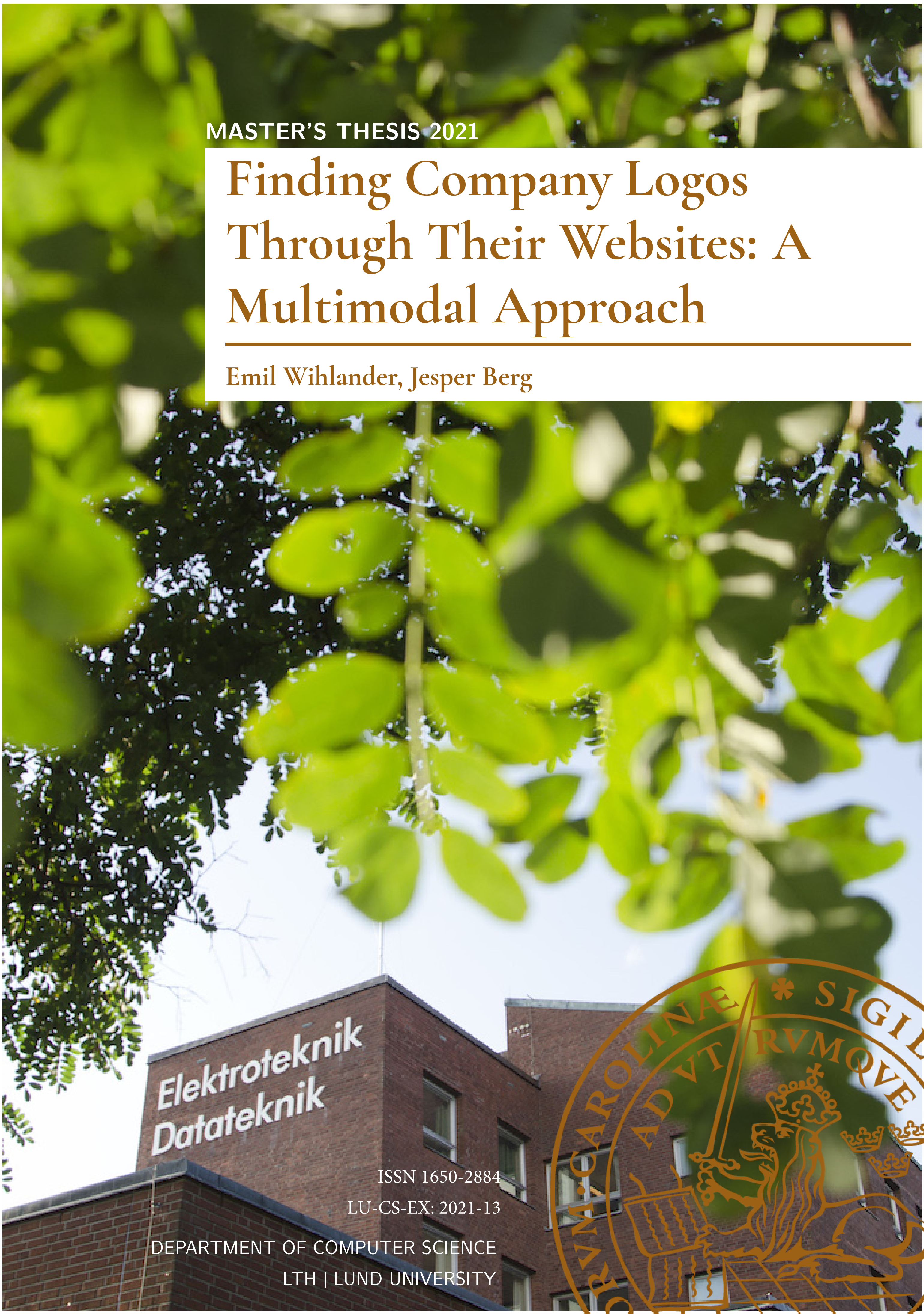
MASTER'S THESIS 2021

Finding Company Logos Through Their Websites: A Multimodal Approach

Emil Wihlander, Jesper Berg



ISSN 1650-2884

LU-CS-EX: 2021-13

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2021-13

Finding Company Logos Through Their Websites: A Multimodal Approach

Att hitta företagsloggor genom deras hemsida: En multimodal lösning

Emil Wihlander, Jesper Berg

Finding Company Logos Through Their Websites: A Multimodal Approach

Emil Wihlander
emil.wihlander.779@student.lu.se

Jesper Berg
jesper.berg.850@student.lu.se

May 20, 2021

Master's thesis work carried out at AFRY Group AB.

Supervisors: Marcus Klang, marcus.klang@cs.lth.se
Frank Camara, frank.camara@afry.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

Retrieving any kind of metadata about a company can be valuable and fetching such information from their website is possible through web-scraping. A good example of this is the logo which has several use-cases in itself. Extracting every image from a site is not that difficult of a task, the obstacle is finding which of all the images is the requested logo.

This thesis presents several different models for classifying if a web-scaped image is the specified company's logo. These models vary from trivial, machine learning text and image classifiers to multimodal solutions which use the previous models. Each model is trained on a diverse data set with companies and websites from multiple countries and cultures. They are thereafter evaluated and compared using a combination of precision, recall and Matthews correlation coefficient (MCC) scores.

Text classification models showed viability in selecting company logos, but image models had problems with context and filtering out logos from other companies or brands. Multimodal methods also had this issue but a weight-based decision-model performed better than any single modal method, with a 6% increase in MCC compared to the best. This indicates multimodality is an encouraging approach to finding company logos.

Keywords: machine learning, multimodality, knowledge engineering, text classification, image recognition, CNN, web-scraping

Acknowledgements

We firstly want to thank AFRY for the opportunity to carry out our thesis with them. Specifically Frank Camara for always making sure we felt welcome, and Rasmus for providing invaluable feedback and knowledge.

We also want to thank Niklas Hedström and Daniel Jannemyr Åhlin for their company and friendship throughout this quarantined time.

Last, but not least, we want to thank Marcus Klang for being a great supervisor, with excellent feedback, encouragement, and directive.

Contents

Chapter 1
Introduction

Retrieving any kind of metadata about a company can be valuable. Since essentially every business, conglomerate to local, has a website, fetching data from that site would be great. Luckily, this is possible through the process of web scraping. As Ryan Mitchell describes it in his book *Web scraping with Python: Collecting more data from the modern web*: "If programming is magic then web scraping is surely a form of wizardry. By writing a simple automated program, you can query web servers, request data, and parse it to extract the information you need" [Mitchell, 2018].

A subproblem where this is applicable is company logos. Usually, one of the first elements to greet a visitor to a website is the logo. This enables it as a good candidate for web-scraping. There are many use-cases where retrieving company logos can be utilized. Examples could be an application which displays information about businesses automatically displaying their logos, or a design tool which shows logos in specific industries as inspiration.

The difficulty in this web-scraping application manifests itself as websites usually has more than one image. Selecting which one of all the images is the logo is simple for humans, but becomes increasingly more difficult and complex for a computer [Lu and Weng, 2007]. It is also important the selected logo is associated to the specific company whose website is scraped, and not some other company or brand's logo.

Text classification and image recognition are two possible of identifying the logo by classifying it as either the logo or not. Both of these techniques are used by a client of AFRY, a Swedish consulting firm, to solve other classification problems. Both methods will be delved into in this thesis in order to get a good baseline for further evaluation of another approach.

This approach is *Multimodality*, the combination of the previously mentioned techniques. In short, multimodality refers to a single domain containing multiple types of data input. In the logo problem, the data inputs are both the image itself and its text source, how it is implemented on the website.

Multimodality provides three main benefits [Baltrušaitis et al., 2018]:

1. Multiple modalities describing the same instance may provide more robust predictions.

2. The multiple modalities may have complementary information, making it possible to extract more information about the same instance.

3. A multimodal model can still work if one modality is not available, for example when the image is available but not the text source of that image or vice versa.

Due to these benefits, the client at AFRY believes using multimodality to identify company logos will increase the performance of their logo detectors. As these are used in a vital part of one of their products, any improvement is welcome and could provide immense value.

To analyse the different approaches, several text classification and image recognition models will be implemented as well as several trivial algorithms. Evaluations on a self-created curated data set, consisting of the text sources and actual images from many websites, will then be performed of these models. To achieve the multimodal approach the previous architectures will then be combined in different constellations, which in turn will also be evaluated. All the models and evaluations will then be compared to each other to reach the results.

The results show that using text classification in general is a viable approach to finding the logo, while the tested image recognition methods were not. Viability indicates both the authors of this thesis and the client at AFRY believe the approach could be used in practice. The main issue in classification methods is selecting the specifically requested logo and not just any logo.

Multimodal approaches showed high potential compared to individual models, but never quite reached it. Most multimodal techniques provided the same performance as the best text classification models. However, using a multimodal weight-based decision-model provided higher classification scores than any single modal method. Therefore, multimodality is shown to be an encouraging approach to finding a company logo through their website.

1.1 Division of Work

This section will describe how the work in this thesis was shared between the authors which will be described as **EW** and **JB**, *Emil Wihlander* and *Jesper Berg* respectively.

The first part of the work was analyzing how images were present on sites, specifically how and what information could be gathered through web-scraping. This was done by both authors as it was integral to the thesis and the approaches which they would later implement.

In general **EW** has worked on the text classification approaches as well as decision-based multimodality, while **JB** has worked on image recognition and model-based multimodality. Therefore follwing the initial web-scraping analysis, the authors researched and evaluated different techniques in their individual fields. At this point, both authors also researched possibilities for handling data sets and planned the annotating tool.

There are two parts of the annotating tool, the web-scraping, which was implemented by **JB**, and the annotating interface, implemented by **EW**. **EW** was then responsible for gathering website information for the companies which were later used in the data set. He was also annotating the final web-scraped images through this tool, while **JB** was working on gathering and annotating screenshots with ground truth boxes for object detection.

Parallel to this, both authors discussed and decided on possible evaluation techniques. Thereafter, **EW** implemented Knowledge engineering, BLSTM and BERT while **JB** implemented

Table 1.1: Distribution of the thesis' writing.

Section	**EW**	**JB**
1		X
1.1		X
1.2	X	X
1.3	X	
1.4		X
2		X
2.1	X	X
2.2	X	
2.3		X
2.4	X	
2.5		X
3	X	
4		X
4.1	X	
4.2	X	
4.3	X	
4.4		X
4.5	X	X
5		X
5.1	X	
5.2		X
5.3	X	X
6		X
6.1	X	X
6.2		X
6.3	X	X
7		X

CNN, Bounding box regression, R-CNN and Detection through classification. The results from all the different methods were discussed and evaluated by both authors.

After these methods had been evaluated, **EW** started working on the two decision-based approaches, policy and weight-based, while **JB** implemented the model-based architecture. Similarly to the previous single modal results, both authors discussed and evaluated the results from all possible models.

The distribution of writing for this thesis can be seen in table 1.1, where the left most column indicate chapters and sections. A single number refers to the text before any section begins, or the entire chapter if no other sections in that chapter follow it.

1.2 Research Questions

The primary goal of this thesis is to find the approach that produces the most accurate result at finding the company logo given a company website. As explained in the introduction, the challenge itself lies in selecting the correct logo from a list of potential candidates.

The research questions therefore are:

- Which text classification methods can be used to identify a company logo from their website and how do they perform?
- Which image recognition methods can be used to identify a company logo from their website and how do they perform?
- Which multimodal methods can be used to identify a company logo through their website and how do they perform compared to the other individual models?

1.3 Related Work

This thesis is based on theory from three main areas: text classification, image recognition and multimodality.

Text classification has many different applications and proposed solutions where earlier implementation focused on word count and knowledge engineering [Sebastiani, 2002] and current implementations focus on neural networks [Devlin et al., 2018, Lan et al., 2019, Clark et al., 2020]. The work done by [Sebastiani, 2002] discuss the viability of knowledge engineering in relation to machine learning and was the basis for including knowledge engineering as a potential solution to text classification in this thesis.

[Hochreiter and Schmidhuber, 1997] proposes a neural network architecture called *Long Short-Term Memory* (LSTM). The network handles context over time better than previous neural networks. The architecture has been shown to be able to solve text classification task [Wang et al., 2018], which was the reason for it to be considered as a potential solution to text classification in this thesis.

In [Vaswani et al., 2017] a new neural network architecture, called TRANSFORMERS, is proposed. It tries to solve the problem with lost context by building relation matrices between the words. This architecture is used in *Bidirectional Encoder Representations from Transformers* (BERT), proposed by [Devlin et al., 2018], which shows great performance for text related tasks, including classification. The performance of BERT spawned multiple variants that promises either smaller models or better performance, or both. This includes ALBERT [Lan et al., 2019], small BERT [Turc et al., 2019] and ELECTRA [Clark et al., 2020]. The performance on text related task was the reason for BERT, and its variants, to be considered as a potential solution to text classification in this thesis.

For image recognition, two methods are explored: image classification and object detection. For image classification, [Shin et al., 2016] shows that pre-trained *convolutional neural networks* (CNN), using transfer learning, can be a successful approach to image classification. The pre-trained CNN that are used in this thesis are VGG [Simonyan and Zisserman, 2014], MobileNet [Sandler et al., 2018], ResNet [He et al., 2016] and DenseNet [Huang et al., 2017].

Object detection is a combination of localization and classification [Wang and Su, 2019] and is an alternative route the pure classification done on both the images and its text implementation. By using a screenshot of the website *bounding box regression* [Lee et al., 2018], *region based convolutional neural networks* [Girshick et al., 2014] and *detection though classification* using *sliding window* [Lampert et al., 2008] or *image pyramid* [Adelson et al., 1984] were employed to try to find and identify the logo.

Multimodal fusion is a field that has grown popular over the last 30 years but still has a lot of standardization to go through, especially with neural networks. First example of successful weight-based late fusion for neural networks was done by [Yuhas et al., 1989] and due to its simplicity, the same method will be used in this thesis. The structure of [Gao et al., 2015] model-based fusion was an inspiration for the model-based architecture used.

[Zahavy et al., 2016] has been our primary inspiration due to similar application, the same modalities (text and image) and proposing fusion methods that are relevant to this work. It does however have multiple labels and the text is natural language compared to this works one label and where the text is a file path. Another paper that has proven helpful is [Baltrušaitis et al., 2018] due to its extensive discussion of all parts of multimodality.

1.4 Outline

Following this introductory chapter, Chapter 2 describes the theory behind the methods used in this thesis as well as the theoretical background necessary to grasp the concepts presented. In Chapter 3, the data set, which was created for this thesis will be explained, including its source and potential problems. Thereafter in Chapter 4, the experimental cycle is presented, as well as the data preprocessing steps, and the training and evaluation pipelines. In Chapter 5, the results for different models are summarized and a selection of them is motivated. Following this, Chapter 6 will discuss the previously presented results, as well as issues in this approach, limitations. Finally, a conclusion and final thoughts will be presented in Chapter 7.

Chapter 2
Theory

To get a better understanding of the concepts and results of this thesis, this chapter will describe the theory behind them. As this thesis has its basis in text classification and image recognition, sections describing these will open this chapter after a short introduction to artificial neural networks. Following this, the idea of multimodality and two approaches to its use is presented. The section will be concluded with a description of the evaluation methods used to assess the results.

2.1 Artificial Neural Network (ANN)

A technique which is useful for both text and image handling is Artificial Neural Network (ANN). This thesis will use different variations of ANN for text classification (section 2.2), image recognition (section 2.3) and multimodal fusion (section 2.4.1). ANN to are heavily inspired by how biological neural networks work where individual nodes, or neurons, collectively learns to produce an output from an input. These neurons are both large in number and interconnectivity. The neurons are distributed in several layers, often described as the input layer, hidden layers and the output layer [O'Shea and Nash, 2015].

A simple example of the ANN structure can be seen in figure 2.1. The input layer transforms the input, often in the form of a multidimensional vector and propagates this further to the first hidden layer. The hidden layers are both responsible to making the decisions and doing the learning [O'Shea and Nash, 2015]. This is based on weights of edges—which is the connections—from the previous layers as well as internal changes which in turn affect the output to be better or worse. Whichever happens, each layer learns if the changes it made were improving or not and can thereafter adjust their behavior and edge-weights to close in on the intended result [O'Shea and Nash, 2015, Zupan, 1994]. The output layer translates the data from the final hidden layer into a result which can easily be processed by humans. For example, it could return a previously defined value which indicates something to the executor, or a percentage of how sure the network is about a specific classification. Several dimensions and values of output is also possible to implement if wanted by adding more neurons in this layer. An example of this is in figure 2.1 where the network produces two values as output for a specific trio of input values [Zupan, 1994]. This thesis will use alteration of this basic ANN into more advanced architectures, especially with more hidden layers, which in turn will be able to classify logos.

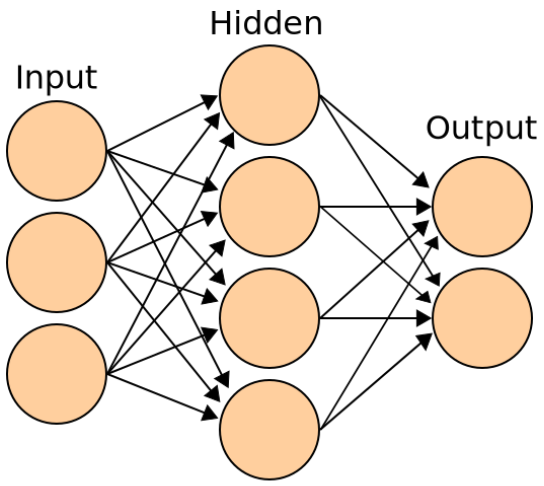


Figure 2.1: An example of a simple ANN structure with only one hidden layer, which can also be called a three-layer ANN.

2.1.1 Machine Learning with ANN

To create an ANN that solves a problem in a domain a large set of examples, called a *data set*, is needed for training the model as well as evaluating its performance. Once a data set is acquired it is split into three pools, *train*, *validation* and *test*. The first two are used for training, while the latter is used for evaluation. The exact splitting ratio used can vary depending on data set size and domain but train is usually the majority of the data set [Shah, 2017].

To train an ANN a training cycle, called an *epoch*, is repeated multiple times. During each epoch the train set is used to try to optimize the edge-weights using an optimization function, *learning rate* defines how large the changes should be [Torres, 2020]. Two examples of optimizers are Adam and AdamW [Ruder, 2016, Loshchilov and Hutter, 2017]. At the end of an epoch the validation set is used to evaluate the performance of the latest edge-weights. The evaluation is done using a loss function which returns a higher number the further away a model is from performing perfectly [Torres, 2020]. An example of a loss function for binary classification is binary crossentropy [Godoy, 2018]. All parameters that can be set before training, such as the ones above, are called *hyperparameters*. The number of epochs that is run is either a specified amount or when a early stopping criteria is met [Brownlee, 2018, Torres, 2020]. A common early stopping criteria is when the loss for the validation set no longer becomes smaller, i.e., it does not perform better than the last epoch [Brownlee, 2018]. Since the training relies on randomness, *patience* can be used to avoid stopping on a temporary stall in loss, for example, a patience of 3 would mean it would stop if the criteria is satisfied three epochs in a row [Brownlee, 2018].

Early stopping is also a good measure against *overfitting*. Overfitting appears when the model performs really well on the train set but much worse on the validation and *test* sets. By stopping when the performance does not get better on the validation set, it increases the chance that the model has found general patterns that are true for all three pools, and not just for the train set [Brownlee, 2018].

After the last epoch the model is evaluated on the test set, using the desired metrics, to get a understanding of its performance on data that has not been used during training at all.

2.1.2 Imbalance

Class imbalance is when there is a large difference in the number of examples of what the model tries to differentiate between. For example, a data set of 5 logos and 95 non-logos. When training ANN, class imbalances can be an issue and degrade their potential performance. This is due to expected equal misclassification cost or equal class distribution in the training algorithms [He and Garcia, 2009]. There are multiple ways to handle class imbalance to improve the learning and in extension the performance of the model. The thesis will look at *oversampling*, *undersampling* and *cost sensitive* methods.

Oversampling This sampling method increases the size of the data set by increasing the number to data points in the minority class. This can either be done through random sampling or by creating synthetic data points based on the existing data points [He and Garcia, 2009]. Oversampling can sometimes lead to overfitting during training but has been shown to be effective for some ANN [Buda et al., 2018]. Random oversampling repeats existing data by continuously adding random data points from the initial minority class set until the desired ratio is reached. There are multiple synthetic methods, and two examples are SMOTE and ADASYN [He and Garcia, 2009]. Both creates new data point generated based on the existing data points. They are however designed to use a set amount of parameters which makes them very difficult to use with variable length sentences requiring complex preprocessing [He and Garcia, 2009]. Therefore, only random oversampling will be tested.

Undersampling This sampling method decreases the size of the data set by removing data points from the majority class. This can either be done through random sampling or informed sampling. A drawback of using undersampling is that relevant data points might be discarded. Informed undersampling tries to compensate for this but might not be successful [He and Garcia, 2009]. Random undersampling removes data points of the majority class at random until the desired ratio is reached. Informed sampling includes multiple methods, but the common denominator is that they use characteristics of the data points to decide which data points to discard. This is complex and out-of-scope, therefore only random undersampling will be tested.

Cost sensitivity Instead of sampling the cost sensitive method can be used during training to improve the performance of the model. This is done by setting different costs for different misclassifications. If r is the cost ratio between the classes, $C_{min \to maj}$ is the cost of mislabeling minority as majority class and $C_{maj \to min}$ vice versa, the equation $C_{min \to maj} \approx r \cdot C_{maj \to min}$ should be satisfied, making it more expensive to mislabel the minority class. It has been shown that this method has a similar performance to oversampling, but requires fewer data points during training, making it faster [Buda et al., 2018, He and Garcia, 2009].

2.1.3 Transfer Learning

A simple way to think about transfer learning is to equate it to a real person's knowledge. For example, if a person knows how to type on a typewriter, it will aid them in learning to write on a keyboard. Or how someone being proficient in chess might be able to use some of their chess knowledge when studying football tactics. In the first example, the two abilities are close in domain and it is therefore easy to understand how they are connected and the knowledge is transferred. The second example however is more subtle where understanding of deeper concepts, such as tempo or opponent weaknesses, could be beneficial knowledge to transfer.

It turns out both ways of passing on knowledge are also possible for ANN [Pan and Yang, 2009]. Therefore, a fully trained classification model can be used to train a new model to classify something the original model cannot, without training it from scratch [Shaha and Pawar, 2018].

This process, called *fine-tuning*, has been showed to produce equal or better results when compared to a new fully trained model on different data sets [Tajbakhsh et al., 2016]. Fine-tuning is implemented by not allowing some layers to re-train, known as *freezing* a layer, while letting others train on a data set containing the new domain. An example of this is freezing layers which extract important information while letting the the final layers learn and adapt to the new data set [Girshick, 2015]. This has the advantage of keeping the well-trained feature extraction while letting the dense neurons learn on those features.

Transfer learning will in this thesis be used in both the text classification method BERT, which will be described in section 2.2.2 and every image recognition approach, described in section 2.3.

2.2 Text Classification

Text classification has since the digitization of documents been an area of interest to automatically group large amount of data. Many different approaches to solve the task has been proposed through-out the years including knowledge engineering and various machine learning methods [Sebastiani, 2002]. This thesis uses knowledge engineering and ANN, more specifically bidirectional long short-term memory (BLSTM) and bidirectional encoder representations from transformers (BERT). All of these has shown to be successful at text classification [Devlin et al., 2018, Sebastiani, 2002, Hochreiter and Schmidhuber, 1997].

2.2.1 Knowledge Engineering

An approach to text classification that predates more modern approaches is knowledge engineering. The approach uses domain knowledge to define a rule set that can be defined through traditional programmed decision trees [Sebastiani, 2002]. More modern machine learning models using methods such deep learning may have higher success rate than knowledge engineering on complex data and domains. But if the data have simpler patterns knowledge engineering still is a viable approach to text classification.

It has been shown that this approach can be successful [Hayes et al., 1990] and the basis of this thesis is that the data set will have some patterns that can be defined with simple rules. These rules can then be combined in different ways to create a better classifier than any single rule. The most basic binary classifier could be to mark every element containing the word *logo* as a logo.

2.2.2 ANN for Text Classification

Another approach to text classification is to use an ANN. State-of-the-art models built on different variations of ANN outperform other approaches in the field of text classification on the vast majority, if not all, of public natural language data sets. Worth noting, this thesis' data set isn't natural language per default but since the work relating to text classification using ANN mostly relates to natural language, the text will be converted into word arrays, see section4.3.1.

A concept that needs to be introduced—in order for ANNs to be able to process text input—is *text vectorization*. Due to ANN being numerical by nature, each word in a text needs to be represented by a numerical value and the approach that is used is *bag-of-words*. Bag-of-words assigns the value 1 to the most common word in the test data, the value 2 to second most common, and so on.

After the vectorization, an embedding layer converts the indexed words into vectors. The idea is to give words with similar meaning a similar value, allowing the following layers to reason based on semantics. The layers after embedding differs based on domain characteristics, complexity and number of classes but two approaches are by either using BLSTM or the transformer network BERT [Hochreiter and Schmidhuber, 1997, Devlin et al., 2018].

Bidirectional Long Short-Term Memory (BLSTM)

An approach that, up until recently, was the industry standard for text classification is the recurrent neural network (RNN) architecture, long short-term memory (LSTM). Ever since it was introduced by [Hochreiter and Schmidhuber, 1997] LSTM has been at the forefront of breakthroughs in machine learning, especially in *natural language processing* (NLP) related fields. It is being used by large corporations such as Google [Sak et al., 2015, Wu et al., 2016], Facebook [Ong, 2017], and Amazon [Vogels, 2016] as well as research-based organizations such as OpenAI [Andrychowicz et al., 2020] and DeepMind [Stanford, 2019].

LSTM was originally developed to handle the vanishing gradient problem that exist in traditional RNN. Vanish gradient means that the context of earlier words are forgotten quickly. This is done by propagating not only the gradient—which is the output of a cell—but the cell state as well, see figure 2.2. In practice, this means that LSTM can handle context over longer periods allowing for more complex patterns to be identified by the neural network.

Since an input text can contain different number of words, text classification is also a possible domain for LSTM due to the possibility to have a variable length input to the model. Also, preceding methods such as bag-of-words did not encode word order loosing potential patterns relating to word ordering [Zhou et al., 2016].

One variation of LSTM is BLSTM. While LSTM feeds the cell output from past words to be able to reason based past context BLSTM extends this by having two LSTMs feeding both backwards and forward. This increases the available information effectively allowing

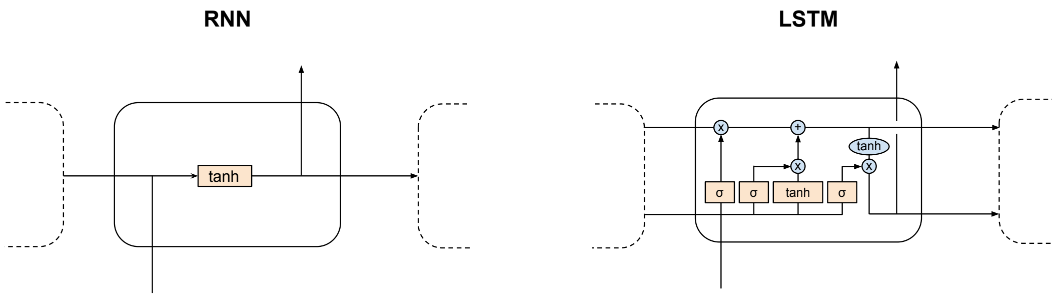


Figure 2.2: An image showing the difference between a traditional RNN and LSTM cell. The LSTM cell, has a more complex structure which allows for both the gradient and cell state to be passed to the next cell.

for more complex reasoning from past and future context. This can be seen in figure 2.3. The first three layers, Input, Encoding and Embedding, handles transforming the words into number arrays. The grey boxes are LSTM cells that feed both forward, and backward which are followed by a dense layer combining the LSTM outputs into a single value.

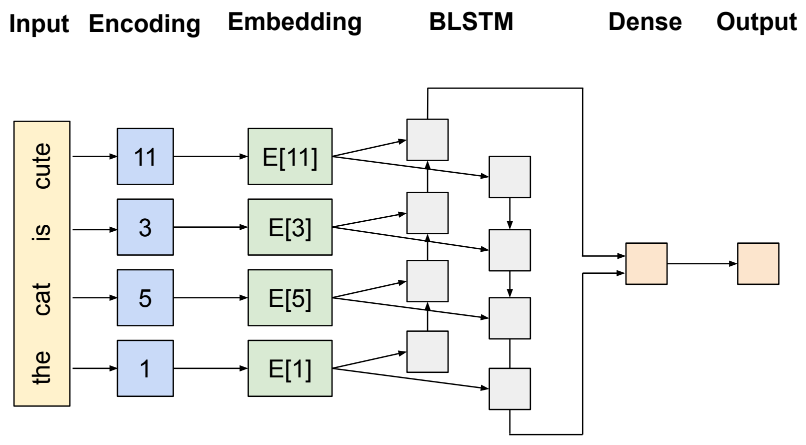


Figure 2.3: A simple binary classification neural network using BLSTM and a single dense layer as a single variable output.

Bidirectional Encoder Representations from Transformers (BERT)

BERT is a machine learning technique introduced by [Devlin et al., 2018] and, when it was published, was state-of-the-art for NLP tasks such as the GLUE benchmark[1]. It is based on the concept of *transformers* which introduces multiple advantages over older methods such as BLSTM. It, in extension to performing better, also can be parallelized during training to a

[1]GLUE is a benchmark for evaluating a models performance on general NLP tasks. `https://gluebenchmark.com/`.

higher degree than other methods, making the training process faster [Vaswani et al., 2017]. It was initially developed to solve translations between languages but can be used for multiple NLP tasks, including classification [Devlin et al., 2018].

Transformers are based on *multi-head self-attention*. Self-attention is the concept of building a relationship matrix between tokens defining how tightly coupled two tokens are. Figure 2.4 shows how the coupling could look between a word and the rest of the sentence during encoding. Multi-head self-attention indicate that multiple relationship matrices are produced to increase the chance of finding different types of relationship between words. While, for example, LSTM cells are sequential requiring the output of pasts cells to train the current cell the relationship matrices are independent and can be trained in parallel.

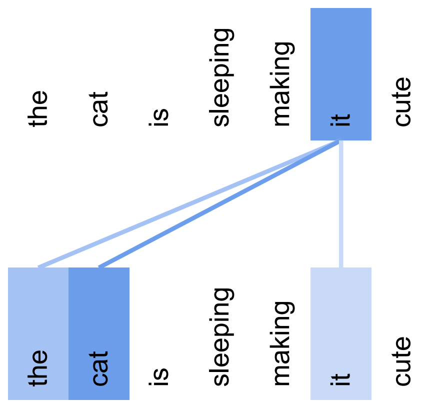


Figure 2.4: How a potential transformer encoder could define the relationship of 'it' to the other words. A stronger blue indicates that there is a stronger relationship.

BERT is a pre-trained language representation model based on transformers. Other transformer-based models such as openAI's GPT is unidirectional, encoding only context from earlier tokens in a sentence [Radford et al., 2018]. BERT, on the other hand, is bidirectional encoding both context from former and upcoming tokens [Devlin et al., 2018]. As with BLSTM, bidirectional transformer representation has been shown to perform better than their unidirectional counterpart [Devlin et al., 2018].

The original BERT created by [Devlin et al., 2018] is magnitudes larger size-wise than its BLSTM counterpart, giving longer training times. This has led to multiple different variations of BERT trying to decrease training times and model size but keeping performance. Examples of these are ALBERT [Lan et al., 2019], small BERT [Turc et al., 2019] and Electra [Clark et al., 2020].

2.3 Image Recognition

Computer vision and image recognition are much-researched interdisciplinary fields of computer science. This includes both image classification, the ability to define whether an image represents a specific *class* or thing, as well as object detection, the ability to locate an object in a larger view [Szeliski, 2010]. Image classification can be implemented by itself, while object detection is a combination of techniques, including classification [Szeliski, 2010]. Both options had to be considered for this thesis as they would be applicable for identifying a company logo through a website. This section will describe these two possibilities and their modern implementations.

2.3.1 Image Classification

What sets image classification apart from other detection algorithms is that it tries to handle a picture as a whole. Often only one object is present in the image and therefore the entire image can be assigned a label matching a predefined class of this object [Wang and Su, 2019]. Examples of classes and therefore labels could be *cat*, *car*, *bottle* and *logo*, depending on what objects should be recognized. The output from this kind of algorithm can be both just the label or an image with the label attached. An example of the latter can be found in figure 2.5. The most viable and simple option to implement this kind of solution is a combination of a convolutional neural network and transfer learning [Shin et al., 2016]. This will be described in this section.

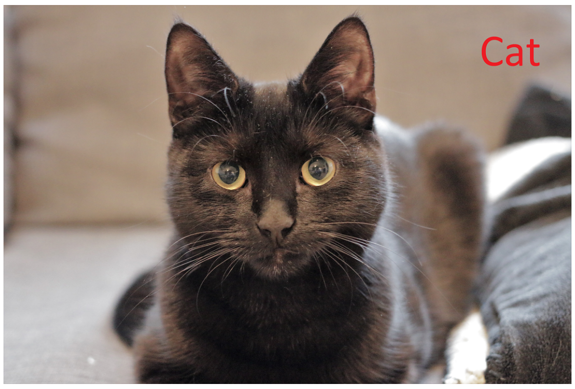


Figure 2.5: An example output from an image classification algorithm which can detect if an image depicts a cat.

Convolutional Neural Network (CNN)

A convolutional neural network, or CNN for short, is closely related to ANN in that they perform the same type of operations to establish a result [O'Shea and Nash, 2015].

One specific branch of ANN is Deep Neural Networks (DNN). DNN are defined as any multilayered ANN and it has been shown to perform well on computer vision, pattern recognition and NLP applications [Liu et al., 2017].

A DNN becomes a CNN when a convolution layer, which finds how the image should be analyzed, is present among the hidden layers [O'Shea and Nash, 2015][Albawi et al., 2017]. However, it is standard to also use two others specific layers, pooling layers, which decrease complexity and fully-connected layers [Albawi et al., 2017].

In the fully-connected layer, also called dense layer, the neurons are interconnected to both the adjacent layers in the same structure as a normal ANN. Therefore, the structure of this layer is the same as the hidden layer in figure 2.1 [O'Shea and Nash, 2015]. These are the layers which draws the final conclusions and classifies the image based on the features extracted in the previous two layers [O'Shea and Nash, 2015]. Increasing the amount of dense layers provide more neurons, which in turn has makes more learning possible [Albawi et al., 2017].

However, these layers are not without drawbacks. Fully-connected layers, especially with many neurons, cause an increase in computational complexity with more risk of overfitting [Albawi et al., 2017]. In the transfer learning used for this thesis, it is these layers which are fine tuned to the specific domain, company logos [Girshick, 2015]. This means different number of dense layers has to be tested in this thesis, to find the optimal learning process without overfitting.

Advanced classification CNN architectures has been developed to further increase performance and efficiency depending on both application and execution platform. Some popular examples of these are VGG [Simonyan and Zisserman, 2014], MobileNet [Sandler et al., 2018], ResNet [He et al., 2016] and DenseNet [Huang et al., 2017].

VGG In comparison to other modern CNN, VGG is often seen as a form of baseline for CNN classification architecture. What VGG showed was that several layers of small convolution kernels could perform admirably on large-scale images [Simonyan and Zisserman, 2014]. This model has in turn been used as a base to build the next three models from. VGG has two different variants depending on the number of layers it contains: VGG16 and VGG19.

MobileNet This model was created, as the name implies, to be run on mobile devices. Therefore, it must be as memory efficient and lightweight as possible. Here, it is solved by using several thin *bottleneck* layers following a convolution layer [Sandler et al., 2018]. Bottleneck layers reduce the dimensionality by using few nodes in comparison to the previous layer.

MobileNet has shown to be extremely efficient when working on image recognition and object detection with different data sets when compared to other CNN architectures [Sandler et al., 2018].

ResNet In contrast to most CNN architectures, which train on unreferenced functions, ResNet uses the residual data from the convolution layers as references in subsequent layers. This does not only provide easier learning on deeper DNN but also makes the models easier to optimize [He et al., 2016].

Three different variants of ResNet have been developed with varying layer counts with that number defined in the name, ResNet-50, ResNet-101 and ResNet-152. These architectures can be interchanged depending on complexity of the problem and efficiency. ResNet was specifically designed for image recognition and therefore performs admirably on such tasks [He et al., 2016].

DenseNet What separates DenseNet from other CNN architectures is mainly the connections between layers. A multitude of dense layers are present within and between the convolution layers, whose output is used by every single subsequent convolution layer. In other words, every dense layer is directly connected to every following convolutional layer, instead of just the consecutive one [Huang et al., 2017]. Examples of improvements this structure brings are alleviating the vanishing-gradient problem and greatly reducing the number of parameters [Huang et al., 2017].

Similarly to ResNet, DenseNet was designed for image recognition. Accordingly, it performs great on these types of problems [Huang et al., 2017]. It also has several layer variants which can be interchanged like ResNet, some examples are DenseNet-121, DenseNet-169 and DenseNet-201 [Huang et al., 2017].

To train and test these models with enough accuracy and comprehensibility, a colossal generic data set is required. When working with image classification the standard to use is ImageNet, which has a total of 3.2 million images [Deng et al., 2009]. ImageNet is built using a tree-like structure where an image can have several labels to aid in both broad and narrow classification [Deng et al., 2009]. For example, an image of a Siamese cat could have labels for mammal, cat and Siamese.

All of these models are available fully trained on ImageNet to be used with transfer learning. These are the completed models which will be used for the transfer learning in this thesis to find the optimal model for identifying logos. These were selected to get a wide array of transfer model complexity and efficiency.

2.3.2 Object Detection

Object detection, in contrast to image classification, handles two different tasks, classification and localization [Wang and Su, 2019]. An example of this can be seen in figure 2.6. Images on which object detection is used usually contain many different objects so finding if a specific object is present and where is a much more difficult and different task. Three examples of object detection algorithms which work quite differently are Bounding box regression [Lee et al., 2018], Region based CNN [Girshick et al., 2014] and Detection through classification. These three methods will be used in this thesis to locate the logo using screenshots of company websites, and will described in this section. These were chosen to get several basic approaches of object detection, as they all work differently, in an attempt to find a viable method of locating and extracting the logo.

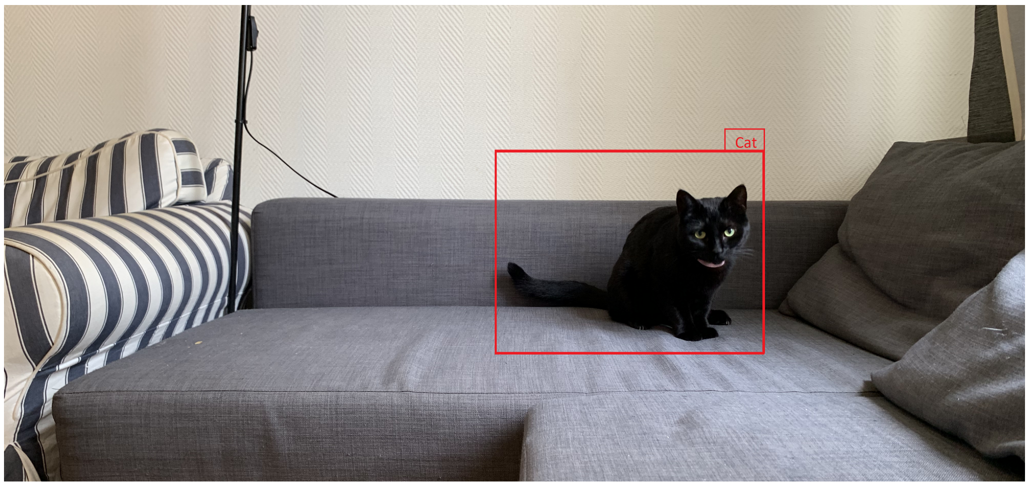


Figure 2.6: An example output from an object detection algorithm which can detect and locate a cat.

Bounding Box Regression

To simplify, this section will only describe bounding box regression with one class label which is what is done in this thesis. If there are several labels present, a separate detection and labeling step is needed.

Bounding box regression, as its name entails, has its basis in the machine learning model of regression [Lee et al., 2018]. The regression which is being calculated here is coordinates for a bounding box around the proposed object.

This kind of model therefore train on images and their *ground truth boxes*. An example of a ground truth box could be the coordinates for the actual four corners of the object in the image [Lee et al., 2018].

Bounding box regression is closely related to image classification, therefore their models are very similar. The difference lies in the fully-connected layers at the end. As described in section 2.3.1, CNN with transfer learning provide a strong basis for feature extraction where only the final layers are fine-tuned.

The fully-connected layers are constructed in such a way that a linear regression takes place here instead of a classification. In short using the input coordinates, these layers in turn train to produce an output consisting of a predicted bounding box [Lee et al., 2018].

An issue with bounding box regression however is that since it does not classify, and only locates possible coordinates for a bounding box, the model assumes the image has the object somewhere in the image. Therefore, if an image which does not include an object of the class to locale, a bounding box will still be predicted. This is in general not useful for this thesis since it will cause logos to be detected where there are none, however as a logo is extremely likely to be on a website is still a possible approach.

Region Based Convolutional Neural Network (R-CNN)

To solve the extra challenge of localization, Region Based Convolutional neural networks (R-CNN) first try to detect where in the image interesting objects could be before trying to classify them. R-CNN does this by employing any region proposal method, however the standard is *selective search* [Girshick et al., 2014].

Selective search works by identifying groups of similarities in the image and recursively combining them [Uijlings et al., 2013]. A graph-based image segmentation algorithm is applied to the image to detect these groups which initially is extremely small. This is however by design, as objects in the image can appear in any scale which this makes possible to identify. After finding the initial groups, the hierarchical selective search algorithm calculates the similarity between each neighbouring pair and starts merging the ones where this value is the highest. Bounding boxes are then drawn over each merged region. Every box is saved as a proposed object, which makes it possible to detect objects at any scale [Uijlings et al., 2013]. As the model will classify objects detected in this manner, it is good practice to also train on a data set extracted using selective search. This will be important later when evaluating the results for this method.

When predicting using an R-CNN model, the potential objects which the selective search algorithm identified are then classified with a CNN model as described in section 2.3.1. As selective search already has found a proposed region, this in combination with a classification label makes it possible to complete the object detection within an image [Girshick et al., 2014].

However, the issue with returning all objects detected using this method is overlapping bounding boxes. Since the selective search algorithm returns all possible objects, some of the suggestions could be subimages or partly overlapping the actual intended object [Uijlings et al., 2013]. The CNN model might predict these as objects as well, of which an example can be seen in figure 2.7. To solve this problem, some post-processing is needed to achieve the optimal bounding box.

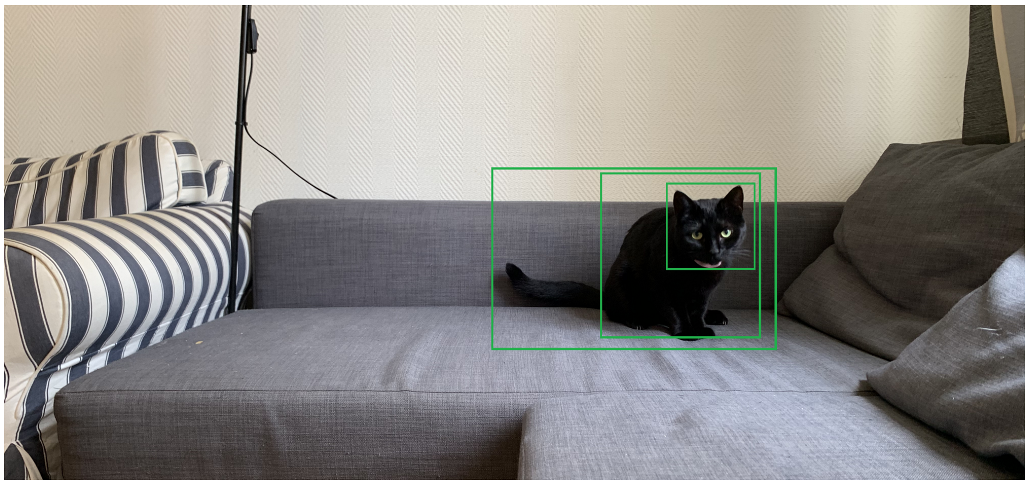


Figure 2.7: Examples of overlapping bounding boxes when trying to detect a cat using selective search.

Before explaining this post-processing step, it is important to explain *Intersection-over-Union* (IOU). IOU is calculated by dividing the area of overlap with the area of union between two boxes. This essentially means the higher the overlap and closer in size the two boxes are, the higher the value. If two boxes do not overlap at all, the IOU would be zero [Rezatofighi et al., 2019].

R-CNN uses *non-maximum suppression* (NMS) to solve the issue of overlapping boxes. After each object has been classified and a *classification score* has been assigned by the CNN the NMS can be applied. For each class which has been detected, if an IOU overlap exists between two boxes of the same class, remove the one with the lowest score [Girshick et al., 2014].

Several new models have been built expanding on the R-CNN idea for both faster performance and different applications. Some examples are Fast R-CNN [Girshick, 2015], Faster R-CNN [Ren et al., 2015] and Mask R-CNN [He et al., 2017]. However, these will not be tested in this thesis, as R-CNN should give a good enough representation of this method. R-CNN has the advantages of selective search, which has the possibility to locate the logos which have not been able to be extracted through normal web-scraping.

Detection Through Classification

In contrast to R-CNN which finds potential objects using selective search, detection through classification techniques tries to classify several subimages, also called *regions of interest* (ROI), of the full image and then uses NMS to locate the different objects. The final bounding box will match one of these ROI. To get satisfactory ROI of different scales and positions to cover the entire image two different methods are employed, sliding window [Lampert et al., 2008] and image pyramid [Adelson et al., 1984].

Sliding window is a region of static width and height which propagates over the entire image. The rate of propagation, or how many coordinates the region moves each time, decides both the amount of ROI to be extracted and therefore the magnitude of overlap between them. A larger number of ROI will increase the chances of finding a more exact bounding box but will come at the cost of efficiency as more subimages have to be classified. This movement in combination with the region can be seen as a window being slid over the image, which is where the name of this method derives from [Lampert et al., 2008].

Image pyramid is a method where an image is scaled down once a processing step is complete, for example when a sliding window has passed over the full range. As the ROI is of static size, when the image is scaled down more of the content is present in the ROI. Whatever processing step was used earlier is then repeated on the down-scaled image. This enables the possibility of finding objects which are larger in the original image than the ROI [Adelson et al., 1984].

Just like R-CNN, these two methods working in tandem produces several potential objects which are then classified and assigned a classification score by a pre-trained CNN. Also, similarly, NMS is then applied where overlapping bounding boxes with lower classification score is suppressed until only one box per predicted object remains.

As the predicted bounding boxes always matches a statically moving ROI of fixed size, it becomes highly unlikely the output will be matching the ground-truth box, which exactly surrounds the object. Therefore, some part of the surrounding composition in the output or the object being cropped is to be expected when using this method.

This method, similarly to R-CNN has the possibility of finding logos not available through web-scraping extraction and will therefore be used in this thesis.

2.4 Multimodality

Modality refers to a type of source, comparable to the real-world senses that can be experienced, sight, smell, hearing, and so on [Baltrušaitis et al., 2018]. Multimodality, therefore, refers to a single domain that contains multiple modalities, e.g., multiple types of data inputs or the challenge to generate one based on another. For example, speech is usually interpreted using both auditory and visual cues, making the domain of understanding speech multimodal (even though it can with some success be solved with a single modality). Multimodality in machine learning indicate that a data set includes different types of media, for example text and audio [Baltrušaitis et al., 2018]. Due to heterogeneity, a combination of modalities can be used to potentially reach a higher success rate than with a single modality [Zahavy et al., 2016].

By introducing multimodality in machine learning multiple challenges emerge that are not part of machine learning with a single modality. [Baltrušaitis et al., 2018] split the challenges into 5 different areas: *representation*, *translation*, *alignment*, *fusion*, and *co-learning*. Most of these are not relevant for the domain of this thesis but the challenge of fusion is.

2.4.1 Fusion

The challenge of fusion occurs whenever classification or regression is the desired output of a multimodal model. Multimodality can help giving robustness through redundant information and complementary information in multiple modalities and by fusing them in the system this can be utilized. There are multiple fusion methods, but the two main categories are *model-agnostic* and *model-based* methods [Baltrušaitis et al., 2018].

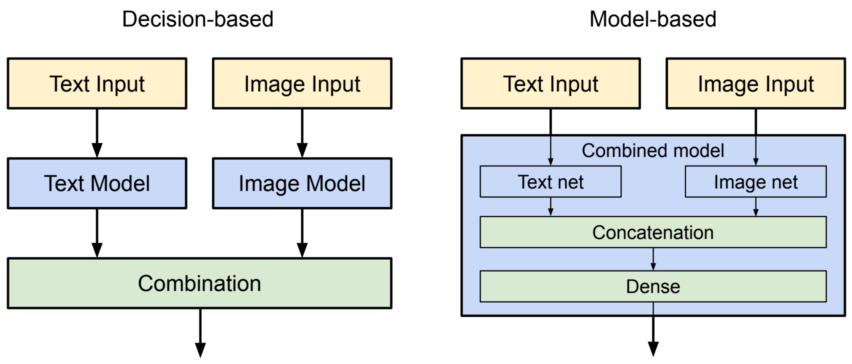


Figure 2.8: A visualization of the difference between decision-based and model-based fusion.

Model-agnostic approaches are when the fusion is done independently of the chosen machine learning models and can be split in three types *late* (decision-based), *early* and *hybrid* [Baltrušaitis et al., 2018]. Late fusion is when the decisions of the individual models are combined into a unified decision. Early fusion is when the representation from the modalities are combine into one representation that is fed into a single machine learning model and hybrid fusion is when both are used and then combined in the end. Only late fusion will be explored due to the complexity of merging text and image data into a unified representation.

Two different late fusion methods will be used in this thesis: a trivial weight-based version and a variant of the policy network proposed by [Zahavy et al., 2016]. The weight-based decision will be done by assigning a weight (importance) to each model and then adding them together to create a unified decision. If w_i is the weight and d_i is the decision for the ith model the following formula (weighted arithmetic mean) is used to produce the unified decision, see equation 2.1.

$$\frac{\sum_{i=1}^{n} w_i \cdot d_i}{\sum_{i=1}^{n} w_i} \tag{2.1}$$

A policy network is a small ANN that will have model outputs as input and produce its own prediction output based of the models. This allows for a more complex reasoning than the pure linear nature of the weight-based approach.

Model-based approaches are when the fusion is done inside of the machine learning model and allows for a more complex combination of the modalities than model-agnostic approaches [Baltrušaitis et al., 2018]. In the case of neural networks model-based fusion is done in a hidden layer combining the output of parallel layers handling the different modalities. See figure 2.8 for a comparison of model-agnostic and model-based approaches.

Oracle

To be able to evaluate the potential performance gain of using multimodality an *oracle model* will be used in this thesis in the same way as [Zahavy et al., 2016]. It takes the output of x number of models and if any of them has guessed the label correctly the output of the oracle will be correct. In practice this means that if a multimodal model comes close to the performance the oracle it has the same output as the correct model(s) often.

2.5 Evaluation

To be able to compare different approaches for finding the company logo, a consistent and comparable evaluation method must be established. In classification problems, such methods have already been researched and tested [Lever et al., 2016], three of which are used in this thesis: Standard metrics (Accuracy, Precision and Recall), *Confusion matrix* and *Matthews correlation coefficient* (MCC). This section will explain these and their usage in this thesis.

2.5.1 Accuracy, Precision and Recall

Before going into these three metrics, it is important to understand what they are based on. As this thesis explores a binary classification problem, where an image either is a logo or not, the labeling of an image can be seen as a positive if it is a logo while not a logo is negative. These identifications can also be swapped when analysing metrics on the other class. Therefore, there are four possible outcomes to the classification, which can also be seen in figure 2.9, which are:

True Positive (TP) Actual positives which were correctly labeled by the classifier. An image which is a logo was labeled as a logo.

True Negative (TN) Actual negatives which were correctly labeled by the classifier. An image which is not a logo was labeled as not a logo.

False Positive (FP) Actual negatives which were wrongly labeled by the classifier. An image which is a not logo was labeled as a logo.

False Negative (FN) Actual positives which were wrongly labeled by the classifier. An image which is a logo was labeled as not a logo.

The first metric is *Accuracy*, which shows the percentage of correctly labeled images compared to the entire data set, see equation 2.2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

Accuracy is a good metric to get a basic idea if the model is working, as high accuracy in general shows a well performing model. The issue with accuracy is when the data set is imbalanced it makes accuracy unreliable. This is due to bias, where it provides an overoptimistic estimation of the classifier ability on the majority class [Chicco and Jurman, 2020]. As websites in general has substantially more images which are not logos, this bias must be

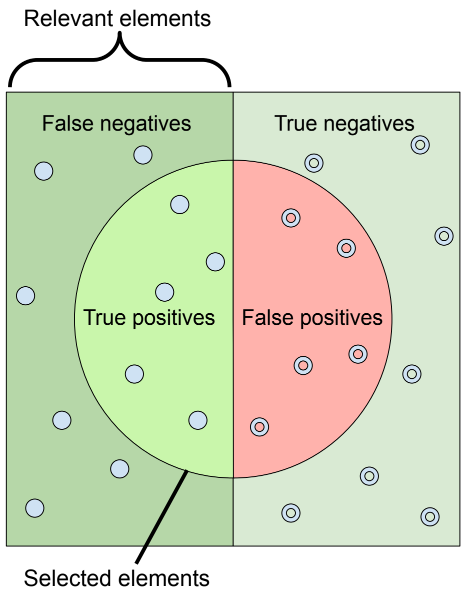


Figure 2.9: A visualization of the possible outcomes and their relations from a binary classification.

taken in consideration when evaluating the models. This is why the two other basic metrics are needed.

These methods are closely linked and are often talked about as a pair, *Precision* and *Recall*. Precision identifies the proportion of positive classifications which was correct, see equation 2.3.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Recall in turn calculates the proportion of actual positives which were identified correctly, see equation 2.4:

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Precision and Recall in combination give a more precise interpretation of a model's performance, where higher and balanced values indicate more accurate and complete classification. However, they usually have a constant tug-of-war where improving precision typically reduces recall and vice versa [Buckland and Gey, 1994].

2.5.2 Confusion Matrix

To get a visual overview of a model's performance, a confusion matrix can be used. In such a matrix, the different outcomes are divided into four quadrants with predicted values on the x-axis, and with actual values on the y-axis [Chicco and Jurman, 2020]. An example confusion matrix is shown in figure 2.10.

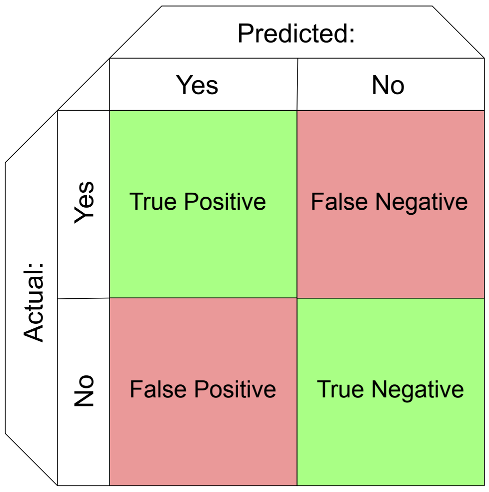


Figure 2.10: A general confusion matrix for a binary classifier. The *True* sections are the wanted results where the predictions match the actual values.

2.5.3 Matthews Correlation Coefficient (MCC)

As described earlier, several different metrics are needed to get a good overview of the performance of a classifier. However, as the number of metrics increase, the comparison between them becomes a problem. For example, is a classifier with higher recall than precision better or worse than a classifier with higher precision than recall, or is a completely balanced classifier the best?

A popular metric to get a single comparable value is F1-score, which is an average of precision and recall. However, F1 can show overoptimistic inflated results on imbalanced data sets [Chicco and Jurman, 2020]. Therefore, to get a single value to order and compare different models and their multimodal combinations, *Matthews correlation coefficient* (MCC) will be used. See equation 2.5.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.5}$$

MCC is a value between -1 and 1, where **1** indicates a perfect classifier, **-1** a perfect inverted classifier and **0** random guesses. Therefore, any classifier with a MCC closer to 1 can be seen as a superior on that specific data set [Chicco and Jurman, 2020].

Not only does MCC solve the issue of comparing classifiers. MCC is also unaffected by the issue of unbalanced data sets [Chicco and Jurman, 2020]. This makes it possible to use all images extracted from a website to evaluate the classifier, instead of discarding non-logos to achieve a balanced test data set.

However, it is important to recognize there is a risk MCC becomes undefined. If any row or column of the confusion matrix has a total of zero values, for example if no negative values were predicted at all, it becomes undefined and cannot be used [Chicco and Jurman, 2020]. Luckily, this should only happen on small data sets and is therefore not a problem in this thesis.

Using several metrics could still provide more robust analysis, which is why precision, recall and confusion matrices are still part of the evaluation. For example, MCC does not provide information about the specifics of the positive values as precision and recall does, which could be valuable information. In some classifying problems, such as the detection of cancer cells, false negatives are very costly and therefore recall has the most importance [Kourou et al., 2015]. MCC only provides an objective classification value where all outcomes are valued the same, which is the case in this thesis.

Chapter 3
Data set

One of the primary challenges when building machine learning models, as well as verifying a more traditional algorithm's performance, is having an annotated data set that represents the whole domain of possible inputs. In this case the data set is company websites, and the images available on them. No data set matching the requirements was found. This section will discuss how the data set was collected and annotated, and discuss potential flaws.

3.1 Forbes Global 2000

The authors initial theory was that websites and logos may look and be developed differently in different regions of the world. It was therefore a priority to build the data set based on companies worldwide to avoid region-based bias.

To accomplish this *Forbes Global 2000* [Andrea Murphy and Touryalai, 2020] list was chosen as the data set. The list includes the world's 2000 largest companies from 64 different countries from all continents but Antarctica, making it regionally diverse. Each company website address was manually collected using a search engine and the Wikipedia-page of said company making the process likely to yield the actual website for each company.

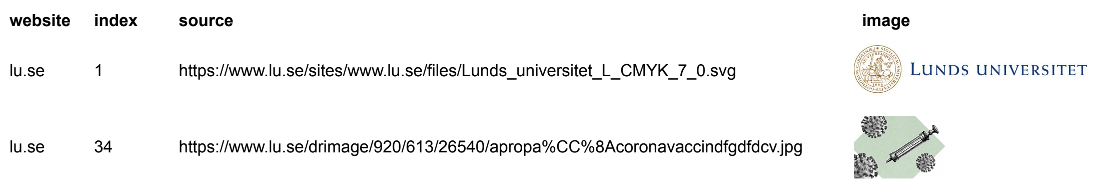
Consequently, the data set is a list of company names and their corresponding website addresses. For image detection a smaller data set of screenshots were collected and annotated using ground truth bounding boxes around the logos. For classification, which is the primary approach, each available graphic element and its corresponding source was extracted.

3.2 Images on Websites

Websites are encoded using HTML, CSS and JavaScript. HTML and CSS allows for graphic elements to be included and rendered in multiple ways making it challenging to successfully extract every element. The different ways to include logos in the data set are as HTML `<img/>`

tags, with the CSS `url()` function as either a url or data uri, inline `<svg>` tag and as a custom font. Every format, except the custom font, was successfully extracted.

Each data point is a collection of the website origin, an incrementing index, an image and its source. The source differs depending on how the image is included. In case the image is included as a path, the path is the source, if the image is included as data uri, that is the source and if it is an inline svg the complete `<svg>...</svg>` is the source. The index is used to identify individual data points since the combination of the index and the website is unique. An example of two data points can be seen in figure 3.1.



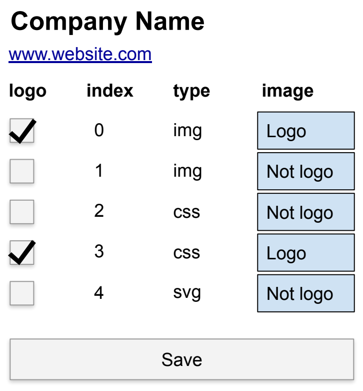
website	index	source	image
lu.se	1	https://www.lu.se/sites/www.lu.se/files/Lunds_universitet_L_CMYK_7_0.svg	
lu.se	34	https://www.lu.se/drimage/920/613/26540/apropa%CC%8Acoronavaccindfgdfdcv.jpg	

Figure 3.1: An example of two data points, one that is the logo and one that isn't a logo.

To make the results reproducible and the data set persistent the website source file, and each graphic element of every website was downloaded and saved, resulting in a data set size of 15.7 GB.

3.3 Annotation Tool

To be able to annotate the data set, which includes more than 100 000 data points, a custom annotation tool was developed. The tool renders each element website-wise and allows the user to set whether the element is the company's logo, see figure 3.2.



Company Name

www.website.com

logo	index	type	image
✓	0	img	Logo
☐	1	img	Not logo
☐	2	css	Not logo
✓	3	css	Logo
☐	4	svg	Not logo

Figure 3.2: A visualization of the interface of the custom annotation tool. The blue boxes would be the actual elements rendered.

3.4 Annotation Guidelines

The most fundamental guideline was "If the image is used as a logo and can be used to identify the company—which owns the website—it is considered a logo" but it is vague due to corner cases. Therefore, during annotation, guidelines were set when corner cases occurred to ensure a consistency in the annotation. The following rules were defined. A visualization of the rules can be seen in figure 3.3.

Yes An image with the logo, company name, and with or without slogan.

Yes An image of an icon that can be considered a logo without the company name.

Yes An image of a monochrome version of the any of the above.

Yes The background can be either transparent or monochrome.

Yes An image can have any size if it is clear it is the logo.

Yes An image of an older logo for the company.

No A logo that is a sub image of a larger image, this includes monochrome backgrounds that are notably larger than the logo.

No A sub part of the logo that cannot be considered a logo by itself.

No A logo that is themed for a specific use, e.g., it is holiday specific or used as a mask on another image.

No A logo that includes a sub brand of the main company.

No A logo that has a background stylized beyond being monochrome.

No A monochrome version of the original logo that differs to much from the original.

The resulting annotated data set is one main list containing all companies and their websites and a separate list with data and labels related to the elements on each company website, see figure 3.4.

3.5 Characteristics

The resulting data set contains 170 895 data points distributed over 1 905 websites. The difference between the number of websites and companies can be explained with websites being unavailable or missing. Out of the all the data points 5 442 are classified as positives (logos) and the remaining 165 453 as negative (not logos) resulting in a large class imbalance of around 3.2% logos. This is seen as reasonable given that websites seldom have many logos and usually multiple other images and icons. The distribution can be seen in figure 3.5. Due to the imbalance the distribution of negatives closely follows the distribution of the total number of images per website.

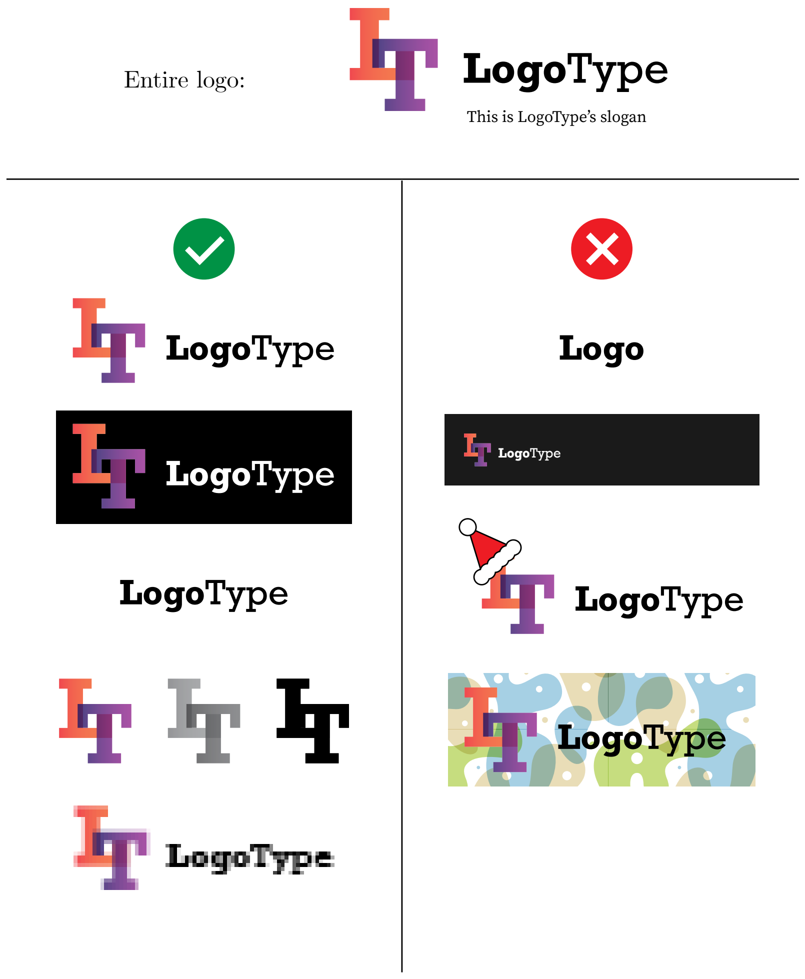
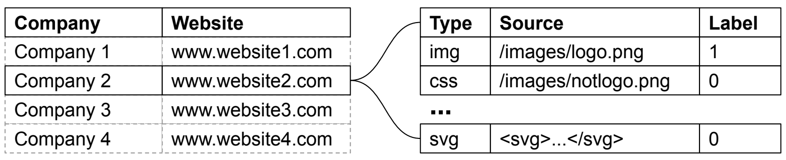


Figure 3.3: An example logo and different variations of it. The variations to the left were annotated as logos, and the variations to the left were not.



Company	Website
Company 1	www.website1.com
Company 2	www.website2.com
Company 3	www.website3.com
Company 4	www.website4.com

Type	Source	Label
img	/images/logo.png	1
css	/images/notlogo.png	0
...		
svg	<svg>...</svg>	0

Figure 3.4: A simplified visualization of how the data set is structured. Data points labelled with 1 it is a logo, data points labelled with 0 is not a logo.

Due to time limitation during the development of the data set a single person annotated all data points. This may have led to two different problems, *annotator bias* and *mislabeled data points*. Annotator bias is bias introduced by annotators knowledge of the area and subjective perception. For example, logos of companies known to the annotator might be less likely to be mislabeled while companies using alphabets unknown to the annotator might be more likely to be mislabeled. Sometimes data points are mislabeled due to no apparent reason.

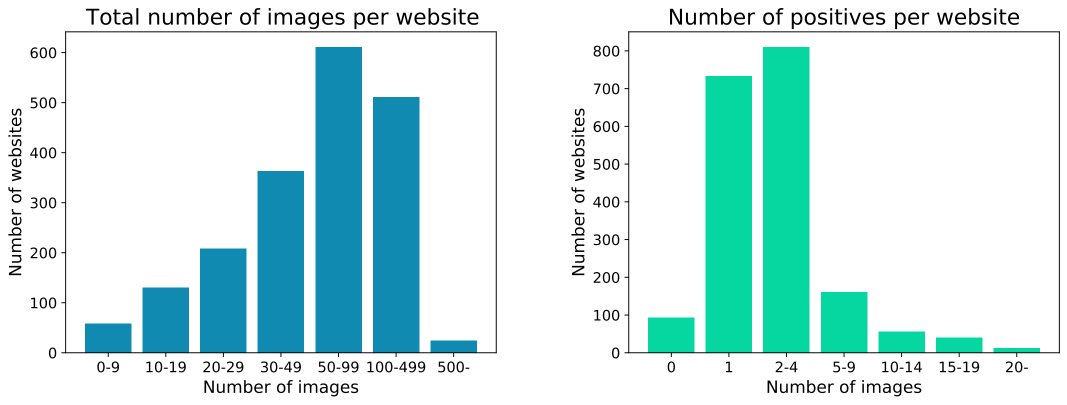


Figure 3.5: Distribution of the data set per website. The left graph shows the distribution of the total number of images per website and the right shows the total number of logos (positives).

This could be tackled by having multiple annotators annotate all data points. All annotators doing the same mistake is less likely than a single annotator.

Chapter 4 Method

In summary, this work can be broken up into these steps:

1. Build a web-scraping tool which collects all images and their sources on a specific site.
2. Assemble and annotate an extensive data set of websites and their images using a custom annotation tool.
3. Implement a trivial algorithm using knowledge engineering as a baseline by studying the collected data.
4. Research and implement different possible text and image classification models.
5. Run extensive testing until the best performing individual models can be identified.
6. Combine these models into the multimodal solutions, a model-based and two decision-based, and run further tests.
7. Evaluate and determine the results.

In general, the steps were also completed in this order, where one was not started before completing the previous. However, some exceptions were present. For example BERT was implemented after performance testing of other models had already begun. These exceptions happened when either unexpected problems occurred, or when new information was gathered which could alter the results, like in the BERT example.

Due to ease of use, Python with Tensorflow-Keras was chosen as the programming language and machine learning backend. Each method was implemented and tested using a simple pipeline of preprocessing the input data, running the model, and evaluating the results, which can be seen in figure 4.1. This section will describe the specific pipelines for the three methods and their technical implementation, after a description of how the data set is split before use.

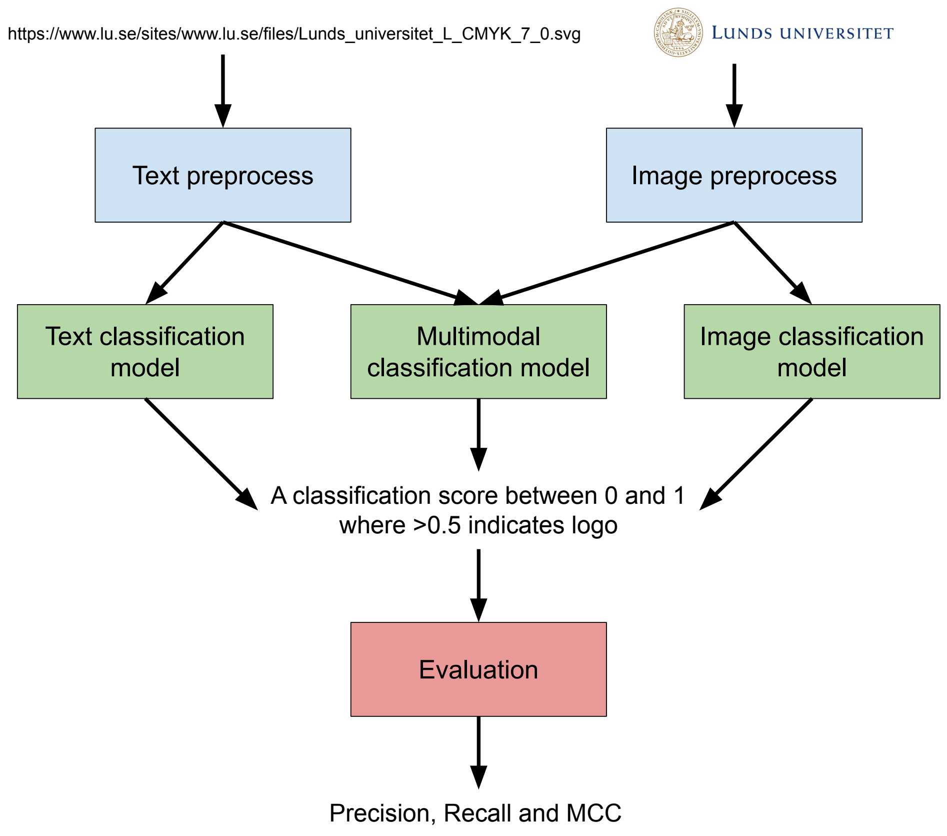

https://www.lu.se/sites/www.lu.se/files/Lunds_universitet_L_CMYK_7_0.svg
Lunds universitet
Text preprocess
Image preprocess
Text classification model
Multimodal classification model
Image classification model
A classification score between 0 and 1 where >0.5 indicates logo
Evaluation
Precision, Recall and MCC

Figure 4.1: A general overview of the different classification pipelines. The preprocessing steps, individual models and evaluation methods vary depending the specific pipeline.

4.1 Train, Validation and Test

For this thesis, a 60/20/20 split (60 % train, 20 % validation and 20 % test) was chosen, due to it being common. It is important that the test set does not intersect with the train or validation set to ensure that the model has not trained on the evaluation data. This is done by physically separate the data on the hard drive but due to the structure of the data set it proved hard to perfectly split it in the desired quota. A random selection of 20% of the website folders were chosen and moved to a separate folder. Since a website may have a variable number of images—*especially* non logos—20% of the websites might not align with 20% of the total data set. After the split, the test set contained 31 767 (19.2% of total negatives) negatives and 1 040 (19.1% of total positives) positives. This was considered close enough to the desired quota.

The likelihood that two files on different websites generate the same text input is very unlikely but there might be overlap when it comes to the images. For example, different social media logos might appear at the bottom of multiple websites to link to their respective profiles. This means that the same image might be in both the train and test data set possibly inflating the performance of the image models during evaluation.

4.2 Primitive Baseline Classification

To have a baseline to compare the results to, a primitive baseline was defined. The baseline chosen was statistical mode of random selection instead of always choosing the majority class. Always choosing the majority class is a good fit with accuracy as the metric, but both precision and MCC becomes undefined with this method, due to division by zero. With statistical mode of 50/50 random selection, all three of the metrics used during evaluation gets a concrete value, making it a better fit.

4.3 Text Classification

The following section will describe the preprocessing and testing pipelines for the different text classification methods. It will follow the same order as the techniques described in section 2.2. The set up for each model is the image text source (see section 3.2) as input followed by the preprocess and then the model itself.

4.3.1 Preprocessing Data

Artificial neural networks for text classification uses words and text vectorization to produce appropriate input for the neural network. The source for each image does not work by default due to not being split into words. A path, for example, is a continuous string of characters without any spaces. To allow for text vectorization a custom preprocess was created to turn the image source into words. Multiple rules for splitting the strings into multiple words was defined: Each non-alphanumeric character is replaced with a space, camel-cased words are split into multiple words and all characters are converted to lower case. Longer strings (such as large svg's and data-uri's) is cut after 5000 characters. For example, `https://www.website.com/path_to/SomeFile.pdf` would become `https www website com path to some file pdf`.

For the knowledge engineering classifier, the only preprocessing needed was to turn the text source into all lower-case.

4.3.2 Pipeline

The text classification methods had the testing pipelines setup as follows.

Knowledge Engineering

By inspecting the train data set and trying to find patterns the authors defined five possible rules for identifying if a source is a logo:

r1 The text contains the substring 'logo'.

r2 The text contains the substring 'favicon'.

r3 The text contains the substring 'icon'.

r4 The last 20 characters of the text contains the company name.

r5 The text does not contain the name of a social network[1].

These rules were combined in multiple different ways with the `and` and `or` operators. The rule-combinations were then run for each data point in the test data set and each evaluation metric was calculated.

BLSTM

The BLSTM model structure can be seen in figure 4.2 and is structured as follows. The models consist of a TextVectorization layer with a variable vocabulary size. The vocabulary size is the number of dierent words that will get its own unique encoding, the most common words are prioritized. It is followed by Embedding layer, with an output size of 64, which is followed by LSTM layer wrapped in Bidirectional layer. After that, a variable number of Dense layers using relu as activation with decreasing output size follows. For example, if 3 extra dense layers would be used, they would have output size 64, 32 and 16, respectively. The final layer is a Dense layer using sigmoid as activation and with output size 1.

During training, an undersampled or oversampled data set was created by, at random[2], choosing data points at the desired ratio from the training and validation data sets. The model was trained using Binary Crossentropy as the loss function, Adam as the optimizer and early stopping on validation loss with a patience of 3. After some minor testing a learning rate of $3 \cdot 10^{-5}$ was chosen.

At first different vocabulary sizes and extra Dense layers were tried to determine whether the complexity of the model had an impact on its performance, then different methods to counter class imbalance were tested. When oversampling was tested it was clear that the performance was worse for all metrics. Since oversampling and cost sensitivity produces similar results, cost sensitivity was not evaluated. For each setup 3 models were built, to minimize the risk that a statistical outlier was used during evaluation.

To analyse which type of text sources the different models mislabeled, a random subset of 10 mislabeled text sources were extracted. The images in this subset were then manually evaluated.

BERT

The BERT model structure can be seen in figure 4.2 and is structured as follows. The models consist of a TextVectorization layer with a variable vocabulary size. The vocabulary size is the number of dierent words that will get its own unique encoding, the most common words are prioritized. It is followed by Embedding layer, with an output size of 64, which is followed by LSTM layer wrapped in Bidirectional layer. After that, a variable number of Dense layers using relu as activation with decreasing output size follows. For example, if 3 extra dense layers would be used, they would have output size 64, 32 and 16, respectively. The final layer is a Dense layer using sigmoid as activation and with output size 1.

During training a undersampled data set with ratio 1:3 was used. The model was trained using Binary Crossentropy as the loss function, AdamW as the optimizer and early stopping

[1]The social networks used: facebook, linkedin, weibo, twitter, instagram, pinterest, reddit, tiktok and youtube

[2]A consistent seed was used throughout the testing.

on validation loss with a patience of 3. After some minor testing a learning rate of $3 \cdot 10^{-4}$ was chosen.

During the completion of this thesis, a bug in Tensorflow 2.4 caused BERT to not work on Windows, which was the operating system used during development. It was fixed on the nightly build for Tensorflow 2.5, however, it was not possible to use GPUs when training on this version. Therefore, it took an unreasonable amount of time to train each BERT model making it not viable to perform substantial testing of BERT.

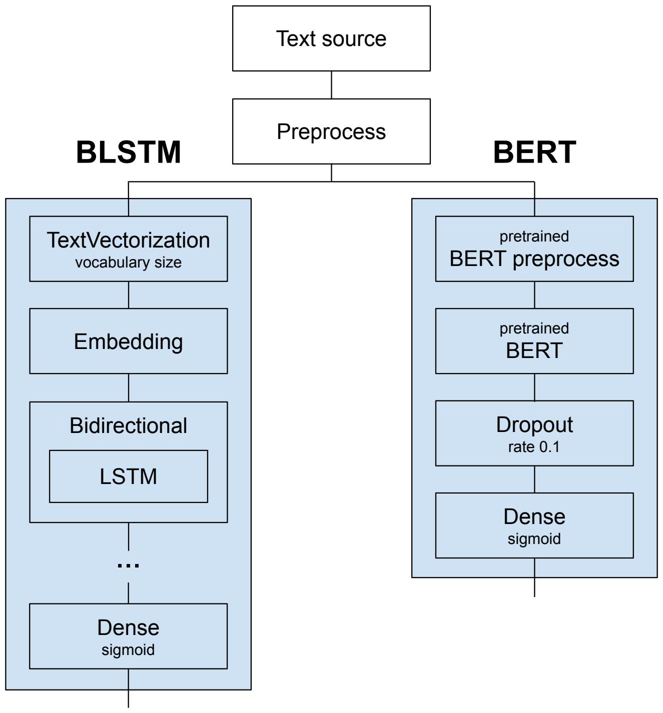


Figure 4.2: Layer structure of BLSTM and BERT classifiers. The dots in BLSTM refers to the variable amount of extra dense layers. The pretrained layers are complete architectures (i.e. multiple layers).

4.4 Image Recognition

This section will describe the different testing pipelines for the four image recognition methods implemented in for this thesis, CNN, bounding box regression, R-CNN and detection through classification. It will follow the same order as the techniques were described in section 2.3.1.

4.4.1 Preprocessing Data

Before using the images from the data set they have to be preprocessed in order for Keras to evaluate them properly.

A challenge when preprocessing image data was image formats. Some format very converted. For example, SVGs could be temporarily saved locally as a supported format before being loaded into memory, or the first frame of GIFs being used.

The transfer models used for image recognition all perform best when the images are a specific resolution, 224x224 pixels. This is because the transfer learning methods used are all trained on ImageNet with images of that size, which has been the standard since ImageNet was first used successfully [Krizhevsky et al., 2012]. Therefore, another important preprocessing step was scaling the image to this resolution. This itself is not a difficult task, but it is important to know the effects this has on the images as it could greatly affect the image and therefore the final classification. When up-scaling, increasing the image resolution using nearest-neighbour interpolation, it greatly affects the images. They become very pixelated, as seen in figure 4.3, which causes the images to be very unclear. When down-scaling images, lowering the resolution, smaller details become more obscure, and images can quickly become crowded. An example of this can also be seen in figure 4.3.

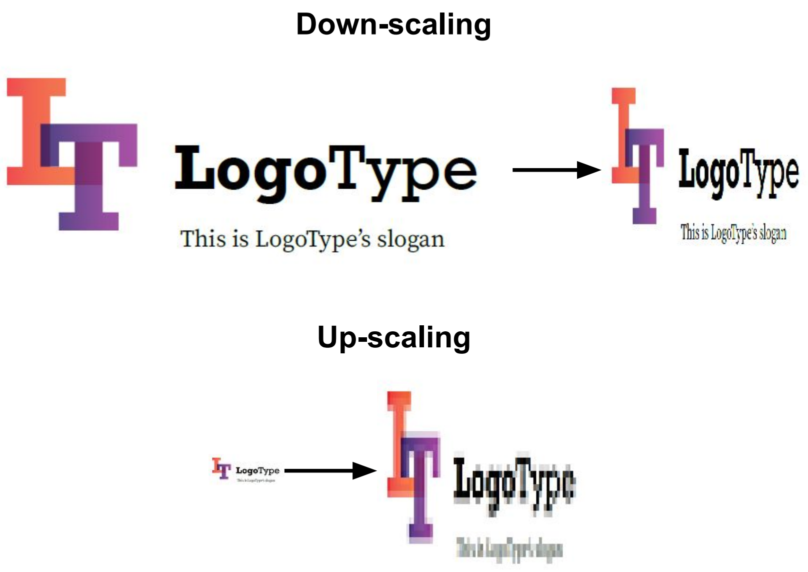


Figure 4.3: Examples of down- and up-scaling images of different sizes to 224x224.

Another image preprocessing challenge was the presence of the alpha channel in images. The alpha channel is responsible for possible transparency in images in RGBA mode. Keras' implementation of transfer learning only supports images in RGB mode, not RGBA. This means no possible transparency in the image could be trained on. As the alpha channel lies in the foreground of many images, just removing this channel often causes artifacts from the other colours where the transparency originally was. To solve this problem, possible alpha values were replaced with a grey background, which is added to the RGB channels. Grey was chosen as it is a neutral color, likely to contrast with the common black and white outlines of logos. It also lets the colours be the focus of the image, which is often the case for logos. It was also seen as unlikely many logos would use grey as part of their logos, because of its neutrality. This increases the amount of visible detail in the image and covers the artifacts. An example of this can be seen in figure 4.4.

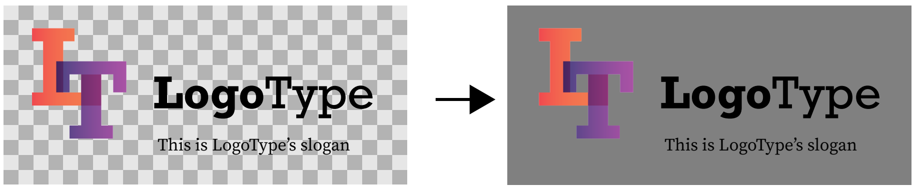


Figure 4.4: Example of a logo where the alpha channel has been replaced by a grey background.

4.4.2 Pipeline - Image Classification (CNN)

The transfer model is loaded, and every layer of that model is frozen. A trainable head is thereafter applied consisting of an 7x7 average pooling layer, a flatten layer and a varying amount of dense layers with relu as activation, before finishing with a sigmoid dense to get the desired classification score. The CNN model layer structure can be seen in figure 4.5. The 7x7 pooling layer decreases the complexity of the model to decrease the runtime, while the flatten layer prepares the data for learning in the dense layers. The combined model is thereafter compiled with a Binary Crossentropy loss function, which is optimized for binary classification, and an Adam optimizer with an initial learning rate of 10^{-4}. Learning rate was selected by running minor optimization tests.

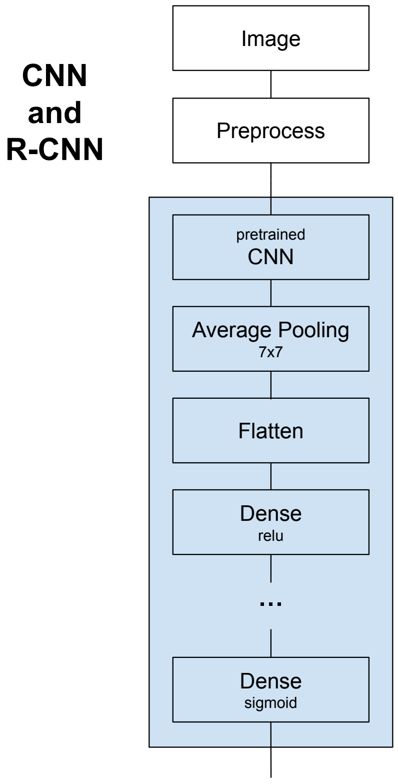


Figure 4.5: Layer structure of CNN and R-CNN classifiers. The dots in refers to the variable amount of extra dense layers.

After the image data has been preprocessed according to section 4.4.1 it is balanced according to a specified ratio. Batches of augmented data using zooms, rotations, shifts and shears were generated from this data using Keras' *flow* function, which increases the amount of data available and therefore can improve the results [Perez and Wang, 2017]. These are then used to train the models.

The model is then used to predict the complete test set of possible candidates. The result from this prediction is then used to calculate the metrics described in section 2.5.

Conforming to this set-up, several tests were then done by the authors. Firstly, tests for finding the best class balance were ran and completed on the full test set. The next batch of tests is aimed evaluating the effect of transfer model and complexity of the CNN model. Therefore, the tests were run with different model complexity as well the different transfer models. To increase the complexity of the models as described in section 2.3.1, pairs of Dense layers were added before retraining and reevaluating each model. The tests were also run on the completed data set, using the best balance ratio found in the previous set of tests. At least one example for every tested transfer model of these final results is shown in table 5.8 with the entire table available in the appendix in table B.1.

To analyse which type of images the different models mislabeled, a random subset of mislabeled images were extracted. The images in this subset were then manually evaluated by the authors.

4.4.3 Pipeline - Object Detection

To test object detection, three methods were implemented by the authors. These were bounding box regression, R-CNN and detection through classification. Each different object detection model had different testing pipelines, as each method works differently. The main thing they have in common is the use of website screenshots to detect the logo rather than the web-scraped images. These separate pipelines and their implementations will be described in this section.

Bounding Box Regression

As the bounding box regression model require screenshots of websites and the ground truth boxes for the logos, as described in section 2.3.2, it uses a separate data set from the image classification model. This is instead built-up using screenshots of the same websites as described in section 3, as well as the bounding box coordinates for the logos in these images.

Once the transfer model is loaded all its layers are frozen. Thereafter a trainable head is applied consisting of a flatten layer, to prepare the data for the fine tuning, and four dense layers with decreasing shape, to get a result. The bounding box regression structure can be seen in figure 4.6. The combined model is thereafter compiled with a Mean Square Error loss function, which is commonly used for regression problems [Toro-Vizcarrondo and Wallace, 1968], and an Adam optimizer with an initial learning rate of 10^{-4}, which was found by minor testing. The model is then trained and evaluated by outputting four values which indicates the corners of a bounding box surrounding the image.

Tests using pretrained VGG-16, a basic CNN architecture described in section 2.3.1, as the transfer model, to get baseline results and analyse the viability of this method, were then concluded by the authors.

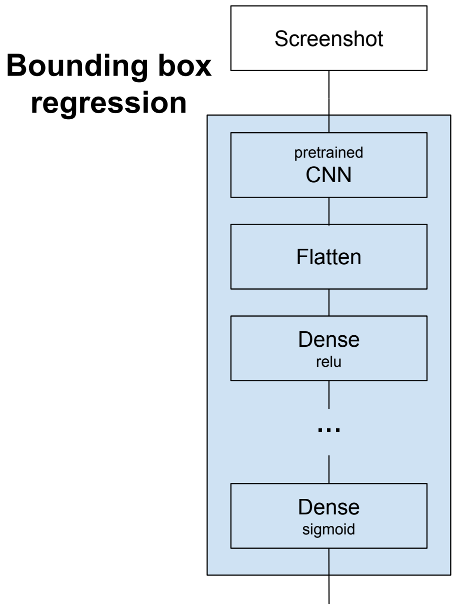


Figure 4.6: Layer structure of the BBR model. The dots refers to the variable amount of extra dense layers with relu as activation. The pretrained layers are complete architectures.

R-CNN

Since it is part of the R-CNN model, as described in section 2.3.2, selective search was used by the authors to get the data set for these models. This data set was built by the authors by having annotated ground truth boxes for several website screenshots and running selective search on these images. If a proposed selective search element had an IOU of 0.7 or higher, it was labeled by the selective search algorithm as positive while elements with an IOU lower than 0.05 were labeled as negative. Other proposed bounding boxes were rejected. These IOU values were selected by running selective search on a smaller amount of website screenshots, and the authors manually analyzing the predicted images and their IOU to find suitable thresholds. The upper threshold should make sure the logo is clearly present in positive images and take up the majority of the image, hence the 0.7 IOU. The lower threshold should ensure the logo cannot be identified in the negative image, however smaller parts of the logo are still okay where it still cannot be recognized, hence the the 0.05 IOU.

The following training was then set-up by the authors identical to the CNN set-up described in section 4.4.2, as the two methods work the same following the data set preprocessing, and seen in figure 4.5. The exceptions are the generated data set used as well as the images in this data set were not preprocessed as selective search already has achieved this. Similarly to CNN, this produces a classification score indicating the presence of a logo, but also a possible bounding box for the logo thanks to selective search.

To evaluate viability, this set-up was firstly trained using MobileNetV2, to get faster training and evaluation, with some different class balance options.

Following this, a different selective search generation was run by the authors to get a separate data set where elements which are similar to logos have a higher chance of being selected as negative. The model was then trained on this new data set and evaluated again.

Detection Through Classification

A sliding window algorithm was implemented by the authors where each subimage was evaluated using the best image classifier from section 4.4.2. This window has the size of 224x224 to conform to the resolution the model has been trained to predict and moved a varying amount of pixels, between 5 to 20 pixels to get an idea of how much detail were needed, per iteration. Once the entire image has been processed by the window, the image is resized, and the window is run again by using an image pyramid. This method then outputs the windows with the highest classification score.

4.5 Multimodal Classification

To achieve multimodality in this thesis, two model pipelines were created. These are the model-based and decision-based fusions found in figure 2.8. As a prerequisite to being able to combine solutions with both these pipelines, trained and evaluated text and image classification models must have been implemented. The separate models are implemented differently but handle the preprocessing of data the same, by using the individual model's preprocessing pipeline.

4.5.1 Model-based

This solution's goal is to combine the different models features before the Dense layers. This is done by using the Concatenate layer, which combines two separate layers into one output.

The two models which are to be combined, BLSTM and CNN, were implemented by the authors equal to their individual architectures except their dense layers. Which in this thesis is an image transfer learning model is loaded and frozen, and a trainable head is placed on top, equivalent to the CNN model. As well as an embedder run, equivalent to the BLSTM model. The outputs from these separate layers are then connected through the concatenate layer to get a single output. This structure can be seen in figure 4.7. To keep the image and text data connected, no augmentation was done on the images. The combined model is complied with a binary cross entropy loss function, the standard for binary classification, and an Adam optimizer with an initial learning rate of 10^{-4} with early stopping from validation loss with a patience of 2. Both the learning rate and early stopping patience were decided by minor testing where they performed the best.

An array of data arrays is used as the input for this type of model. However, it is crucial the separate data arrays are of the same type, as the concatenation requires the data set to be comparable. Therefore, preprocessing is handled simultaneously and equally for both text and image data, which in contrast would not be necessary for model-agnostic approaches.

Similarly to when testing other classification methods, the first step is finding the best balance ratio for this kind of model, in the same way as in section 4.4.2. Therefore, a baseline model was set up by the authors with a very large vocabulary size, to ensure all words that could help classify an image had a unique encoding, and DenseNet-169 as the image classification transfer model, as this architecture had shown promise in early implementational testing. The baseline model was thereafter trained on several variants of balance ratios, the results of which can be found in table 5.9.

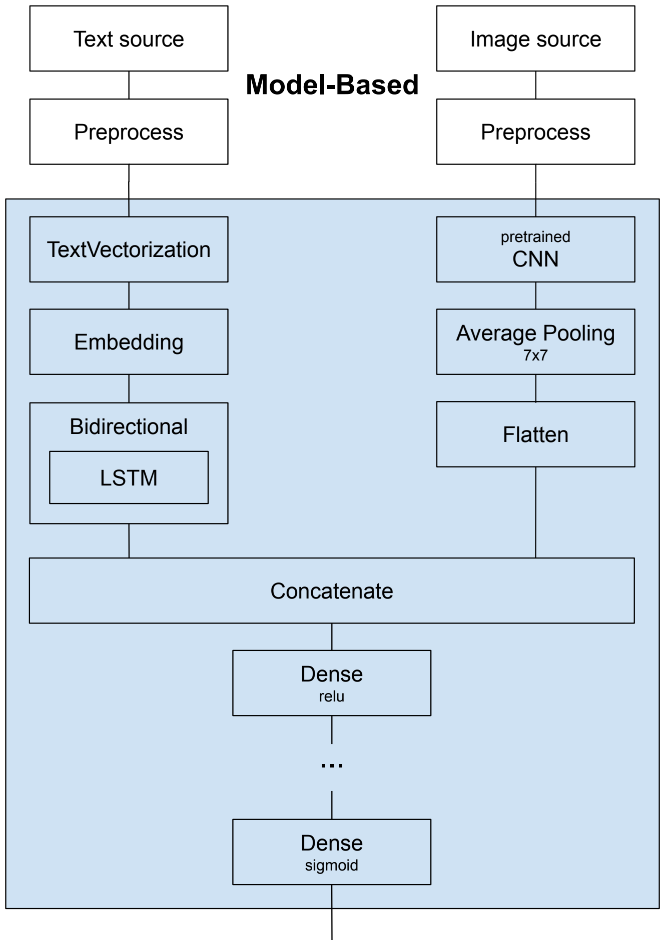


Figure 4.7: Layer structure of the Model-based model. The dots in refers to the variable amount of extra dense layers.

Subsequent testing was then done by the authors using the balance ratio to find the best performing model. Due to constraints from the concatenation, the number of variables which can be varied in this solution is quite limited. Therefore, only separate image transfer models was tested. Otherwise, each model has a large vocabulary size, for the same reason as the previous model, and 3 Dense layers before the classifying layer, as as previous testing showed this was a reasonable balance between complexity and performance. Each model was also trained 3 times and the separate metrics averaged. The final model-based results can be seen in table 5.10.

Following this, analysis was performed on falsely predicted values to get a perception of what kind of images were labeled incorrectly. This was done by randomly selecting a large amount of incorrect values and the authors manually evaluating them to find patterns.

4.5.2 Decision-based

In contrast to model-based approaches, decision-based multimodality uses completed and fully trained models to produce an output.

For each data point, the image is evaluated by a fully trained image classifier while the text is separately evaluated by knowledge engineering rule-set and a text model. The outputs are then used to calculate a weight-based average with pre-defined weights, or through a policy network. The policy network is inspired by the policy network by [Zahavy et al., 2016], and is a simple ANN with an Input layer followed by a variable number of Dense layers with relu activation. The last layer is a Dense layer using sigmoid activation with output size 1. All layers except the last have the same size as the number of inputs.

The policy network was trained with different model inputs and number of extra Dense layers. For the weight-based it was first evaluated for the text and image models with the highest MCC. The weight for the text model was fixed while the image model weight was changed, the results were plotted to determine if the global maximum had been found for all three metrics. Then the weights with the highest MCC was chosen when the output from the knowledge engineering setup was included. The same process as for the image model weight was repeated with the knowledge engineering setup.

Chapter 5
Results

The best results for each single modal method can be seen in table 5.1. The models here are first the random baseline followed by models extracted from the tables 5.3, 5.4 and 5.8 respectively. The names of the models are so that they can be identified in their separate tables. This can therefore also be seen as the baseline for single modality methods when comparing to multimodal variants. In general text classification methods performs better than image recognition methods.

Table 5.1: The best performing model for each single modal method and a baseline by random selection.

Name	Model	Precision	Recall	MCC
BL	Random baseline	0.0317	0.5000	0.0000
KE6	Knowledge engineering	0.5679	**0.7760**	0.6512
BLSTM9	Text model	**0.6414**	0.7154	**0.6662**
CNN4	Image model	0.3786	0.6250	0.4652

The multimodal method with the leading performance was a weighted decision-based model, using all three individual models.

Table 5.2: Weight ratios, precision, recall and MCC for the best performing multimodal method.

BLSTM9 weight	CNN4 weight	KE6 weight	Precision	Recall	MCC
1.00	0.50	0.50	0.6457	0.7904	**0.7042**

Further, this chapter presents the results of all implemented models and experiments. As part of this, the reasoning for which models which were combined into the multimodal solution will also be explained. Therefore, this chapter will be divided into three sections, for each of the classification methods, with the multimodal results last.

5.1 Text Classification

This section will present the result of the three approaches to text classification, knowledge engineering, BLSTM and BERT.

5.1.1 Knowledge Engineering

Table 5.3 present the results from different rule-combinations. The results indicate that r1 and r2 both are able to identify logos individually, where r1 has a slightly higher MCC. KE3 indicates that r1 and r2 are complementary since the combination performs much better than the individual rules. The results also indicate that r3 and r4 doesn't improve the performance while r5 gives a slight improvement. KE6 has the highest MCC of 0.6512 and is therefore considered the best knowledge engineering variation.

Table 5.3: Precision, recall and MCC of some variations of which rules are applied. r*x* refers to the rules defined in section 4.3.2.

Name	Decision rule	Precision	Recall	MCC
KE1	r1	0.4955	0.5260	0.4940
KE2	r2	**0.7654**	0.2510	0.4292
KE3	r1 or r2	0.5593	0.7760	0.6459
KE4	r1 or r3	0.1074	**0.8817**	0.2562
KE5	r1 and r5	0.5055	0.5260	0.4995
KE6	(r1 or r2) and r5	0.5679	0.7760	**0.6512**
KE7	(r1 or r2 or r4) and r5	0.4988	0.7904	0.6131

5.1.2 BLSTM

The results for model size can be found in table 5.4 where different vocabulary sizes and extra dense layers were tested. The model with the highest MCC is BLSTM9, which is the largest model, size-wise. The reason no tests with larger models were conducted was due to time constraints and that the smallest models performing almost as well as the largest model, making it unlikely to yield much better results with larger models.

The results for different sampling methods can be found in table 5.5. *Undersample 1:x* refers to how many negatives there is in the train data set per positives, *oversample scale x* refers to how many times each positive is repeated and no sample refers to the train data set unmodified, which has an class imbalance of approximately 1:30. No clear patterns can be seen for different model size setups as the best model for each metric seem random. Undersampling has a clear pattern where a more balanced data set gives higher recall and lower precision.

Table 5.4: Evaluation of model sizes using an undersampled data set with a 1:12 ratio. The full results can be found in table A.1.

Name	Vocabular size	Extra dense layers	Precision	Recall	MCC
BLSTM1	5000	1	0.6177	0.7163	0.6535
BLSTM2	5000	3	0.6075	**0.7471**	0.6619
BLSTM3	5000	6	0.6032	0.7221	0.6479
BLSTM4	10000	1	0.6037	0.6971	0.6364
BLSTM5	10000	3	0.5816	**0.7471**	0.6467
BLSTM6	10000	6	**0.7174**	0.5394	0.6116
BLSTM7	15000	1	0.6091	0.7087	0.6449
BLSTM8	15000	3	0.6273	0.6798	0.6412
BLSTM9	15000	6	0.6414	0.7154	**0.6662**

Table 5.5: Evaluation of sampling methods for the data set. A vocabulary size of 5000 and 3 extra dense layers were used. The full results can be found in table A.2 and A.3.

Method	Precision	Recall	MCC
Undersample 1:1	0.3658	**0.8673**	0.5436
Undersample 1:3	0.5017	0.8423	0.6360
Undersample 1:6	0.5572	0.7779	0.6454
Undersample 1:9	0.5964	0.7913	**0.6754**
Undersample 1:12	0.6022	0.7337	0.6527
No sample (≈1:30)	**0.6512**	0.6337	0.6308
Oversample scale 5	0.5647	0.8019	0.6605
Oversample scale 10	0.4557	0.6029	0.5063
Oversample scale 30	0.3578	0.6000	0.4410

5.1.3 BERT

The results for different BERT variants can be found in table 5.6. The initial results indicate that the performance of BERT is worse than BLSTM but due to the limitation described in section 4.3.2, many combinations could not be tested.

Table 5.6: Evaluation of different BERT variants. An undersampled data set with the ratio 1:6 was used, since the training was faster with less train data, and 1:6 was usable during BLSTM.

Name	Variant	Precision	Recall	MCC
BERT1	Small bert L-2 H-128 A-2	0.3878	0.0180	0.0780
BERT2	Small bert L-8 H-512 A-8	0.4153	**0.5142**	**0.4427**
BERT3	Small bert L-12 H-768 A-12	0.3981	0.4433	0.4002
BERT4	Electra small	**0.6910**	0.1548	0.3183
BERT5	Albert base	0.4688	0.1010	0.2068

5.1.4 Model Selection

Since the knowledge engineering solution is quite trivial, the best possible model is the one with the highest MCC from table 5.3. Alas, that model and what is to be used in decision-based approaches is KE6.

As BERT was not completely tested, only BLSTM is explored enough to be used in further evaluations and can be declared the best text classification model. Therefore it will be the best performing model from table 5.4. This model is BLSTM9 as the MCC for this is the highest among them.

5.2 Image Recognition

This section will describe the results of the different image recognition models. CNN evaluation was performed on the data set while the object detection methods used website screenshots. Testing showed only using CNN for image classification was producing satisfactory outcomes with the best models having a MCC of around **0.46** with a recall higher than precision at ~ **0.63** and ~ **0.35**, respectively.

5.2.1 Image Classification (CNN)

Table 5.7 show the results of different data set balancing ratios on the CNN model. These were done on identical ResNet-101 model, as these had shown during early testing to provide reasonable baselines.

Table 5.7: Precision, Recall and MCC values from training identical ResNet-101 with different ratios of class balances and random oversample.

Balance ratio	Oversample	Precision	Recall	MCC
1:1	No	0.1939	0.8308	0.3670
1:1	3	0.2110	0.7808	0.3729
1:3	No	0.3262	0.6192	0.4253
1:3	1	0.3041	**0.6692**	**0.4257**
1:7	No	**0.4628**	0.4423	**0.4349**

1:3, oversampled 1:3 and 1:7 all have similar MCC values, but the latter has relatively different precision and recall scores. The large recall score for the 1:3 balances is high enough to outweigh the small difference in MCC between these and 1:7, in combination with the difference in precision being relatively small. Therefore oversampled 1:3 was determined as the best balancing method and will be used for subsequent testing.

Table 5.8 show a subset of Precision, Recall and MCC values from training of several logo image classification models using different transfer models and complexity, with a ratio of 1:3 and oversampling for class balance. This subset has the best performing complexity for each separate transfer model.

Table 5.8: Subset of the evaluation of different complexity and transfer models for the CNN model

Name	Transfer model	Extra dense layers	Precision	Recall	MCC
CNN1	ResNet-50	0	0.3785	0.5769	0.4460
CNN2	ResNet-101	0	0.3604	0.5933	0.4401
CNN3	ResNet-152	4	0.3800	0.6154	0.4624
CNN4	ResNetV2-50	2	0.3786	0.6250	**0.4652**
CNN5	ResNetV2-101	6	0.3813	0.5913	0.4537
CNN6	ResNetV2-152	0	0.2914	**0.7058**	0.4274
CNN7	MobileNetV2	4	0.3779	0.5625	0.4397
CNN8	MobileNetV3 Small	6	0.3495	0.6500	0.4540
CNN9	MobileNetV3 Large	4	0.3143	0.6788	0.4373
CNN10	DenseNet-121	0	0.3362	0.6740	0.4527
CNN11	DenseNet-169	2	0.3289	0.6413	0.4354
CNN12	DenseNet-201	0	**0.3955**	0.5462	0.4443
CNN13	VGG-16	0	0.3356	0.6183	0.4319
CNN14	VGG-19	6	0.3770	0.5510	0.4343

In general, neither model complexity nor choice of transfer model affected the results in any major way. The MCC-values were all slight above 0.4 with the best performing models at ~ 0.46. Precision and recall varied slightly more depending on transfer model and complexity, but no greater correlation was detected.

It was observed subsets of 10 mislabeled images consistently had 8-9 logos, but not for the specific company currently matching to. For example, was often the logo for a partner, subsidiary company, product, or brand detected in addition to the actual company's logo. The remaining images were often minimalist images such as arrows or check marks.

Similar analysis was performed on false negative subsets. These were observed being more complicated logos, with for example more text or many smaller design elements. However, many were also regular logos which in theory should be labeled correctly.

5.2.2 Object Detection

The results for the three methods tested, Bounding box regression, R-CNN and Detection through classification, were all inconclusive and will be discussed further in section 6.2.

5.2.3 Model Selection

As image classification using CNN was the only viable option for detecting the logo using image recognition, this method is what was used in the multimodal approaches.

Since the choice of transfer model did not affect the results in general, any option would be essentially equivalent. Therefore ResNet, MobileNetV3 and DenseNet were chosen to be used for testing the model-based multimodal solution.

For the decision-based approach, ResNetV2-50 with two extra dense layers was selected as architecture for the image recognition section, as this had the highest MCC in addition to the highest recall among the models with similarly high MCC.

5.3 Multimodal Classification

This section will present the results from the different multimodal approaches. First it will show the model-based results, where the models are combined before producing a label, followed by decision-based, which combines the predictions from previous models.

5.3.1 Model-based

Table 5.9 shows precision, recall and MCC values from training identical concatenation models with different ratios of class balances and random oversample.

Table 5.9: Results from training identical concatenation models for different ratios of class balances.

Balance ratio	Oversample	Precision	Recall	MCC
1:1	No	0.1149	0.9471	0.2825
1:1	3	0.4362	0.0788	0.1753
1:3	No	0.4482	**0.7481**	0.5619
1:3	1	**0.5728**	0.5596	0.5522
1:7	No	0.5195	0.7308	**0.6015**

Alike the same test for earlier CNN models 1:3, 1:3 oversampled and 1:7 balance ratio performs the best, however in contrast to that test the MCC for 1:7 is significantly higher than both 1:3 and 1:3 oversampled. The recall for 1:3 is also only slightly higher than 1:7. Therefore the 1:7 balance ratio is determined as the best performing ratio and will be used in subsequent testing.

Table 5.10 shows Precision, Recall and MCC average values from three separate trainings of the text and image concatenation model with different image transfer models.

As can be seen in table 5.10, 5.4 and 5.8 the model-based network normally had higher recall than both BLSTM and CNN models, however lower precision and MCC than BLSTM by itself.

In general, similarly to when using image classification, the choice in transfer model did not greatly affect the final metrics. However, in contrast to those results there is a clear best model, using MobileNetV3 Large. This model does not only produce the highest MCC but also the highest precision. While other models do show higher recall, the difference in MCC compared to other models is enough to determine this as the best possible model. The confusion matrix, which can be seen in figure 5.1, visualizes the results when running this model on the test data set.

Analysis for the negative values for this model was then performed. A distinct pattern was found both in false positive values as well as false negative. When examining a large amount random samples for false positive values it was observed the overwhelming majority of images were logos of companies or brands other than the specifically requested company.

Table 5.10: Average values from three separate trainings of the model-based approach.

Image Transfer model	Precision	Recall	MCC
ResNet-50	0.4453	0.8019	0.5785
ResNet-101	0.4982	0.7801	0.6058
ResNet-152	0.4932	0.7891	0.6066
ResNetV2-50	0.5503	0.6426	0.5665
ResNetV2-101	0.5585	0.6667	0.5958
ResNetV2-152	0.5290	0.6965	0.5872
MobileNetV2	0.5426	0.7247	0.6110
MobileNetV3 Small	0.5778	0.6522	0.5995
MobileNetV3 Large	**0.5970**	0.7099	**0.6362**
DenseNet-121	0.4861	0.8061	0.6106
DenseNet-169	0.4984	0.7910	0.6109
DenseNet-201	0.3733	**0.8333**	0.5327
VGG-16	0.5459	0.7192	0.6103
VGG-19	0.5226	0.6856	0.5822

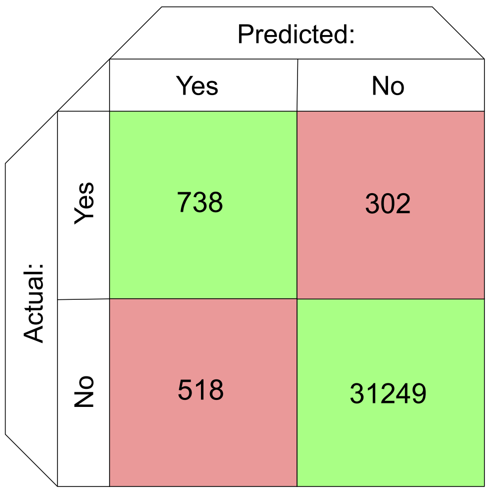


Figure 5.1: The averaged confusion matrix for the model-based multimodal solution from three separate trainings.

For example, subsidiaries for larger conglomerates or specific product brands. Similar examinations of random false negative images showed a majority were upscaled as part of the preprocessing.

5.3.2 Decision-based

This model uses the three best single modal models, which are KE6, BLSTM9, and CNN4.

By using an oracle model, as described in section 2.4.1, the theoretical upper bound was calculated using the different permutations of inputs. The oracle results can be seen in table

5.11. The results indicate that BLSTM9 and CNN4 has the highest complementary information in their output since it has the highest MCC when two models are combined. The combination of BLSTM9 and CNN4 yielded an MCC of 0.8647 which is 0.1985 higher than the best individual model (BLSTM9: 0.6662). While the combination of BLSTM9, CNN4 and KE6 yielded an MCC of 0.8940 which is 0.2278 higher than the best individual model (BLSTM9: 0.6662). This indicates that there is a large potential MCC gain of around 0.2 by using the image model and text model or all inputs.

The result of the policy networks can be seen in table 5.12 and a confusion matrix for the highest MCC is available in figure 5.2. While more input models performed slightly better an increase in model size the affect the model's performance by much, were is was slightly worse for two input models and slightly better for three input models.

The policy models (table 5.12) have worse MCC (best: 0.6255) than the best text-based approaches (BLSTM9: 0.6662), which is a decrease in MCC by 0.0407.

The results for using different weights for weight-based average with a text and image model can be seen in figure 5.3. The weight of BLSTM9 is fixed to 1 while the x-axis is the image model weight e.g., when going toward 0 the BLSTM9 is prioritized, and when going toward infinity the CNN4 is prioritized. The results for fixed weights for the image and text model and variable weight for knowledge engineering can be seen in figure 5.4. The weight of the BSLTM9 is fixed to 1 and the weight for the CNN4 is fixed to 0.5 while the x-axis is the KE6 weight.

Figure 5.3 and 5.4 shows that the weight ratio between BLSTM9 and CNN4 has large effects on the performance of the weight-based model while the inclusion of KE6 only had minor effects. Table 5.13 shows the best weights for every metric and input setup. A confusion matrix for the weights with the highest MCC can be seen in figure 5.5. When only BLSTM9 and CNN4 was included, the best model has an MCC of 0.6903 which is 0.0241 higher than the best individual model (BLSTM: 0.6662), while the model that included all three had an MCC of 0.7042 which is 0.0380 higher than the best individual model (BLSTM9: 0.6662).

Table 5.11: Results from the oracle model. Input models can be seen in table 5.1.

Inputs	Precision	Recall	MCC
BLSTM9 + CNN4	0.8126	0.9298	0.8647
KE6 + CNN4	0.7606	0.9288	0.8349
BLSTM9 + KE6	0.7220	0.8663	0.7835
BLSTM9 + KE6 + CNN4	**0.8412**	**0.9577**	**0.8940**

Table 5.12: Results from the decision-based policy models. Input models can be seen in table 5.1.

Inputs	Extra Dense Layers	Precision	Recall	MCC
BLSTM9 + CNN4	1	**0.6924**	0.5692	0.6169
BLSTM9 + CNN4	2	0.6886	0.5635	0.6119
BLSTM9 + CNN4 + KE6	1	0.6923	0.5798	0.6227
BLSTM9 + CNN4 + KE6	2	0.6860	**0.5904**	**0.6255**

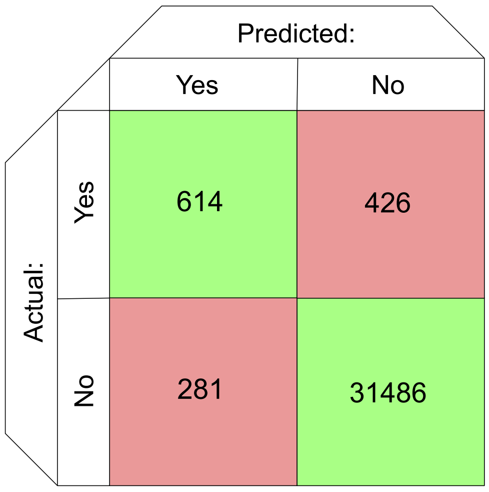


Figure 5.2: The confusion matrix for the policy model with the highest MCC.

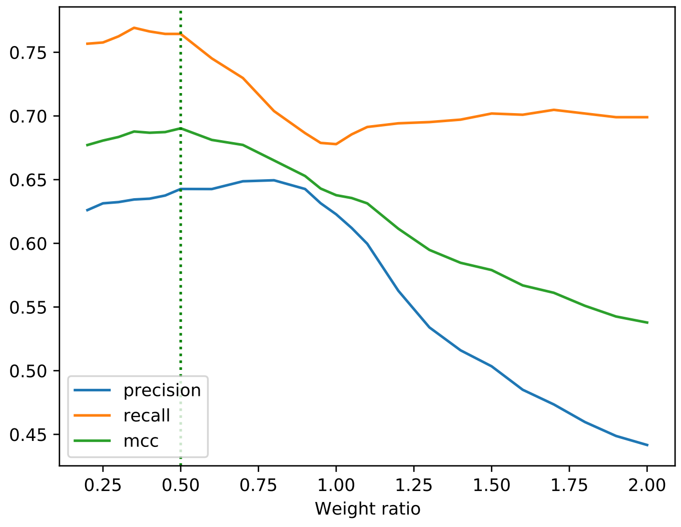


Figure 5.3: A graph showing the precision, recall and MCC using different weight ratios between BLSTM9 and CNN4. MCC is largest at the weight ratio 0.5.

The authors looked at a subset of the misclassified images from both approaches and it was clear to them that false positives are logos from other brands or icons, and all models agree that it is a logo. False negatives appear when the text model has, with strong confidence, classified it as a negative, while the image model has classified it as a positive.

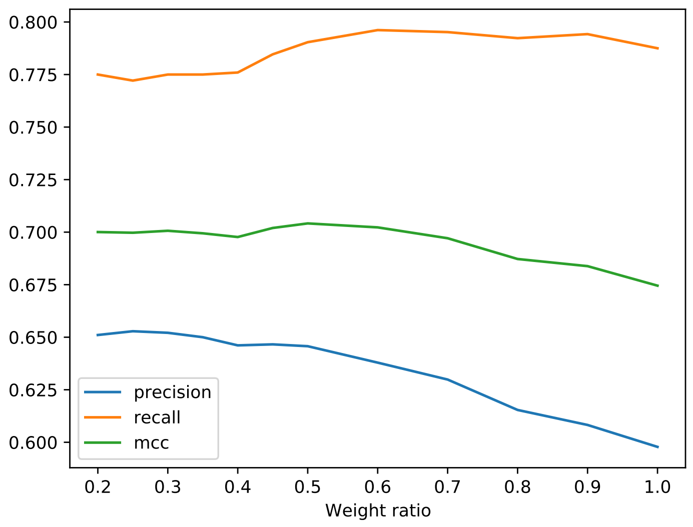
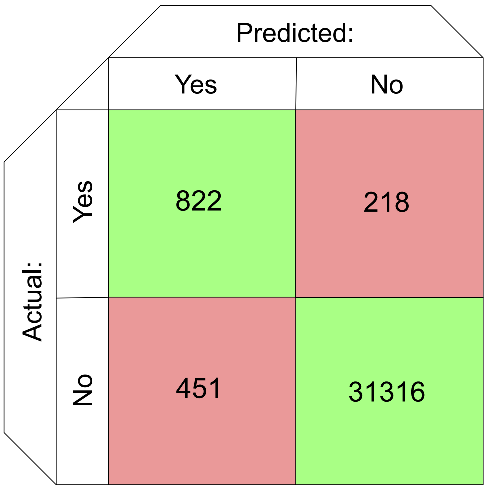


Figure 5.4: A graph showing the precision, recall and MCC using different weights for KE6. MCC is largest at the weight ratio 0.5.

Table 5.13: Weight ratios relative to BLSTM9 with the highest precision, recall and MCC, respectively. Full results can be seen in table C.2.

BLSTM9 weight	CNN4 weight	KE6 weight	Precision	Recall	MCC
1.00	0.35	0.00	0.6344	0.7692	0.6878
1.00	0.50	0.00	0.6427	0.7644	0.6903
1.00	0.80	0.00	0.6495	0.7038	0.6651
1.00	0.50	0.25	**0.6528**	0.7721	0.6997
1.00	0.50	0.50	0.6457	0.7904	**0.7042**
1.00	0.50	0.60	0.6379	**0.7962**	0.7023



		Predicted:	
		Yes	No
Actual:	Yes	822	218
	No	451	31316

Figure 5.5: The confusion matrix for the weight-based multimodal solution with the highest MCC.

Chapter 6
Discussion

This chapter will discuss what has been observed and determined during the thesis. It will begin discussing and evaluating the results which were presented in the previous chapter. After that, the problems and possible solutions to object detection in this application will be described, before ending with general limitation for this thesis.

6.1 Results discussion

In the following section, the results from the previous chapter will be discussed to form a basis to which answer the research questions presented for this thesis. The different classification methods will be discussed in the same order as they are presented in chapter 5.

6.1.1 Text Classification

As seen in table 5.1, text classification proved to have a higher performance than image classification, especially precision and MCC. The knowledge engineering approach and BLSTM were able to produce similar MCC while BLSTM produced better precision and knowledge engineering produced better recall. BERT was not able to match the other text classification methods, but due to the limited testing a much better result might be able to be reached.

Notable for the false negatives were that the text model mostly was confident in its guess. A possible approach to tackle this could be to gather more metadata for every image. For example, neither the DOM path nor the id and classes for the `<img/>` tag were collected. This could potentially improve the performance of the text classification, since it might contain complementary information when the image source doesn't include data that indicates it is a logo.

The performance of both the knowledge engineering and BLSTM model were close to the best models making them reasonable approaches to detect a specific company's logo.

6.1.2 Image Recognition

The results for the object detection methods tried in this thesis were inconclusive in finding a company's logo from their website, as can be seen in section 5.2.2. This will be discussed further in section 6.2, so this section will only focus on image classification.

Using a standard CNN with transfer learning to find a company's logo is possible, but arguable in its performance. The main issue with this approach is context. The model itself has no information about what it is trying to detect except for the image itself. Therefore, it performs very well in finding logos in general, but not for specific companies. This can be observed when analysing the false positive values from this method, which can be seen in section 5.2.1, where the overwhelming majority of these values were logos, just not for the specified company.

This lack of context could in turn cause other issues. As several false negative images were logos which should have been detected, there is a high possibility unwanted logos affected the training process negatively. The CNN model therefore performs worse at both being a pure logo classifier *and* a specific logo classifier.

There is accordingly no reason to use a single modal CNN model to detect a specific company's logo.

6.1.3 Multimodal Classification

When analysing the oracle model in table 5.11, it becomes clear using multimodality in this application has a high potential of achieving better results than the individual models due to the very metrics. When combining just the text model and image model through the oracle, as seen in table 5.11, the MCC increases significantly compared to the two models individually. This indicates the two modalities complement each other admirably and a multimodal approach should be possible.

However, none of the techniques tested in this thesis were able to achieve the potential presented by the oracle. Using weighted decision-based multimodal models did provide higher MCC than either individual model, but still far away from the oracle's potential.

As can be seen in 5.10, 5.12 and 5.1, the other multimodal approaches did not manage to attain an MCC higher than the best performing single modal text model, but it did come close. This indicates those techniques are similar in viability to the text model as it could still used in practice with about equal results. Therefore, multimodal approaches in general are viable options, and further evaluation would be needed to declare whether they perform better in a business sense, which means how they perform at specific product tasks like the ones presented in chapter 1, instead of a classification sense.

Reasons for why no multimodal model were able to achieve the oracle's potential are hard to pinpoint, however it can be broken down into two categories: Differences and similarities in the single modal models which were combined.

When looking at differences, the combined models differ quite greatly. Examples of major differences between the models are hyperparameters and complexity, which are two things which can have major effects on the final results. The CNN model also uses transfer learning while the BLSTM model is built completely from scratch which could severely affect the training of model-based approaches. It might therefore not be as simple to combine several greatly different models as initially believed. Deeper analysis in combining models of different complexity would have to be done to find correlations between that complexity and better results.

Due to the limitations while training BERT it was not used during evaluation of the multimodal approaches. A theory is that BERT would be a better fit than BLSTM for the model-based approach. This is due to both the image models and BERT using transfer learning. This would make it more likely to find hyperparameters that fit both individual models during training, as they would have similar complexity. Unfortunately, the authors were not able to test this.

Similarities come more in analysing the individual classifications. For example, both the text and image models had issues with excluding other logos and brands other than the one specifically wanted, which can be seen in section 5.1 and 5.2.1. This issue was then observed further in all the multimodal approaches, as described in 5.3. Similar gaps in complement might be larger than the oracle can notice. For example, when one model is barely right while the other is very wrong, the oracle will predict the result as correct while no combined model would ever produce the proper value.

However, as several multimodal models using weighted decisions had higher MCC than any single modal model, which can be seen in table 5.1 and 5.13, it is clear multimodality is a valid approach even in the classification sense.

6.2 Problems with Object Detection

Early in the process it became clear image classification was the only truly viable method of finding the logo. All object detection methods showed individual issues which caused them to be dropped from consideration and further evaluation after initial testing. This section will describe these problems, their causes and what eventually lead them to be impractical in this thesis.

6.2.1 Bounding Box Regression

Early in testing it became clear this method had problems. The model correctly started predicting the location of logos to be in the top left corner, which is standard for western websites, but the predicted boxes were never consistently close to the ground truth boxes. Whenever a logo was not in the top left, the implemented model would still predict the location to be there, indicating no image analysis occurred.

The issue which presented itself was the size of both the image and the logo. Since bounding box regression always assumes a logo is present, the input must be an image where that is true. Therefore, a screenshot of the site was used as it is expected to contain the logo. The previously mentioned issue then manifested itself.

As most CNN algorithms, including the transfer learning methods used in this model, resizes the input image to specific size, the resolution of the screenshot was too large in comparison. This resize, combined with the logo usually being a small part of the entire site, caused it to be too small to be detectable by the CNN. This is what caused the model to always predict the logo to be in the upper left corner. Therefore, usage of bounding box regression in this context, with large images with a small desired object, is not possible. The algorithm worked as expected, but the context and implementation were not compatible.

A solution to this could have been using a sliding window as described in section 2.3.2 as it processes parts of the image. However due to the caveat that the bounding box regression model assumes an object is present this became impossible, as it would have returned possible bounding boxes for every sliding window. There would then be no indication which of these boxes were false positives without manual confirmation, which in turn makes this method unusable. If the assumption of an object being present was removed, it would no longer follow the algorithm of bounding box regression.

6.2.2 R-CNN

The R-CNN model identified essentially any unicolour shape or word on another unicolour background as a logo. As this is one of the most common logo designs, this indicates there are too few counterexamples of logos when training the model. Even though R-CNN detects the actual logo in many cases, the number of false positives greatly outweighs those predictions.

As is done when using R-CNN as described in section 2.3.2, the data set used to train this model was generated by running selective search on a large number of website screenshots, with ground truth boxes to separate the positive and negative data points. As detailed in section 2.3.2, running selective search gathers a great number of possible images, which in itself is positive. Each screenshot where selective search found subimages matching the ground truth produced about 1-5 positive images. However, since data set balance is crucial for binary classifiers, most negative data points had to be removed, leaving only 1-5 per positive.

The negative images were widely different, some examples of which can be seen in figure 6.1, which was expected. However, since most of these were discarded the data set quickly became smaller and more inconsistent. This is turn caused the results to be inconsistent. As indicated by the results of data set balance testing on the normal CNN model in table 5.7, the balance ratio achieved in here was believed by the authors to be enough to achieve consistent results, but that was not the case. Adding further negative data points still produced inconsistent results up to a point where the authors decided to stop evacuating this approach.

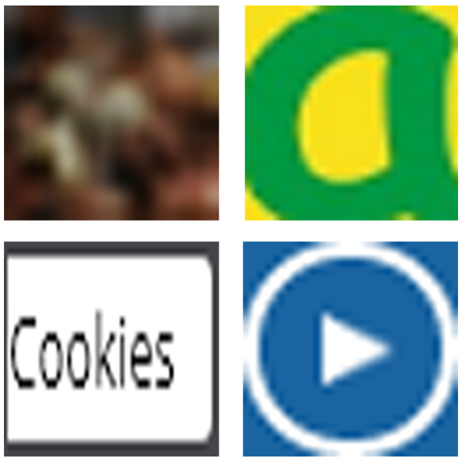


Figure 6.1: Four examples of negative images found by using selective search on website screenshots when using ground truth boxes for positives.

To solve this issue, another selective search was run by the authors to get a more precise data set. This time, ground truth boxes were added by the authors for parts of the screenshots which were previously incorrectly labeled as logos by the R-CNN model. These ground truth boxes were then instead labeled as negative, which means they were significantly more likely to be part of the negative data set and therefore used as counterexamples when training the R-CNN model. This however did not solve the problem, which implies the data set is too small and inconsistent, as websites has too many differing elements. As then continued usage of R-CNN would necessitate further increasing the data set while not guaranteeing improved results, this method also fell through and no deeper evaluation was completed.

6.2.3 Detection Through Classification

As this method requires an already trained classifier, it used the best performing CNN from section 5.2, CNN4. With this model, the bounding boxes discovered seldom matched the ground truth. Often parts of the website's background were part of the discovered box, or the logo itself was cropped. The model also had inconsistent results when analysing part of sites which were not images at all. As a previously fully trained model is used in this approach, no changes could be made to fix this. The authors therefore determined there were two larger issues with this method.

Firstly, this method's predicted bounding boxes seldom matches the ground truth. Even when a company logo was detected, major part of the background was often detected, or the logo cropped. In an attempt to solve this, different sizes of the sliding window, image pyramid ratios and propagation rates were tested. However, it quickly became clear the inconsistency of where the logo was located and its size on the screenshot was too big between different websites. This made it an impossible task of trial and error to find the sweet spot where the model would consistently produce an output where this issue was prominent.

Secondly, as the classifier used only trained on implemented images, there were several parts of screenshots it had never seen anything close to. Examples of this are empty space or large amount of text. When this occurs, the output itself becomes inconsistent and it performs worse than the classifier by itself. Even on runs on basic sites, this model returned many false positives.

These issues in tandem were enough to warrant the decision by the authors not to continue with further evaluation.

6.3 Limitations

The biggest limitation in this thesis is the data set. Using Forbes Global 2000 had positive attributes such as having a diverse number of business fields as well as having companies from different countries and cultures. The issue however is when using the 2000 biggest companies, many of these are conglomerates, with several subsidiaries, and mega-corporations, which own a multitude of products and brands. The websites for these however usually contain the logos for these subsidiaries, brands and products, which is exactly the false positives the designed models struggle with. Therefore, it is highly likely the results are skewed negatively due to the test set used, and any specific model could perform better on a more heavily curated data set. It is believed by the authors the models perform very well on more basic websites or smaller companies as the risk of having other logos decreases. It is however not certain a different data set would show any significant improvement when using multimodality over single modal approaches.

There is also no evaluation for if a logo is found on a specific site. For example, when used in practice it could be completely satisfactory if the model finds a high-resolution logo while failing to detect one of lower resolution which was upscaled during preprocessing. However, when evaluating the results in this thesis, missing a low quality logo is evaluated just as bad as missing the only logo on a site. Therefore, it could be interesting to have some kind of evaluation and metric for how many correct logos are detected for each individual site. For example, finding one out of two logos on a specific website could be better than finding zero logos when only one is available. There is also an interest in prioritizing images of higher resolution as they are most likely preferable to the user, but this as well is out of scope for this thesis.

This is directly related to using only MCC, precision and recall as evaluation metrics. These do not tell the whole story. Just because MCC is similar between different solutions does not necessarily mean they are equivalent. It shows there is no obvious advantage to using one model over the other in a classification sense, which is the crux of this thesis.

An important limitation when using any form of machine learning is processing power. Due to the ongoing COVID-19 pandemic, the models used in this thesis were implemented, trained and tested on home computers, and it was not possible to use larger cloud systems or computing power through AFRY. This meant training different models could take anywhere between 15 minutes to several days. Therefore, the extremely high amount of processing power training neural networks require caused a lower number of tests and possible models to be made. For example, there is a large amount of possible hyperparameters for every single model created, but the limited processing power and time made it impossible to optimize the models completely.

Chapter 7
Conclusion

Following the work done in this thesis, the answers to the research questions are the following:

- *Which text classification methods can be used to identify a company logo from their website and how do they perform?*

  Both knowledge engineering and BLSTM produced results with metrics within the same range, with an MCC of ~ 0.66. This, in combination with false image analysis, was determined by the authors to be satisfactory and be used in practice. In small scale testing, BERT performed worse than the two other methods, but technical difficulties made it impossible to draw further conclusions than that. These results can be viewed in section 5.1.

- *Which image recognition methods can be used to identify a company logo from their website and how do they perform?*

  When analysing image recognition methods, it quickly became clear only image classification was a viable option, with object detection having too many problems, described in section 6.2. Image classification using CNN and transfer learning performed worse than its text counterpart with an MCC of ~ 0.47, which can be seen in section 5.2. However, it could not differentiate between the specified websites logo and other logos or brands. This is due to lack of context which is crucial in this kind of specific classification. Therefore, it is concluded there is no reason to use this kind of pure image classification when trying to find a specific logo, and a more general CNN should be used instead.

- *Which multimodal methods can be used to identify a company logo through their website and how do they perform compared to the other individual models?*

  Model-based as well as policy and weighted decision-based multimodal methods were all able to identify company logos though their website, as can be seen in section 5.3. Multimodality also showed high potential when using an oracle model. According to the oracle, the multimodal combination had an MCC upper bound of 0.8940, which can be seen in section 5.3. This is very high in its own right, as an MCC of 1 indicates a perfect classifier, and significantly higher than individual single modal models used to achieve this, at ∼ 0.66 and ∼ 0.47. Even though it was not possible to reach that limit in this thesis, it does show how multimodality can achieve better results by using their single modal counterparts.

When compared to the individual models, the multimodal models all performed about equal to the knowledge engineering and BLSTM models, and much better than just image classification on its own. The exception to this is the weight-based decision-based model which produces better results than any single modal model with an MCC of 0.7042, higher than both the best text classifier at 0.6662 and the best image classifier at 0.4652. This, in combination with the great potential from the oracle, makes it clear finding a company logo through their website is a good application of multimodality.

This master's thesis has shown that text classification and multimodal methods are possible options of classifying specific company logos using the images from their website, while image recognition is not. This has been done by implementing and testing several different models, from machine learning to knowledge engineering, in these different categories. These methods have then been compared to each other using precision, recall, MCC and confusion matrices to estimate their performance.

There is still further analysis to be done to find how these different methods would perform in other applications compared to just pure classification. For example, prioritizing high resolution images or making sure at least one image per website is found. However, we are hopeful this thesis will be a good first step and proof-of-concept for multimodality in this context and other classification problems.

Future work

Adding another form of evaluation to assess the models from other approaches would be interesting. This could also be expanded into having a larger and more diverse data set, in the sense that both larger corporations and smaller local business were included.

Another interesting possibility would be to add some other form of modality to the process to provide context about the specific company whose logo is searched for. As the models in general had problems filtering out other brands or company logos from websites, adding more context could possibility remove a large chunk of the false positive values.

In a multimodal sense, more studies into model-based versus decision-based approaches would be exciting. For example, which use cases one performs better than the other and more concrete pros and cons to those approaches as well as more deep dives into them specifically.

References

[Adelson et al., 1984] Adelson, E. H., Anderson, C. H., Bergen, J. R., Burt, P. J., and Ogden, J. M. (1984). Pyramid methods in image processing. *RCA engineer*, 29(6):33–41.

[Albawi et al., 2017] Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee.

[Andrea Murphy and Touryalai, 2020] Andrea Murphy, Hank Tucker, M. C. and Touryalai, H. (2020). Global 2000 - the world's largest public companies 2020. `https://www.forbes.com/global2000/`.

[Andrychowicz et al., 2020] Andrychowicz, O. M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., et al. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

[Baltrušaitis et al., 2018] Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2018). Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

[Brownlee, 2018] Brownlee, J. (2018). A gentle introduction to early stopping to avoid overtraining neural networks. `https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/`.

[Buckland and Gey, 1994] Buckland, M. and Gey, F. (1994). The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.

[Buda et al., 2018] Buda, M., Maki, A., and Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259.

[Chicco and Jurman, 2020] Chicco, D. and Jurman, G. (2020). The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13.

[Clark et al., 2020] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

[Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Gao et al., 2015] Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., and Xu, W. (2015). Are you talking to a machine? dataset and methods for multilingual image question answering.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[Godoy, 2018] Godoy, D. (2018). Understanding binary cross-entropy / log loss: a visual explanation. `https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a`.

[Hayes et al., 1990] Hayes, P. J., Andersen, P. M., Nirenburg, I. B., and Schmandt, L. M. (1990). Tcs: a shell for content-based text categorization. In *Sixth Conference on Artificial Intelligence for Applications*, pages 320–326 vol.1.

[He and Garcia, 2009] He, H. and Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284.

[He et al., 2017] He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708.

[Kourou et al., 2015] Kourou, K., Exarchos, T. P., Exarchos, K. P., Karamouzis, M. V., and Fotiadis, D. I. (2015). Machine learning applications in cancer prognosis and prediction. *Computational and structural biotechnology journal*, 13:8–17.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

[Lampert et al., 2008] Lampert, C. H., Blaschko, M. B., and Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE.

[Lan et al., 2019] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

[Lee et al., 2018] Lee, S., Kwak, S., and Cho, M. (2018). Universal bounding box regression and its applications. In *Asian Conference on Computer Vision*, pages 373–387. Springer.

[Lever et al., 2016] Lever, J., Krzywinski, M., and Altman, N. (2016). Classification evaluation. *Nature Methods*, 13(8):603–604.

[Liu et al., 2017] Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[Lu and Weng, 2007] Lu, D. and Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International journal of Remote sensing*, 28(5):823–870.

[Mitchell, 2018] Mitchell, R. (2018). *Web scraping with Python: Collecting more data from the modern web*. " O'Reilly Media, Inc.".

[Ong, 2017] Ong, T. (2017). Facebook's translations are now powered completely by ai. `https://www.theverge.com/2017/8/4/16093872/facebook-ai-translations-artificial-intelligence`.

[O'Shea and Nash, 2015] O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[Pan and Yang, 2009] Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[Perez and Wang, 2017] Perez, L. and Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`.

[Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.

[Rezatofighi et al., 2019] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 658–666.

[Ruder, 2016] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

[Sak et al., 2015] Sak, H., Senior, A., Rao, K., Beaufays, F., and Schalkwyk, J. (2015). Google voice search: faster and more accurate. `https://ai.googleblog.com/2015/09/google-voice-search-faster-and-more.html`.

[Sandler et al., 2018] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47.

[Shah, 2017] Shah, T. (2017). About train, validation and test sets in machine learning. `https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7`.

[Shaha and Pawar, 2018] Shaha, M. and Pawar, M. (2018). Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660. IEEE.

[Shin et al., 2016] Shin, H.-C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., and Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[Stanford, 2019] Stanford, S. (2019). Deepmind's ai, alphastar showcases significant progress towards agi. *Medium*. `https://medium.com/@stanford__ai/deepminds-ai-alphastar-showcases-significant-progress-towards-agi-93810c94fbe9`.

[Szeliski, 2010] Szeliski, R. (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.

[Tajbakhsh et al., 2016] Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312.

[Toro-Vizcarrondo and Wallace, 1968] Toro-Vizcarrondo, C. and Wallace, T. D. (1968). A test of the mean square error criterion for restrictions in linear regression. *Journal of the American Statistical Association*, 63(322):558–572.

[Torres, 2020] Torres, J. (2020). Learning process of a deep neural network. `https://towardsdatascience.com/learning-process-of-a-deep-neural-network-5a9768d7a651`.

[Turc et al., 2019] Turc, I., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

[Uijlings et al., 2013] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

[Vogels, 2016] Vogels, W. (2016). Bringing the magic of amazon ai and alexa to apps on aws. `https://www.allthingsdistributed.com/2016/11/amazon-ai-and-alexa-for-all-aws-apps.html`.

[Wang et al., 2018] Wang, J.-H., Liu, T.-W., Luo, X., and Wang, L. (2018). An lstm approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*, pages 214–223.

[Wang and Su, 2019] Wang, S. and Su, Z. (2019). Metamorphic testing for object detection systems. `https://arxiv.org/abs/1912.12162`.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

[Yuhas et al., 1989] Yuhas, B. P., Goldstein, M. H., and Sejnowski, T. J. (1989). Integration of acoustic and visual speech signals using neural networks. *IEEE Communications Magazine*, 27(11):65–71.

[Zahavy et al., 2016] Zahavy, T., Magnani, A., Krishnan, A., and Mannor, S. (2016). Is a picture worth a thousand words? a deep multi-modal fusion architecture for product classification in e-commerce. *arXiv preprint arXiv:1611.09534*.

[Zhou et al., 2016] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., and Xu, B. (2016). Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.

[Zupan, 1994] Zupan, J. (1994). Introduction to artificial neural network (ann) methods: what they are and how to use them. *Acta Chimica Slovenica*, 41:327–327.

Appendices

Appendix A
BLSTM model testing

The full results from testing different BLSTM setups. Table A.1 shows all models produced while testing different vocabulary sizes and number oif extra dense layers. Table A.2 shows all models produced while testing different undersampling ratios and table A.3 shows all models produced while testing different oversampling scales.

Table A.1: Full results of the BLSTM model size testing with an undersampled data set with a 1:12 ratio. * #True positives + #False negatives = 0 resulting in undefined precision and MCC.

Iteration	Vocabulary size	Extra dense layers	Precision	Recall	MCC
1	5000	1	0.6177	0.7163	0.6535
2	5000	1	0.5844	0.7356	0.6431
3	5000	1	0.5917	0.7010	0.6314
1	5000	3	0.6340	0.6846	0.6472
2	5000	3	0.6075	0.7471	0.6619
3	5000	3	0.6176	0.7144	0.6525
1	5000	6	0.7679	0.0827	0.2459
2	5000	6	0.5630	0.6144	0.5741
3	5000	6	0.6032	0.7221	0.6479
1	10000	1	0.6037	0.6971	0.6364
2	10000	1	0.5990	0.6635	0.6177
3	10000	1	0.5814	0.7144	0.6317
1	10000	3	0.6052	0.6942	0.6359
2	10000	3	0.5832	0.7413	0.6450
3	10000	3	0.5816	0.7471	0.6467
1	10000	6	0.6808	0.4942	0.5686
2	10000	6	0.7174	0.5394	0.6116
3	10000	6	UNDEF*	0.0000	UNDEF*
1	15000	1	0.5322	0.7144	0.6023

2	15000	1	0.6091	0.7087	0.6449
3	15000	1	0.5341	0.6625	0.5802
1	15000	3	0.6273	0.6798	0.6412
2	15000	3	0.6702	0.6058	0.6259
3	15000	3	0.6242	0.6212	0.6103
1	15000	6	0.6414	0.7154	0.6662
2	15000	6	0.7551	0.3231	0.4841
3	15000	6	0.6260	0.6471	0.6244

Table A.2: Full results of testing different undersampled data sets on a model with a vocabulary size of 5000 and 3 extra dense layer was used. The last setup used no undersampling giving a ratio of approximately 1:30.

Iteration	Scale	Precision	Recall	MCC
1	1:1	0.3080	0.8673	0.4937
2	1:1	0.3485	0.8692	0.5298
3	1:1	0.3658	0.8673	0.5436
1	1:3	0.4819	0.8452	0.6235
2	1:3	0.4789	0.8385	0.6187
3	1:3	0.5017	0.8423	0.6360
1	1:6	0.4733	0.8000	0.5997
2	1:6	0.5477	0.7837	0.6420
3	1:6	0.5572	0.7779	0.6454
1	1:9	0.5436	0.7731	0.6349
2	1:9	0.5964	0.7913	0.6754
3	1:9	0.4572	0.7596	0.5726
1	1:12	0.6047	0.7250	0.6500
2	1:12	0.6022	0.7337	0.6527
3	1:12	0.5696	0.7433	0.6378
1	≈1:30	0.6712	0.6106	0.6290
2	≈1:30	0.6853	0.4587	0.5493
3	≈1:30	0.6512	0.6337	0.6308

Table A.3: Full results of testing different oversampled data sets on a model with a vocabulary size of 5000 and 3 extra dense layer was used.

Iteration	Scale	Precision	Recall	MCC
1	5	0.5647	0.8019	0.6605
2	5	0.5482	0.7375	0.6222
3	5	0.5289	0.6240	0.5594
1	10	0.4344	0.5952	0.4898
2	10	0.3988	0.6269	0.4798
3	10	0.4557	0.6029	0.5063
1	30	0.3482	0.5327	0.4078

2	30	0.3235	0.6106	0.4201
3	30	0.3578	0.6000	0.4410

Appendix B
CNN model testing

Table B.1 shows the full results for the CNN fine tuning with different transfer models and number of extra dense layers.

Table B.1: Full results of the CNN fine tuning testing with a 1:3 ratio and 1 random oversample per positive image.

Transfer model	Extra dense layers	Precision	Recall	MCC
ResNet-50	0	0.3785	0.5769	0.4460
ResNet-50	2	0.3566	0.6038	0.4416
ResNet-50	4	0.3065	0.6702	0.4280
ResNet-50	6	0.3105	0.6577	0.4269
ResNet-101	0	0.3604	0.5933	0.4401
ResNet-101	2	0.3080	0.6308	0.4154
ResNet-101	4	0.2594	0.7231	0.4045
ResNet-101	6	0.3954	0.5250	0.4351
ResNet-152	0	0.3516	0.6115	0.4410
ResNet-152	2	0.3565	0.6423	0.4562
ResNet-152	4	0.3800	0.6154	0.4624
ResNet-152	6	0.3491	0.5885	0.4304
ResNetV2-50	0	0.3338	0.6712	0.4498
ResNetV2-50	2	0.3786	0.6250	0.4652
ResNetV2-50	4	0.3002	0.7048	0.4345
ResNetV2-50	6	0.3517	0.6327	0.4491
ResNetV2-101	0	0.3124	0.6798	0.4361
ResNetV2-101	2	0.2614	0.7500	0.4146
ResNetV2-101	4	0.3007	0.7067	0.4356
ResNetV2-101	6	0.3813	0.5913	0.4537
ResNetV2-152	0	0.2914	0.7058	0.4274
ResNetV2-152	2	0.2445	0.7510	0.3988

ResNetV2-152	4	0.3059	0.6692	0.4272
ResNetV2-152	6	0.2586	0.7481	0.4114
MobileNetV2	0	0.3580	0.5673	0.4283
MobileNetV2	2	0.3250	0.6135	0.4223
MobileNetV2	4	0.3779	0.5625	0.4397
MobileNetV2	6	0.2961	0.6192	0.4020
MobileNetV3 Small	0	0.3257	0.6808	0.4469
MobileNetV3 Small	2	0.3403	0.6288	0.4393
MobileNetV3 Small	4	0.3582	0.6240	0.4505
MobileNetV3 Small	6	0.3495	0.6500	0.4540
MobileNetV3 Large	0	0.2372	0.7240	0.3838
MobileNetV3 Large	2	0.3269	0.6317	0.4303
MobileNetV3 Large	4	0.3143	0.6788	0.4373
MobileNetV3 Large	6	0.2911	0.6212	0.3987
DenseNet-121	0	0.3362	0.6740	0.4527
DenseNet-121	2	0.3727	0.5981	0.4505
DenseNet-121	4	0.3349	0.6731	0.4514
DenseNet-121	6	0.3167	0.6596	0.4324
DenseNet-169	0	0.3104	0.6779	0.4338
DenseNet-169	2	0.3289	0.6413	0.4354
DenseNet-169	4	0.3187	0.6365	0.4258
DenseNet-169	6	0.2860	0.6933	0.4187
DenseNet-201	0	0.3955	0.5462	0.4443
DenseNet-201	2	0.2598	0.7606	0.4163
DenseNet-201	4	0.2968	0.6750	0.4217
DenseNet-201	6	0.2717	0.7240	0.4161
VGG-16	0	0.3356	0.6183	0.4319
VGG-16	2	0.3572	0.5135	0.4059
VGG-16	4	0.3134	0.6346	0.4210
VGG-16	6	0.3670	0.5519	0.4281
VGG-19	0	0.3741	0.5385	0.4272
VGG-19	2	0.2897	0.6740	0.4154
VGG-19	4	0.3294	0.6115	0.4248
VGG-19	6	0.3770	0.5510	0.4343

Appendix C
Decision based model testing

Table C.1 shows the full results for the policy model using different input models and model sizes. Table C.2 shows the full results for the weight-based model using different weights for the image model and knowledge engineering model.

Table C.1: Full results of testing different variations of the policy model. * #True positives + #False negatives = 0 resulting in undefined precision and MCC.

Iteration	Input models	Hidden layers	Precision	Recall	MCC
1	BLSTM9 + CNN4	1	0.6924	0.5692	0.6169
2	BLSTM9 + CNN4	1	0.6885	0.5654	0.6129
3	BLSTM9 + CNN4	1	0.6875	0.5606	0.6097
1	BLSTM9 + CNN4	2	0.6886	0.5635	0.6119
2	BLSTM9 + CNN4	2	UNDEF*	0.0000	UNDEF*
3	BLSTM9 + CNN4	2	UNDEF*	0.0000	UNDEF*
1	BLSTM9 + CNN4 + KE6	1	0.7014	0.5308	0.5993
2	BLSTM9 + CNN4 + KE6	1	0.6963	0.5577	0.6123
3	BLSTM9 + CNN4 + KE6	1	0.6923	0.5798	0.6227
1	BLSTM9 + CNN4 + KE6	2	0.6974	0.5740	0.6219
2	BLSTM9 + CNN4 + KE6	2	0.6860	0.5904	0.6255
3	BLSTM9 + CNN4 + KE6	2	0.6972	0.5423	0.6039

Table C.2: Full results of trying different weight ratios between the text and image model. The text model has a fixed weight of 1.

CNN4 weight	KE6 weight	Precision	Recall	MCC
0.20	0.00	0.6261	0.7567	0.6772
0.25	0.00	0.6314	0.7577	0.6807

0.30	0.00	0.6324	0.7625	0.6835
0.35	0.00	0.6344	0.7692	0.6878
0.40	0.00	0.6351	0.7663	0.6868
0.45	0.00	0.6375	0.7644	0.6873
0.50	0.00	0.6427	0.7644	0.6903
0.60	0.00	0.6426	0.7452	0.6812
0.70	0.00	0.6487	0.7298	0.6773
0.80	0.00	0.6495	0.7038	0.6651
0.90	0.00	0.6427	0.6865	0.6529
0.95	0.00	0.6315	0.6788	0.6430
1.00	0.00	0.6228	0.6779	0.6378
1.05	0.00	0.6120	0.6856	0.6356
1.10	0.00	0.5997	0.6913	0.6314
1.20	0.00	0.5627	0.6942	0.6115
1.30	0.00	0.5340	0.6952	0.5948
1.40	0.00	0.5160	0.6971	0.5847
1.50	0.00	0.5034	0.7019	0.5790
1.60	0.00	0.4850	0.7010	0.5670
1.70	0.00	0.4735	0.7048	0.5612
1.80	0.00	0.4597	0.7019	0.5510
1.90	0.00	0.4488	0.6990	0.5425
2.00	0.00	0.4417	0.6990	0.5378
0.50	0.20	0.6511	0.7750	0.7000
0.50	0.25	0.6528	0.7721	0.6997
0.50	0.30	0.6521	0.7750	0.7006
0.50	0.35	0.6500	0.7750	0.6994
0.50	0.40	0.6461	0.7760	0.6977
0.50	0.45	0.6466	0.7846	0.7020
0.50	0.50	0.6457	0.7904	0.7042
0.50	0.60	0.6379	0.7962	0.7023
0.50	0.70	0.6299	0.7952	0.6971
0.50	0.80	0.6154	0.7923	0.6872
0.50	0.90	0.6082	0.7942	0.6838
0.50	1.00	0.5978	0.7875	0.6745

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2021-05-07

EXAMENSARBETE Finding company logos through their websites: A multimodal approach
STUDENTER Emil Wihlander, Jesper Berg
HANDLEDARE Marcus Klang (LTH)
EXAMINATOR Flavius Gruian (LTH)

Hitta företagsloggor med maskininlärning genom text och bild

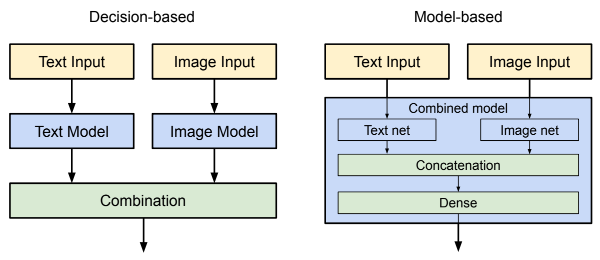
POPULÄRVETENSKAPLIG SAMMANFATTNING **Emil Wihlander, Jesper Berg**

Att automatiskt hämta ett företags logga från deras hemsida kan vara användbart i flera sammanhang. Men hur vet en dator vilken av alla bilder på hemsidan som är loggan? Detta arbete utforskar olika tekniker för att identifiera loggan, inklusive *multimodalitet* vilket använder både bild och text data för att hitta rätt.

Ibland finns flera typer av indata tillgängligt när något ska tolkas, till exempel använder människor både munrörelser och ljud när de tolkar tal. Detta kallas för multimodalitet. Multimodalitet har länge varit intressant i samband med maskininlärning och flera fall finns där det varit lyckat att använda sig av det. Detta eftersom de olika modaliteterna kan innehålla kompletterande information. Målet med detta arbete är att låta datorn avgöra vilka bilder på en hemsida som är företagets logga – men varför är detta problem multimodalt? Jo, bilder på hemsidor har, utöver bilden i sig, även en sökväg som webbläsaren använder sig av för att läsa in bilden. Detta innebär att det finns både bilden i sig samt en text som representerar varje hemsidebild.

I examensarbetet jämförs olika metoder för att låta datorn avgöra bilderna och texterna separat. De metoderna som testas är enkla tekniker så som "innehåller sökvägen ordet *logo*" samt maskininlärning för både bild och text. Efter detta kombinerades modaliteterna genom att kombinera resultatet från de individuella metoderna med ett viktbaserat medel och med maskininlärning. En stor modell som tar både bilderna och texterna som indata testades också.

Bilden nedan visar hur de olika sätten att kombinera modaliteterna ser ut. Den vänstra visar när resultatet från individuella modeller kombineras med en beslutsmodell och den högra hur en stor modell med båda indata ser ut.



Resultatet visar att textklassificering, både genom enkla metoder så väl som maskininlärning, lyckas hitta loggor med rätt hög precision. Bildklassificering presterar sämre då den har problem med kontext och har svårt att koppla ihop rätt logga med rätt hemsida. De multimodala lösningarna visar överlag liknande resultat som textklassificeringen, men en viktbaserad beslutsmodell presterar bättre än någon modell med endast en modalitet. Kompletteringstester mellan modaliteterna visade också att multimodalitet hade väldig hög potential. Dessa två resultaten indikerar på att multimodalitet är ett bra angreppssätt i detta kontext.