

SITSEEEING: USING MACHINE LEARNING TO CLASSIFY SEGMENTED WEB PAGES

BY

Shekhar Dewan

A thesis submitted to the
Department of Mathematics and Computer Science
Mount Allison University
in partial fulfillment of the requirements
for the Bachelor of Science degree with Honours
April 22, 2020

Contents

Contents	2
Acknowledgments	3
Abstract	4
1. Overview	5
1.1 Context	5
1.2 Introduction	6
1.3 Describing the SiteSeer System	7
1.4 Tracing the Development of SiteSeer	9
1.5 Related Approaches	15
1.5.1 Segmentation	15
1.5.2 Classification	18
1.6 Extending the SiteSeer System	20
1.6.1 Deep Learning	21
1.6.2 Machine Learning	24
2. Classification	26
2.1 Overview	26
2.2 Understanding the Data	27
2.3 Deep Learning (CNN)	35
2.4 Classical Machine Learning	40
3. Contributions and Future Work	46
3.1 Comparing our Results	46
3.2 Conclusion	54
3.3 GitHub Repository	54
References	55
Main Text	55
Figures	58

Acknowledgments

I would, above all, like to acknowledge the network of possibility in which I am embedded, for which both individuals and our society at large are responsible. Stated differently, I would like to thank the collective efforts of humanity that enable people such as myself to pursue research peacefully.

As for individuals who have permitted me to occupy a desirable position in this network, I thank above all my parents, who derived meaning from sacrifice and the opportunities it created for my brother and myself. Those in my personal life have variously afforded or diverted my attention in relation to the thesis, both of which have enriched the work, and my life.

Importantly, I want to thank my advisor, Dr. Michael Cormier, whose shoulders I have stood on, and whose unending patience and insight have permitted me to travel much further than I would have on my own. Finally, I would like to thank the institutes of human fulfilment and knowledge, of which the university is one manifestation. May it be that more excellent scholarship is forthcoming as a result of these structures, and that it is lifted by the rising tide of global prosperity.

Abstract

The aim of this thesis is to extend the SiteSeer system—previously developed by Dr. Cormier—by incorporating machine learning into its classification component. The SiteSeer system segments web pages into regions, which are then assigned a label from a set of labels called the ontology. It is this second component, the assignment of a label based on the appearance of a region of the web page, that we used machine learning for in this project. Multiple machine learning architectures were attempted for this, including a Convolutional Neural Network (CNN), and a Random Forest Ensemble—two architectures we focus on. The Random Forest showed the best performance, while the CNN's performance fell approximately in the middle. In the best case, we improved accuracy by 40% over weighted random predictions (weighted according to the imbalance in class membership), from 35% to 75%. We also roughly tripled performance compared to the earlier system, going from 20% to 59% accuracy. These results constitute a significant advancement in the performance of the SiteSeer system, and demonstrate that machine learning is an effective technique for classifying the present dataset, despite numerous challenges. The findings could be extended by combining the machine learning with the Hidden Markov Tree from earlier iterations of the system, by collecting more data, and by improving the initial segmentation algorithm that generated the dataset.

1. Overview

1.1 Context

The area of computer vision attempts to develop techniques that enable computers to interpret and usefully manipulate information that humans consider visual. This can be distinguished from image processing: while a low pass filter may reduce noise in an image, computer vision concerns itself with what an image represents, i.e., its meaning, especially in relation to the world.

While we want to 'teach' computers tasks humans can do, a computer's internal representation of information is different from ours. So, the extent of a computer's understanding, even with a 'general' learning algorithm, falls short of a human's, because the computer's internal representation is much narrower in scope. Consider an image classifier which learns to distinguish 2D images with the help of labeled examples. The classifier's scope of knowledge is limited to the image it sees, without a sense of what the image represents in the world, or how it may look from a different perspective, or any notion of how things behave in the world. A human, on the other hand, distinguishes scenes based on multisensory, interactive input, and a complex set of earlier experiences. This is a major reason why computers are poor at generalizing—their internal representations aren't as rich, and aren't informed by an understanding of the world.

But there is progress: developments in computer vision, especially in machine learning, have enabled us to build systems that are more general, better performing, and more robust than previous systems, by enabling them to learn from examples rather than requiring manually codified systems of rules (hand-written programs). Thus, current work in the field sits between the two extremes of hand-coded programs and of general

artificial intelligence approaching something resembling human generality of reasoning. It often involves replacing earlier logic-based systems with more recent probabilistic and statistical techniques. Not only have machine learning techniques allowed us to find unprecedented applications in computer vision, but they also harbour the possibility of far more general systems down the road: systems that learn from our increasingly vast quantities of data, and that possess several mechanistic advantages over us, being reliable, replicable, and often cheap.

Although the prospect of solving computer vision through a general machine intelligence system is not yet in sight, techniques in the field of computer vision can solve important practical problems, and the focus of our project is one such problem, in the context of understanding web pages.

1.2 Introduction

Our problem focuses on interpreting the content of a web page by exploiting its visual structure. Since web pages are man-made, interpreting them differs from interpreting natural images. Web pages have a relational and hierarchical structure (often represented by a tree), and their structure is intentionally created to convey semantic information to users. Therefore, while a web page is similar to an image in the sense that it contains visual information represented at the level of pixels, interpreting the semantic content of a web page also requires understanding its various sections, such as the header or advertisements. The sections are artificially created, and often natively digital, unlike images which may depict natural scenes, contain artifacts, and represent analogue scenes digitally. Thus, web pages differ sufficiently from natural images to merit further investigation, and potentially, a different set of techniques.

Since classification in computer vision has mostly focused on natural images [3], web pages provide an opportunity to extend the scope of computer vision techniques to man-made images. Another advantage of analyzing web pages is that ample data is

available, and the problem is immediately practical, as parsing web page structure has applications for systems such as screen readers, which we discuss next.

Assistive screen readers are used by visually impaired users [29]. While these readers are effective at parsing content with a simple structure, such as a single-column text document without annotations or appendices, they struggle with the complex interfaces that are a part of daily computer use. One of the major areas of struggle is the content of web pages on the internet, which have a variety of complexities in their structure [14]. In particular, news web pages are popular amongst users, and have structures which are complex, varied, and dynamic.

Earlier, Cormier created a system (which we are coining SiteSeer) which can understand the structure of complex web pages, such as news web pages, by segmenting them into semantically significant sections, and then labeling the sections (we discuss the details later) [3]. Such a system, once fully developed, would allow screen readers and web scrapers to choose which sections they would like to draw on, and in what order. Thus far, SiteSeer has undergone several revisions, and is able to segment as well as classify web pages. Our aim here is to improve its performance and its generalizability by incorporating machine learning into its classification component, which has thus far been lacking [3]. We describe the current state of the SiteSeer system next, then review other approaches in the literature, and finally describe the extensions we are proposing.

1.3 Describing the SiteSeer System

The SiteSeer system, described in Cormier 2018 [3], uses a purely vision-based and principled Bayesian approach to segmenting and classifying web pages [3]. SiteSeer segments and then classifies the segments of the rendered image of a web page, based on the eMine ontology (introduced later). This purely vision-based approach presents several advantages, such as independence from the underlying implementation of a web page, and potential generalizability to other man-made images. The extendability

of the SiteSeer system, along with its theoretical properties and practical performance, makes it an attractive system to refine, in the hopes that we can obtain better web page classification performance. We now outline this system, along with its development across time.

Currently, the SiteSeer system uses probabilistic reasoning in a Bayesian framework to produce a hierarchical, tree-structured segmentation of a web page. Initial segmentations are created by proceeding from the root node representing the entire page, and then detecting locally significant edges to form smaller regions. This process is then repeated, until no more regions can be segmented further (thus, it is a top-down process with bottom-up edge detection) [3]. A Hidden Markov Tree (or HMT—we introduce this more fully later) is then created based on this structure (see figure 1.1), where each node is associated with an observation (image) corresponding to the region of the web page that was segmented. Since child nodes are further segmentations of parents (and leaves represent regions which cannot be segmented further), parent observations contain child observations, but not vice versa. Features are then extracted from these observations in the form of feature vectors, and these are used to associate possible labels with their probabilities. Simultaneously, the probability of labels based on the relationships between regions is also assigned (it is unlikely that the main content of a news page is nestled in the sidebar), and these are jointly considered to yield a final classification based on the most likely label for each of the nodes of the tree, given the observations and structure of the tree [3].

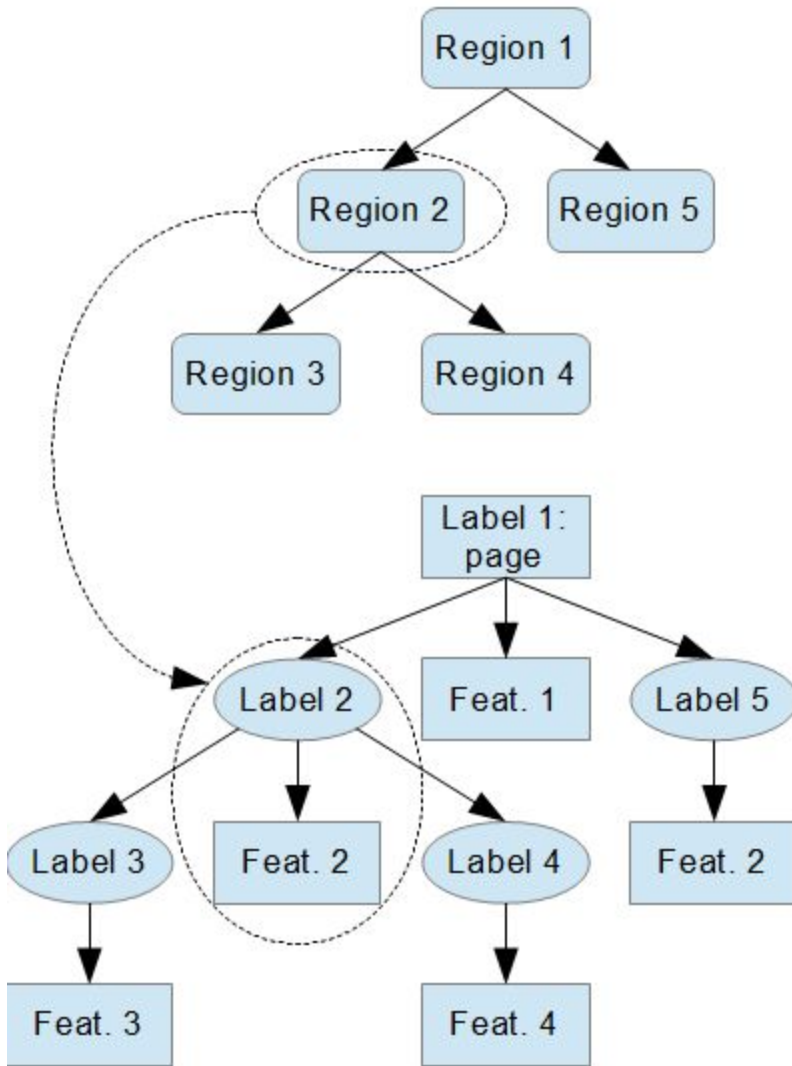


Figure 1.1 (from [3]): The generation of the HMT from the segmentation tree. The relationships between the regions of the segmentation tree are preserved, and labels and features corresponding to the regions are added, resulting in an HMT.

1.4 Tracing the Development of SiteSeer

The SiteSeer system was first proposed in [14]. It produces a hierarchical segmentation tree of a web page based solely on its visual appearance. The key tools for the segmentation are edge detection at the low level using certain prior assumptions combined with a top-down approach proceeding from the entire web page to generate

smaller regions, until no more regions can be further segmented. Some of the assumptions for identifying regions are that they must be separated by edges, and must be closed, rectangular, and axis-aligned [14]. This approach performs well on pages satisfying these assumptions (figure 1.2), but can struggle with pages with less orthodox structures which violate these assumptions (figure 1.3) [14].

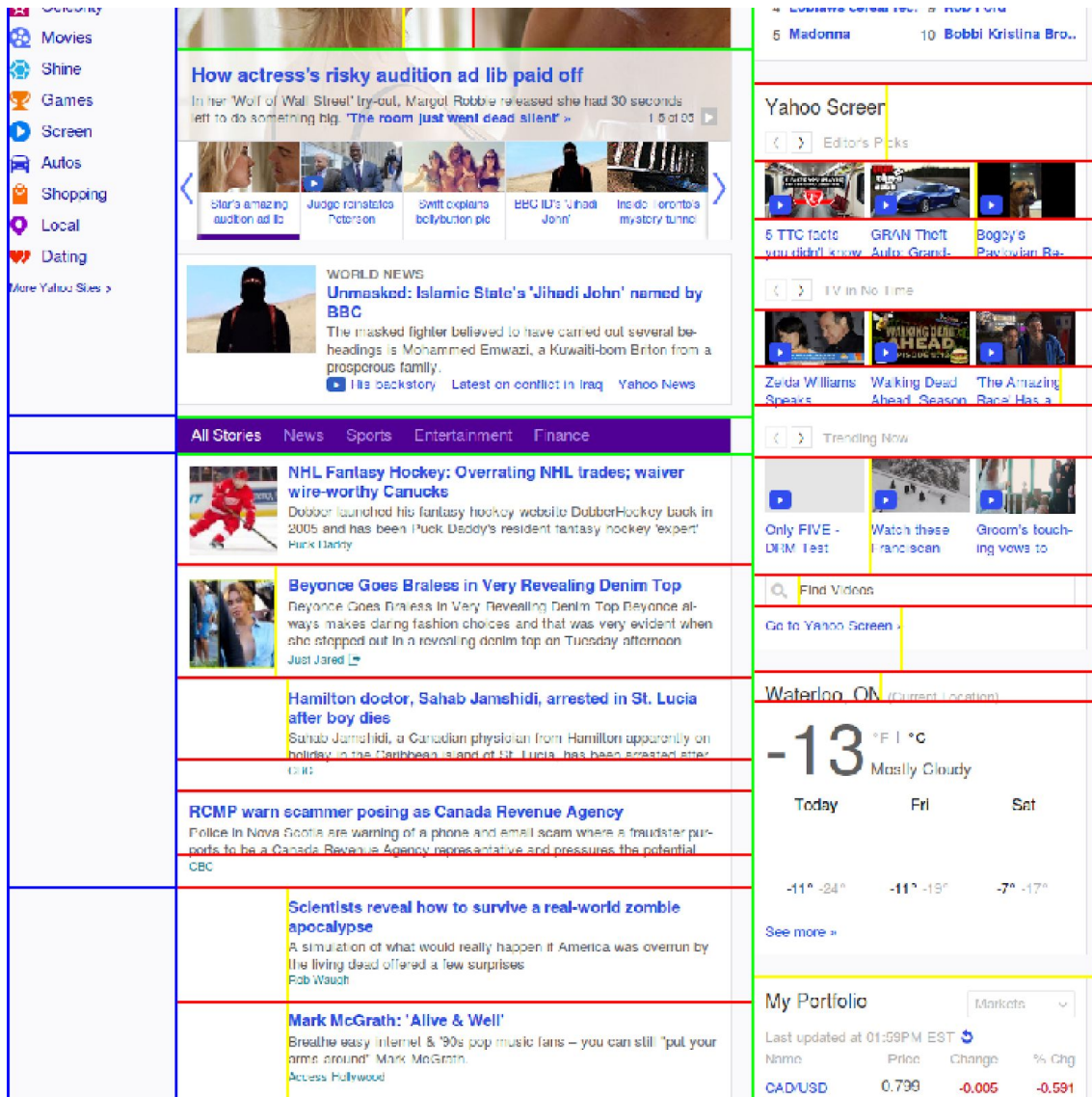


Figure 1.2 (from [14]): A relatively successful web page segmentation.

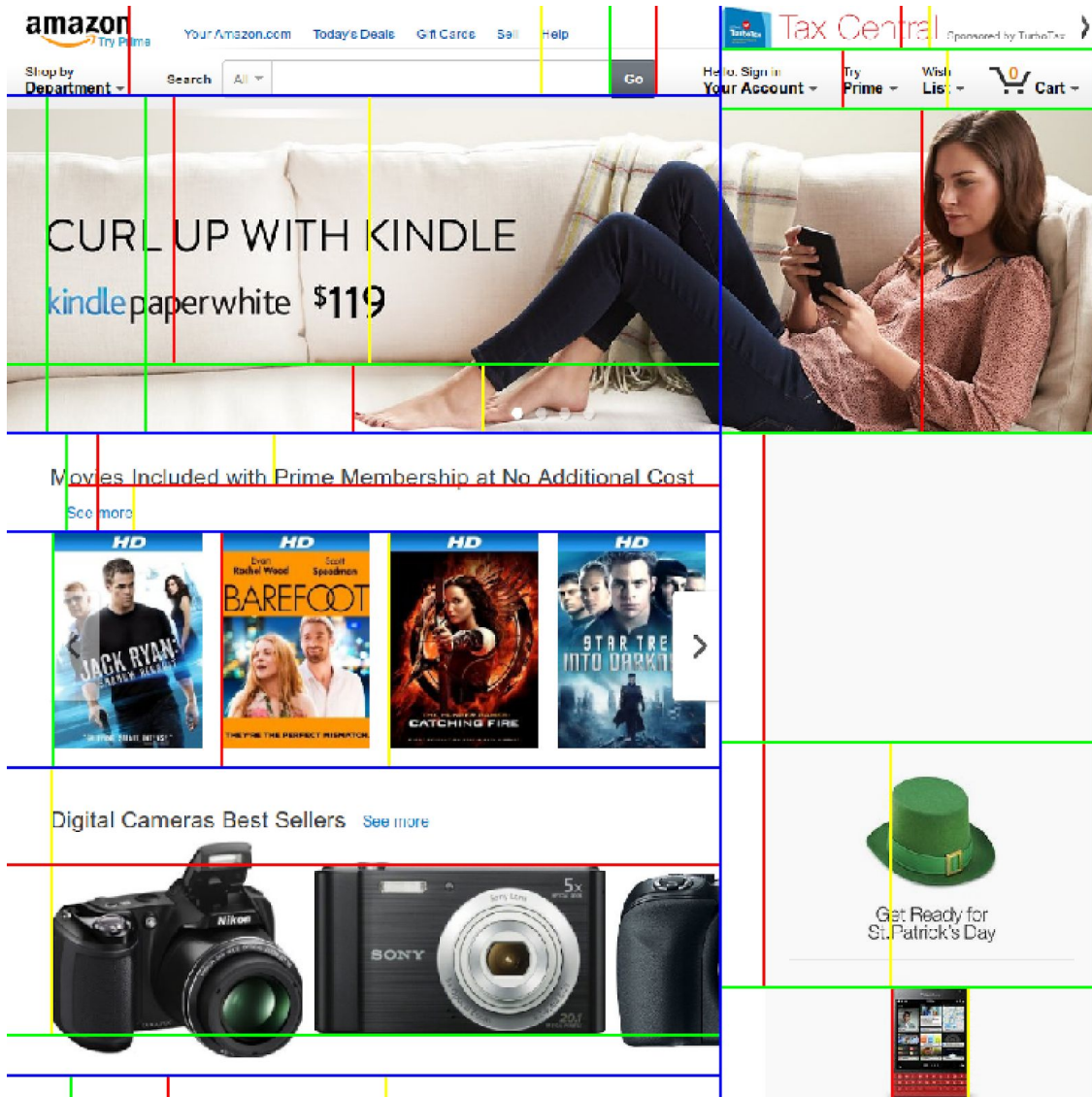


Figure 1.3 (from [14]): A less successful web page segmentation.

The next development in the SiteSeer system was the addition of a means of classifying the segmented regions by assigning labels to them [15]. The produced segmentation tree is used to generate an HMT, which combines the hierarchical segmentation with images of the segmented regions (observations) and the probability of various classes of labels being assigned to each node of the HMT. These labels are then assigned on the basis of the maximum joint a posteriori probability with respect to the relationship

between the nodes associated with the labels and the observations associated with each node (see figure 1.4). This is accomplished with a message-passing algorithm called Belief Propagation [20]. The labels used are ARIA labels (described below). An HMT is used because we can extend the structure of the segmentation tree by incorporating labels and observations associated with each node. The tree structure permits us to use exact instead of approximate inference for message passing, which is preferable and comes with certain guarantees [3].

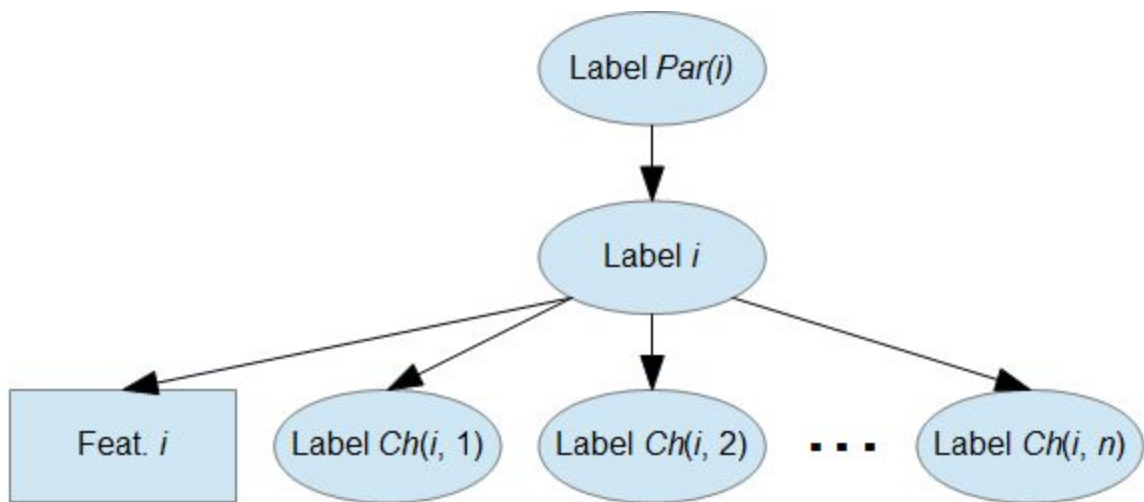
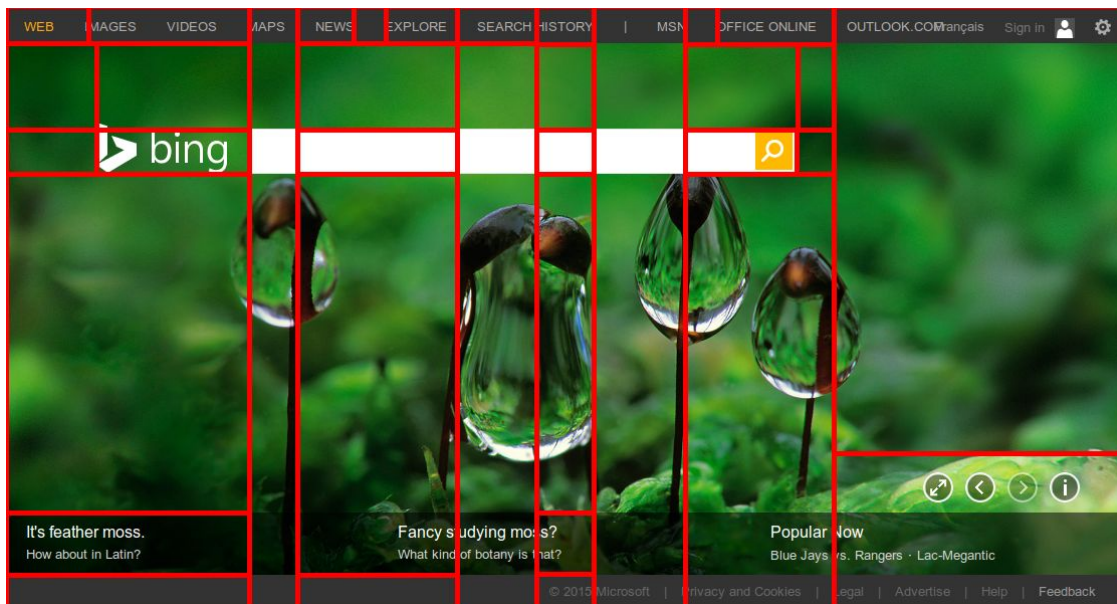


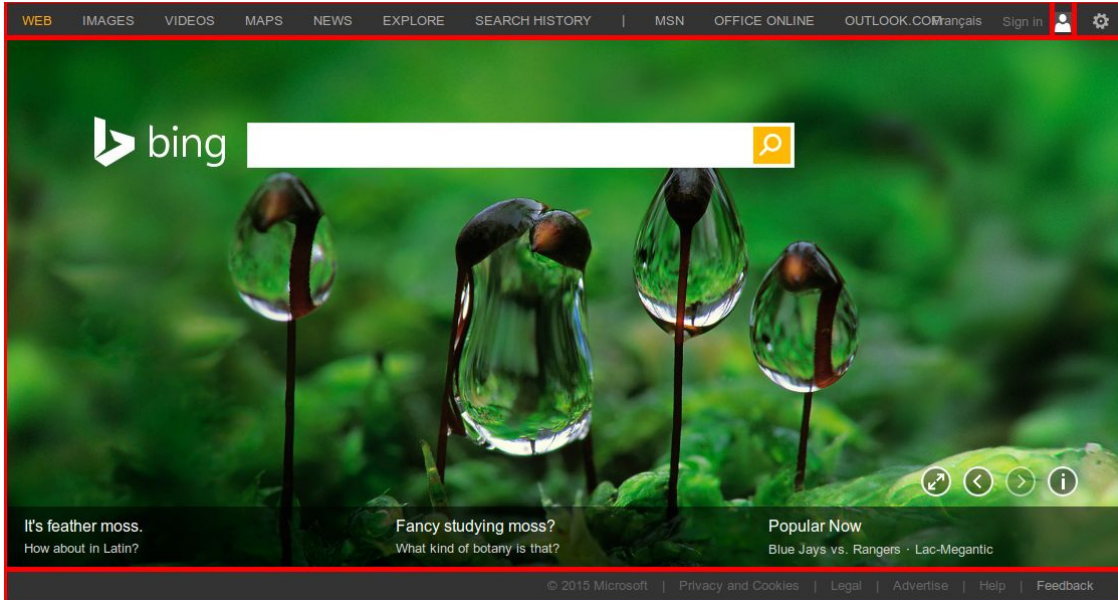
Figure 1.4 (from [3]): The above figure shows the statistical relations between the edges in the HMT. For the final assigned labels, we use Belief Propagation in both directions (from parent to child and child to parent) to obtain the maximum a posteriori distribution of labels.

One way to enable screen readers to parse a web page is to use a set of labels (called an ontology) to identify various regions of a web page. One of the more common sets of these labels is ARIA labels [1]. While they present one of the more common approaches to labeling web pages (which can be used for classification tasks), ARIA labels require developers to manually apply them. This means only a relatively small portion of web pages use ARIA labels, and of those that do, many have unlabeled regions. If we also consider that web pages and their styles can be revised relatively frequently, we can see

the drawbacks of the additional labour required to update the labels, and the requirement for these labels to be updated in accordance with elements shown on a web page. Thus, developer-implemented ARIA labels are an imperfect, but common, solution to the problem of parsing web pages for screen readers [3].

In 2017, Cormier et al. [16] made improvements to the segmentation portion of the SiteSeer system. The assumptions about edge placements and assumptions relating to the probability of a locally significant line were refined, resulting in improved performance when gauging the probability of a line between points being significant, better segmentation of forms, and better discrimination of a whole natural image versus regions of a web page that ought to be segmented. Although much improved (figures 1.5, 1.6), the segmentation algorithm still struggles with certain pages where structural assumptions are violated (see figure 1.7). This will ultimately also impact classification performance.





Figures 1.5, 1.6, 1.7 (from [16]): Figure 1.5 shows a problematic segmentation by the earlier version of the system. Figure 1.6 shows an improved segmentation by the new version. Figure 1.7 depicts a web page excerpt the current system still struggles with because it violates certain structural assumptions.

In 2018, Cormier et al. [17] developed a pipeline to evaluate the SiteSeer system. It allowed for the comparison of the segmentation algorithm to real segmentations produced by humans (using a drawing interface), and also allowed for evaluation of the classification labels when compared with the ground truth labels of the same regions. Thus, the pipeline enabled not only end-to-end performance evaluation, but also the evaluation of certain intermediary steps, allowing for an effective means to gauge the efficacy of future improvements to the system.

Additionally, Cormier et al. [17] switched to the eMine ontology [2]. One of the major differences between the eMine and ARIA labels is the level of granularity: by analogy, eMine labels are more concerned with identifying a car, whereas ARIA labels operate at the level of each wheel, or the windshield. Thus eMine labels are better suited for the segmentation approach of SiteSeer, which proceeds in a top-down fashion, and will identify the 'car' before the parts that compose it [2].

Cormier et al. [17] demonstrated that the SiteSeer system had strong segmentation performance, but the classification accuracy left significant room for improvement. Therefore, we are focusing on improving the classification portion of the SiteSeer system, and we will describe our proposed approach after briefly surveying some existing literature in this area.

1.5 Related Approaches

Before discussing extensions to the SiteSeer system, we survey approaches to web page segmentation and classification in the literature, and contrast them with our approach.

1.5.1 Segmentation

One strategy for segmenting web pages is to make use of the Document Object Model (DOM) tree of the web page, which is a tree structure in which each node represents an object (often an HTML object) in the web page (described in [4]). While this does not

require sophisticated computer vision algorithms, the approach suffers from certain drawbacks. The DOM tree (see figure 1.8) is generated from the underlying implementation of the code, and pertains to the structure of the code implemented for the web page, rather than what is visually significant to the user. Especially for more complicated web pages, this might mean that the segmentation produced on the basis of the DOM tree is quite different from one that would be produced by a user looking at a web page. Because of its dependence on code, the approach is tied to the underlying implementation, and will not extend to other man-made images that do not have underlying code implementation. Finally, since DOM trees contain objects as nodes, objects such as images which cannot be further segmented by a purely DOM-based approach would require the inclusion of certain techniques from computer vision.

An example of a DOM-based segmentation algorithm is called the Vision-based Page Segmentation Algorithm (VIPS), proposed by Cai et al. in 2003 [5]. This was further developed by Akpinar and Yesilada in 2013 [6], who also extended the range of tags the model could handle. While the system is well regarded, it is difficult to compare to SiteSeer in terms of segmentation quality because differing approaches are used. However, as already described, the dependency on the DOM tree places certain limits on the algorithm. In addition, as web development frameworks evolve, a model like this needs to be manually updated in accordance with changes in the nature of the underlying implementation.

In contrast, a purely vision-based approach would be able to segment all elements of the web page, be completely independent of the underlying implementation, and better reflect the segmentation which would be produced by the user of a website.

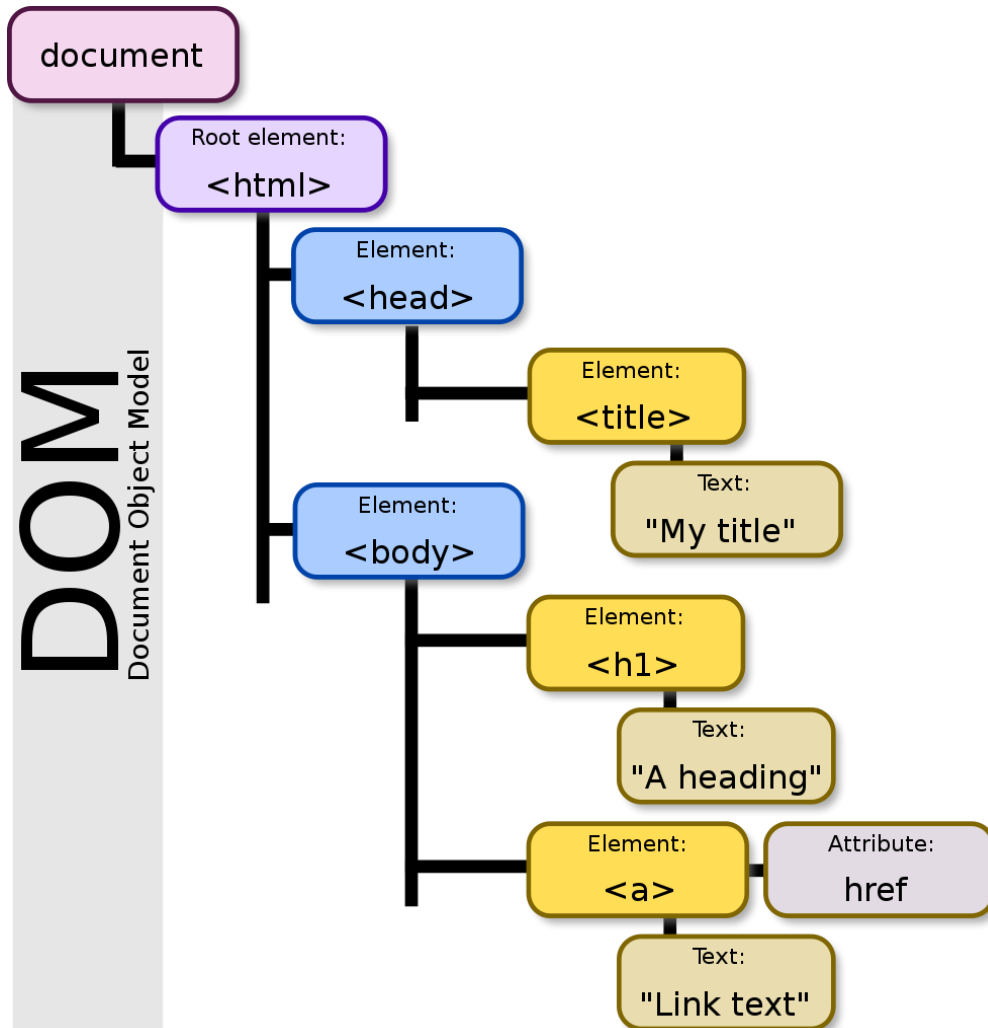


Figure 1.8 (from [A]): The DOM of a hypothetical web page, which is represented by a tree structure associating various elements of the page with the nodes containing the information in the corresponding element.

Barkol et al. [7], and Cao et al. [8] performed vision-based segmentation, but with a logic-based approach, rather than a probabilistic or machine learning approach. While such an approach is independent from the underlying implementation, the program logic has to be manually written, and is unlikely to cover the full range of websites, which show large variation.

Finally, neither of these two systems classify the segmented regions, unlike SiteSeer, which performs segmentation and classification in one common framework. We turn now to systems that classify web page regions.

1.5.2 Classification

Certain approaches have been proposed to improve web page classification performance. One is the labeling of sections of a web page from a set of labels in an ontology. For instance, ARIA labels are used by some websites, but they need to be manually added and updated. Even among the websites that use ARIA labels, many regions of their web pages are unlabeled. Other ontologies, such as eMine [2], would have similar drawbacks if manually implemented. However, if a means could be devised to automate the assignment of labels to regions, the usefulness of these ontologies would be vastly enhanced. Such an approach is possible with vision-based region classification. We now describe systems that attempt classification of web page regions.

In 2006, Chen, Zhong, and Cook [19] outlined an approach employing a Generalized Hidden Markov Model (GHMM) for classifying regions in a web page. A Hidden Markov Model (HMM) model is a system used to model Markov processes. The model is arranged sequentially, where nodes are associated with discrete time-steps, at which certain observable (hidden) states are paired with observations. The hidden states are typically assumed to influence the observations, and each state is assumed to be conditionally independent of other states given the previous state. Thus, we can infer the state of the modelled system as it evolves based on the changes in the observations, which permits us to use earlier time-steps to make more informed predictions about the current and future state of the system. A GHMM substitutes singular observations with a set of observations at each discrete time-step, but still maintains the assumption of one-dimensionality in the sequence of states. Thus, although most of the assumptions are preserved, more complex systems where hidden states are associated with a set of observations can be modelled by a GHMM. To adapt this to the 2D structure of a web

page, Chen et al. [19] used a depth-first traversal of the tree-structured segmentation to produce a sequence of states. Their goal was to obtain the sequence of most likely states (labels), given the observations (set of features). It should be noted that web pages are not sequence structures, and partly for this reason, the SiteSeer system has used a Hidden Markov Tree instead of an HMM for classification, which is derived from the segmentation tree. As described in the next section, we are proposing incorporating machine learning instead of an HMT, which we intend to reincorporate in the future.

Romberg et al., in 2000 [9] and 2001 [10], described two systems that use Hidden Markov Trees (HMTs) for multiscale image classification. A Hidden Markov Tree generalizes the GHMM to permit a tree structure, which enables better flexibility and more complex relationships. The lack of cycles permits certain assumptions and algorithms, such as exact inference in a Belief Propagation algorithm, which was described by Pearl in 1988 [20]. In Romberg et al. [9], complex wavelets are used with Hidden Markov Trees. This differs from SiteSeer in that the structure and size of the HMT in this system are fixed, and the authors are trying to solve a different problem (trying to represent components of a mixture model pertaining to wavelets rather than high-level classes of these regions). In 2001, Romberg et al. [10] proposed a system that classifies textures in a given image. Again, the HMT has a fixed structure, and textures are a class based on one kind of visual difference, rather than semantically significant components of a web page. Thus, use of an HMT for region classification in SiteSeer differed significantly from other work in the literature.

Semantic segmentation and instance segmentation are two kinds of outcomes with respect to simultaneous segmentation and classification. Of these, instance segmentation is more appropriate for web pages, as it produces segmentations identifying each instance of a class, in addition to identifying which pixels belong to which class [3]. Several approaches have been proposed with respect to instance segmentation (See Yang et al. [11], Li et al. [12], Arnab et al. [13]). Of these, the one

proposed by Yang et al. [11] is most relevant. In 2012, Yang et al. [11] developed an instance segmentation for 2.1D natural images. These images contain some information about depth by means of image features such as occlusion, but are still largely 2-dimensional. The algorithm developed by Yang et al. [11] uses a Bayesian approach with a tentative top-down segmentation which is refined by a bottom-up classification (both occur simultaneously). Yang et al. [11] concern themselves with natural images containing occlusion, which renders their problem space significantly different from ours. In addition, their system performs segmentation and classification separately, using object classes. So, instead of identifying instances of objects as in Yang et al. [11], our system is able to segment the entire web page, which we classify after the segmentation.

In surveying the literature, we have compared the assumptions and tradeoffs SiteSeer makes, along with their justification. One area that remains underexplored is machine learning, which we believe could provide a useful addition to the SiteSeer system. We turn now to our proposed attempt to incorporate machine learning into SiteSeer.

1.6 Extending the SiteSeer System

To recap, the SiteSeer system has several desirable properties. Using a principled Bayesian approach, and purely visual information (which aligns closely with the area of computer vision), it takes advantage of an HMT structure with a message passing algorithm to make use of context and maximize the likelihood of its classifications. In addition, a pipeline is set up for the evaluation of the system, which enables us to gauge its performance in comparison with humans. However, at the low level, it uses manually encoded priors for segmentation, and rudimentary feature extraction for classification.

Since the high-level aspects of the SiteSeer system are sophisticated as well as theoretically motivated, it stands to reason that the low-level aspects of the system involving hand-coded logic for interpretation and extraction from images be targeted for

improvement. Since the segmentation is quite performant [17], and since altering the segmentation will render the pipeline unable to evaluate the performance of the classification without generating new data, the classification seems to be the best component of the system to modify. Therefore, we seek in this project to improve the performance of the SiteSeer system by incorporating deep learning.

1.6.1 Deep Learning

Given the size of the dataset of individually labeled regions (a little under 8,000 images), deep learning might be feasible, and is worth investigating. With enough data, end-to-end deep learning is highly performant, often surpassing any other technique [23]. For computer vision, Convolutional Neural Networks (CNNs) are the prevailing architecture for deep learning, and have had breakthrough results on multiple computer vision problems. In addition to their performance, CNNs are somewhat understood and somewhat interpretable. Thus, for our deep learning approach, we propose incorporating a CNN-based architecture to classify our image, with the possibility of combining our predictions with the HMT down the road. For an overview of CNN architectures, see [26] and [27].

The CNN would be trained by providing it with images and labels. It would then be expected to output the probability of an input image being associated with each of the potential output classes. The CNN may need to be relatively small, be able to handle images of different dimensions, and need an output layer that provides the probability of each output class. Several means are available to achieve each of these, and our ultimate decision will depend on theoretical properties, implementation feasibility, and performance.

One challenge pertaining to our dataset is that the images are of differing resolutions, while most CNN architectures are only able to handle images of certain fixed

resolutions. While resizing is an option we explore, we potentially lose some valuable information about the dimensions of the image, which could help the classifier if retained. We decided to try CNN models that could handle images of varied sizes, as well as ones that required resizing, as resizing can save training time and allows us to use more standard approaches.

In terms of dealing with images of differing sizes, we considered using Spatial Pyramid Pooling (SPP), which offers a principled approach that works well in CNNs [18]. The approach involves adding an SPP layer before the final fully connected layers, with the layer having the ability to resize images of any input size to desired dimensions. However, for implementation reasons, this was eschewed in favor of Fully Convolutional Neural Networks (FCNNs), which we discuss next.

FCNNs differ from CNNs in that they lack fully connected layers right before the final layers to make the predictions. This change allows them to accept input images of any dimensions, since the fully connected layers have fixed size requirements. The tradeoff is that the FCNN has poorer generalization beyond the training data, but if the image dimension is vital information (which resizing would eliminate), we anticipate this is a tradeoff worth testing. See [21] and [22] for more information on FCNNs.

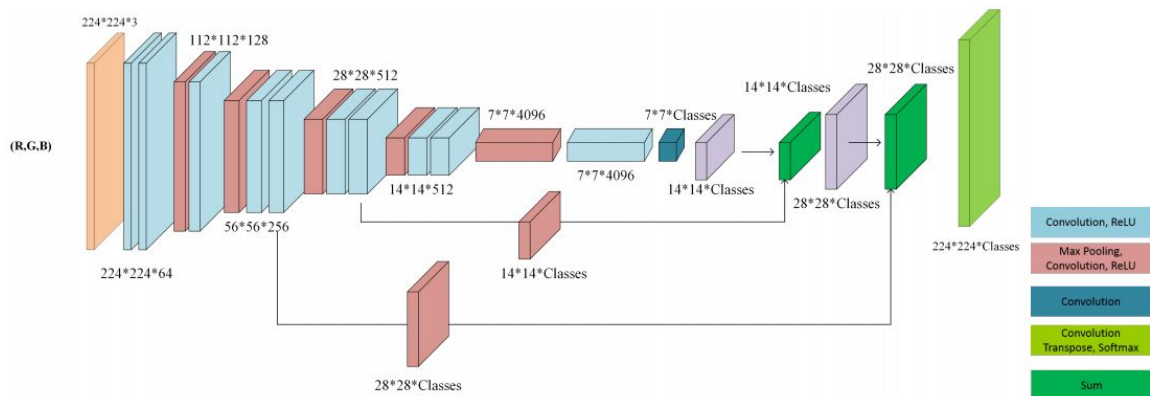


Figure 1.9 (from [21]): A Fully-Connected-Neural-Network (FCNN) architecture. The one implemented by us allowed for images of any size, and predicted any number of classes.

For dealing with a dataset of limited size, we anticipate constraining the size of our CNN and using regularization. Another possibility is data augmentation, which involves generating more data by varying images in the current dataset slightly, by transformations such as rotation or the addition of noise. While this is common with natural images, it is less useful in rendered web pages as there is little noise, and all the information conveyed is semantically significant, which altering the image may disrupt. For instance, rotating an image may change a horizontal box to a vertical one, and those often signify different things in web pages. Thus training a classifier to treat them both the same may actually hinder performance on new data. If the data still proves to be insufficient, we may generate additional labels, but this could also be a future extension to the system if we further refine and develop our evaluation and training pipeline.

We intend to use three CNN architectures—an FCNN, the architecture seen in [23], and a scaled down version of VGG-16 [25] developed by Dr. Cormier. If all three architectures perform similarly, we intend to focus on the architecture seen in [23], as it is a well-known, classic architecture. Otherwise, we will emphasize the best performing architecture.

1.6.2 Machine Learning

Originally, we had only intended to explore deep learning as part of this project. However, during the course of the work, I proposed an alternative approach for consideration, and based on its performance, it proved promising enough to investigate separately. This section has been added after we began running the experiments, to provide information about this approach.

While CNNs can handle images adeptly, classical machine learning techniques are not effective at parsing images. Instead, we attempted to generate some data about the image, such as its position relative to the page, the log of its area, aspect ratio, length of its sizes, etc. The full list can be seen in our GitHub repository for which a URL is provided at the end of Section 3. In addition, we tried using statistics generated about the image, but as this did not aid performance, the approach was shelved for future investigation.

For machine learning, our approach was empirical; we tested a range of classifiers from the Scikit-learn library [24], and upon finding that the performance of the Random Forest Ensemble Classifier (hence called Random Forest) was notably better, we emphasized that approach. Random Forests are based on bootstrapping, which involves taking samples (typically with replacement) of the data, and fitting on those subsamples (this also permits simple fits on those subsamples). These fitted models are then aggregated together, in a procedure called bootstrap aggregation (bagging). For a classification problem, this aggregation could involve a voting procedure, where the classification with the most votes is selected. The Random Forest involves using decision trees to fit each of the samples, which in aggregate perform the classification (see [28] for more information).

In theory, the machine learning approach and the deep learning could be combined, and the predictions could be jointly considered with those of the HMT to provide better results down the road. These possibilities are discussed further in Section 3.

For the scope of this project, we use existing data from news web pages for training and evaluating our system, with the intention of building a system which will eventually be generalized to most web pages, and other man-made images.

2. Classification

2.1 Overview

Initially, we had proposed using deep learning with a CNN as our primary approach. However, as we examined the dataset further, and as implementation details became clearer, certain limitations were perceived. Some of these challenges are described in the corresponding sections, to inform future researchers who want to use our dataset or methods.

These challenges led us to an alternative approach which did not use deep learning, and relied on data about the image rather than the raw pixels themselves. As our results indicate, this choice provided better accuracy and greater consistency. Accordingly, it will be preferred in future iterations of the SiteSeer system, although both approaches will be explored.

What became clear is that apart from the difficult aspects of the classification problem described below, we were constrained by the size of our dataset. This is good news, as it implies that with more data, we may achieve even better performance, and may start to attain results with practical feasibility.

We turn now to discuss our dataset, following which we will discuss deep learning with the convolutional neural network, followed by the classical machine learning approach used here.

2.2 Understanding the Data

7,869 images were used as the dataset, collected in earlier versions of the SiteSeer system [17]. The images spanned 27 classes, of which 19 had more than ten members (totalling 7,840 images), with six having more than 250 members (totalling 7,076 images). Due to the high class imbalance (see figure 2.2 and 2.3), we eliminated classes with fewer than ten members, and for certain procedures, we only selected classes with at least 250 members; this still retained 90% of our sample. Another significant challenge was the variation in image size, both across (see figure 2.4) and within classes (see figure 2.5).

It is worth discussing our ontology here, which is shown in figure 2.1.

```
1: "Page",
2: "Article",
3: "Body",
4: "BreadcrumbTrail",
5: "Caption",
6: "ComplementaryContent",
7: "Container",
8: "Copyright",
9: "Figure",
10: "Footer",
11: "Header",
12: "Icon",
13: "Label",
14: "Link",
15: "List",
16: "Logo",
17: "Margin",
18: "Menu",
19: "MenuItem",
```

```
20: "Nothing",
21: "Separator",
22: "Sidebar",
23: "SpecialGraphic",
24: "Title",
25: "TitleBanner",
26: "empty",
27: "",
```

Figure 2.1: Label ontology.

It is important to notice that these labels do not relate to the appearance of the image, but instead to its function on the page. For instance, the logo across companies can look very different, and a box containing a news report can have varying dimensions, fonts, and images across sites, depending on the popularity of the story. This results in dissimilar looking images being classified into the same category. We can also see that sometimes images from two different classes can look quite similar, as shown in figure 2.7. To tackle this, SiteSeer previously used a Hidden Markov Tree, which uses data about the position of an image relative to its parents and siblings, to help disambiguate similar looking but semantically distinct images. We intend to incorporate this tree back into our system to adjudicate between images when our machine learning is uncertain, but for the scope of our project, only the machine learning was included, and thus we anticipated separating similar looking images as being a challenge.

The images below shed light on various aspects of our data, many of which pose challenges to our project. Figure 2.2 shows class balance across all classes on a log scale, figure 2.3 shows class imbalance in classes with at least 250 members, figure 2.4 shows some statistics about image dimensions across all classes, figure 2.5 shows image dimension statistics within a class, figure 2.6 shows variance of images within an example class, and figure 2.7 shows similarities in images placed into separate classes.

Collectively, the images demonstrate the challenges encountered in the classification problem, and suggest that some images may be tricky to classify. Considering the range of variance within any given class, it becomes clear that for smaller classes, many more examples would be required for a pattern to emerge.

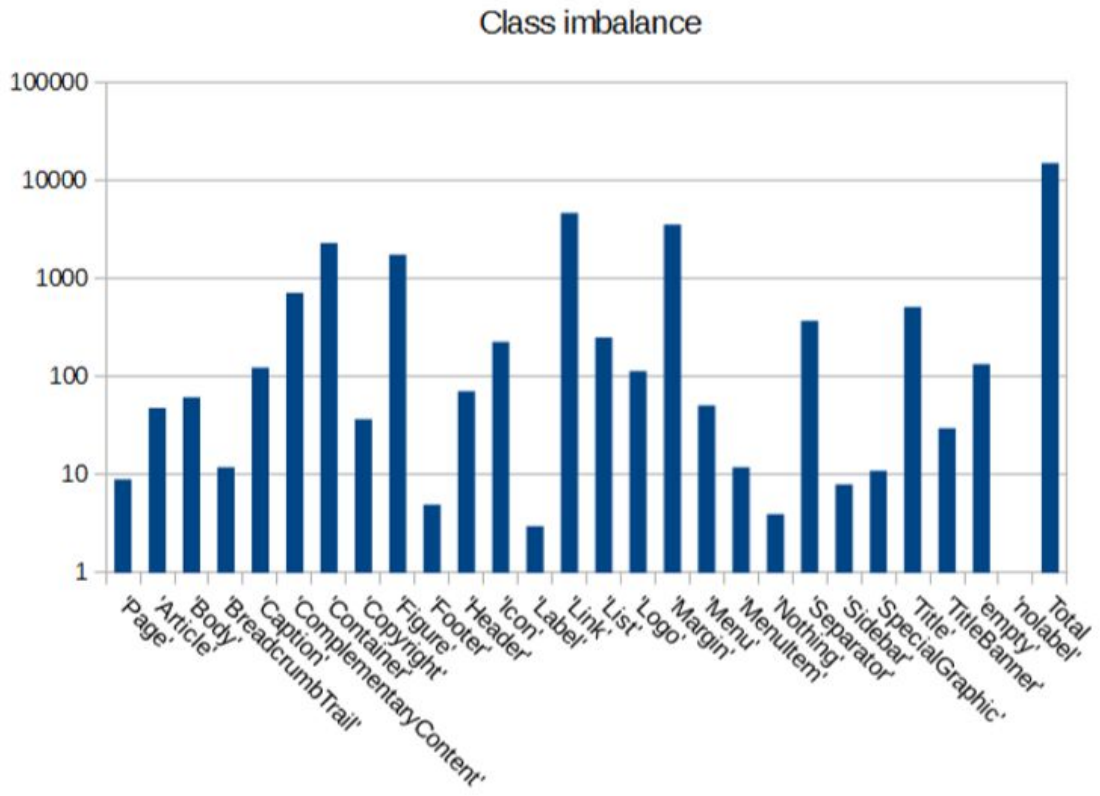


Figure 2.2: Class imbalance among all classes on a log scale. Note that this implies a very high degree of class imbalance. From [3]

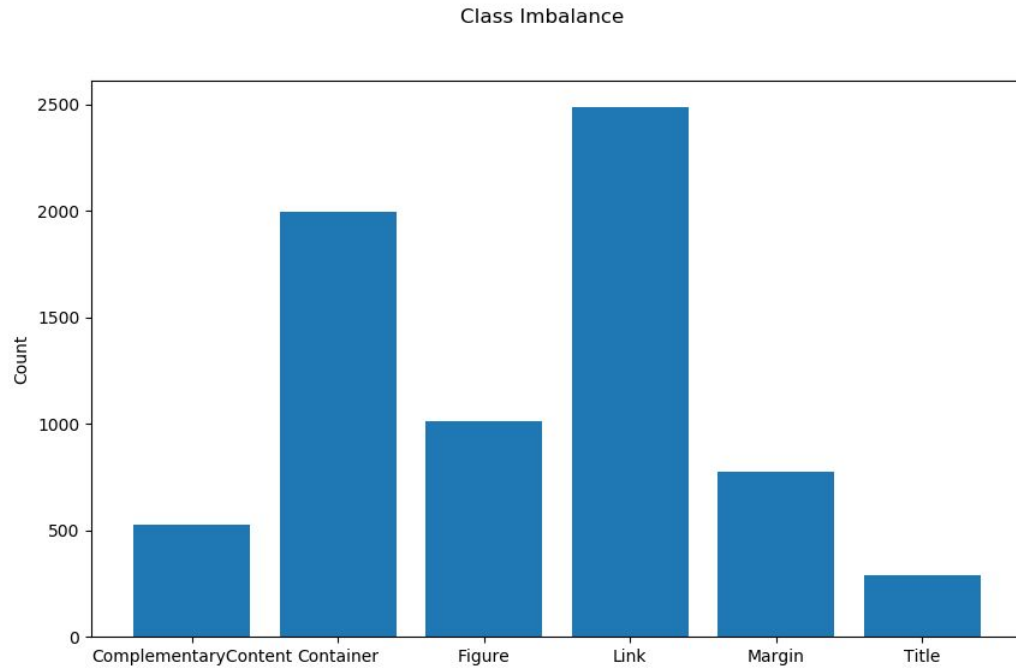


Figure 2.3: Class imbalance amongst 5k pruned classes with more than 250 members. Although less imbalanced than when we consider all classes, this is still quite imbalanced.

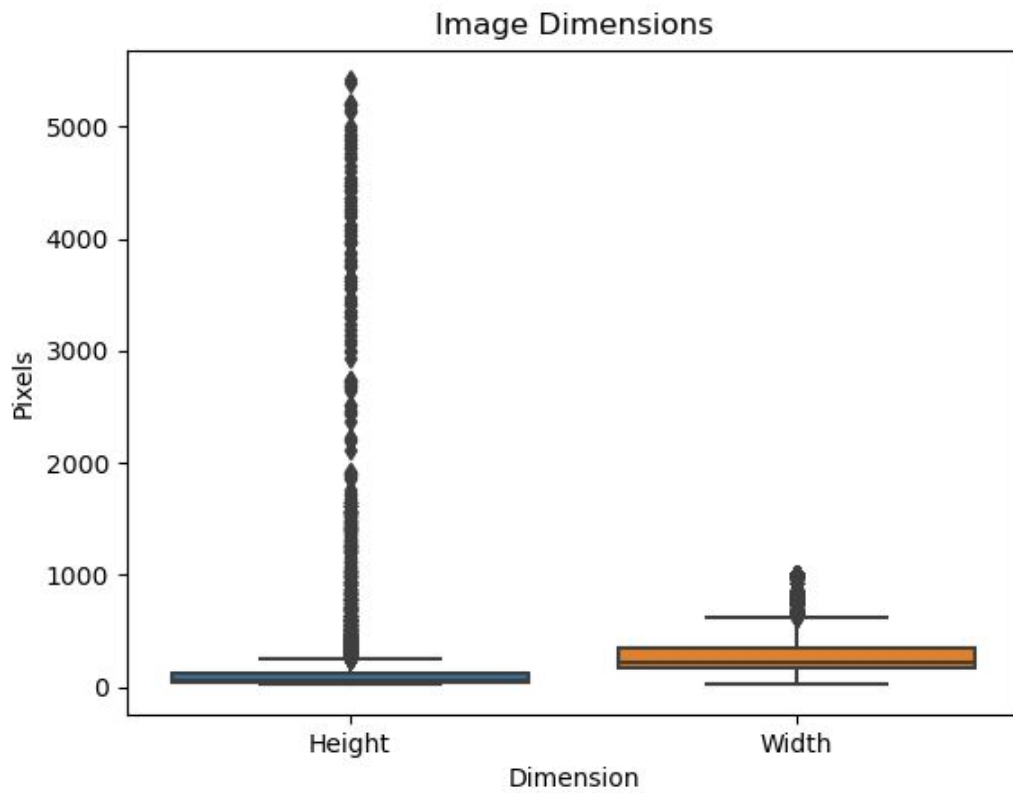


Figure 2.4: Image size differences across the dataset. Clearly, there is large variation in the range of image sizes present.

Images can also be quite varied within a class, as seen in figure 2.5.

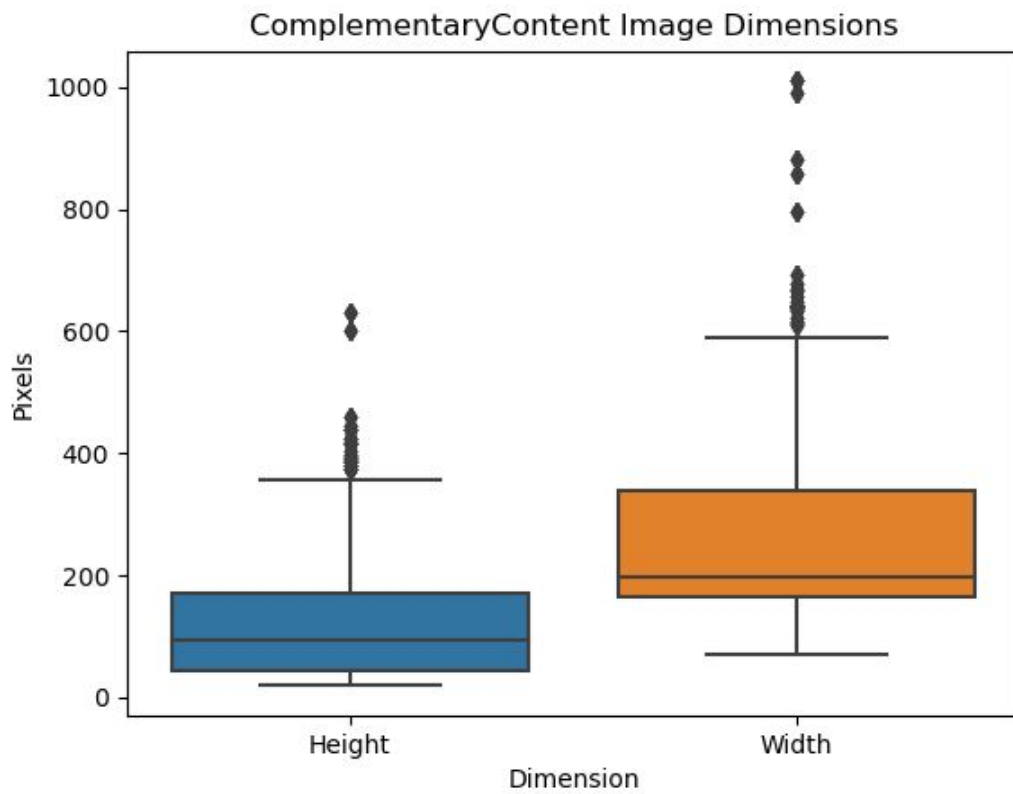


Figure 2.5: Image size variation within Complementary Content Class. Since image dimensions vary significantly within classes, we anticipate separating them to be challenging. This is further examined in figure 2.6.

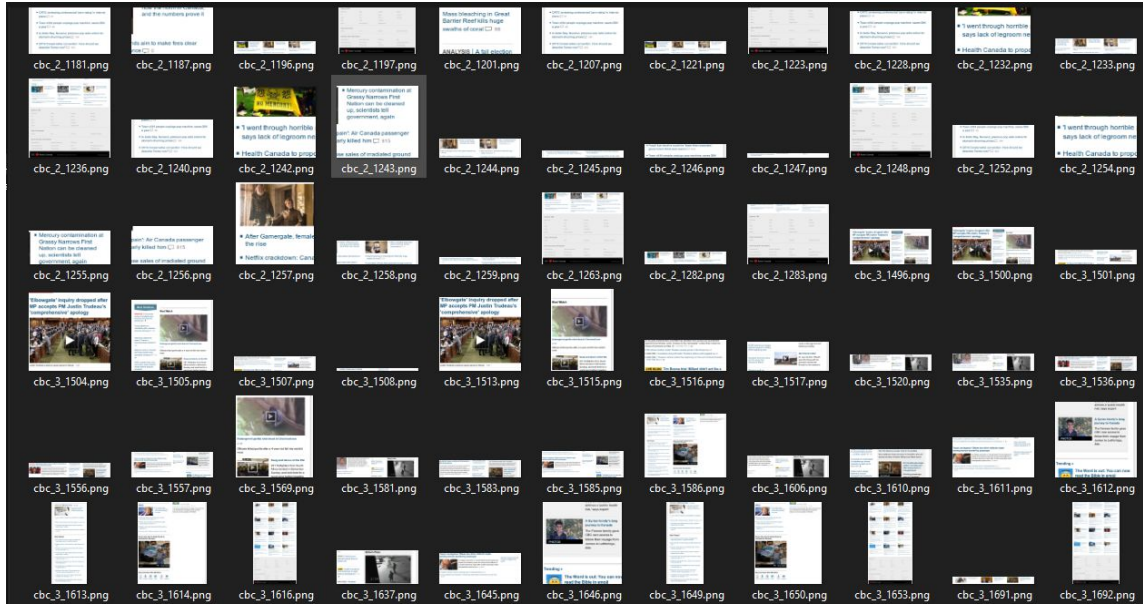


Figure 2.6: Some images within the container class. As can be seen, the images look quite different, and have widely varying dimensions. This suggests that prima facie, we should anticipate many cases where it is difficult to decide which class a particular image belongs to.

Figure 2.7: The left image above is from the "Page" class, while the right image belongs to the "Body" class. Notice the similarity between the two images, which only differ by a slim section at the top.

The data available, as portrayed in the figures above, is quite different from data we hope for in machine learning, where we want classes to be balanced, the images to be of the same resolution and granularity, and the categories to be easily distinguishable. And so, prior to running the experiments, we anticipated limits on performance.

2.3 Deep Learning (CNN)

We employed three CNN architectures for deep learning—an FCNN, a CNN resembling a scaled-down version of [23], and the minicnn2 architecture adapted by Dr. Cormier from VGG-16 [25]. Their structural details can be found on the associated GitHub page (see Section 3); the important notion here is that the FCNN, unlike the others, was able to train on images of differing sizes, while the others required image resizing and padding.

For training, we initially set the minimum class member threshold to ten, but we increased this to 250 (leaving us with 6 classes) to reduce class imbalance and to better diagnose performance. This threshold was retained since, as we later explain, we experienced variability in training performance. We obtained similar results from all three architectures, namely about 35% accuracy without rebalancing the classes. It turned out that they were all picking the most probable class. After rebalancing the classes, we obtained about 55% performance. Since the performance of the FCNN did not differ significantly when it was trained on the multi-resolution images as opposed to the resized and padded ones, it was generally trained and tested on the padded, uniform resolution images, as the training time with images of varied sizes was very long. We speculate that similar performance between the two cases occurred because our padding approach preserved information about aspect ratio and orientation by resizing the image into an even multiple or factor of 224 on the larger side, and scaling

the smaller side by the same multiplier. We used zero padding (which means that all three channels on the RGB spectrum were set to zero, resulting in black), and since all our web pages were depicted in white, the web page and padding were easily distinguishable.

This also suggested that the neural network was not significantly benefiting from the text on the image (as the resizing was lossy and eliminated significant visual information), but was perhaps relying more on the visual cues and aspect ratio. This could be offset by more data; in the meantime, it is possible that performance will improve if we choose a larger standard image size, and then position the resized images such that their position corresponds to their relative position on the page. This could further help disambiguate classes, and would be worth investigating if we want to explore the deep learning approach further.

Pertaining to the results of the CNN, it is vital that we distinguish the average performance (about 55%) from the performance spread (18-72%). While on average our performance is acceptable, the variability is concerning, and makes it difficult to gauge whether the model will generalize well, and if so, which iteration of it will generalize well. The spread can be a result of changes in the distribution of our data when the images were copied into different datasets (the data was streamed due to memory limitations, and therefore needed to be preallocated to training, validation, and test sets, and was not re-randomized per run), or from the initialization of the weights of the CNN. With more computational power, it will be possible to disambiguate these, because the data can be reallocated multiple times with fixed training initialization weights, and the model can be retrained with differing initializations with particular data allocations. Due to limitations in computational power, we can only state that there was significant variation in both cases, and that we anticipate this will bear out under further investigation. Information about variability of the performance of the CNN architectures can be seen in the figures below of training graphs and confusion matrices.

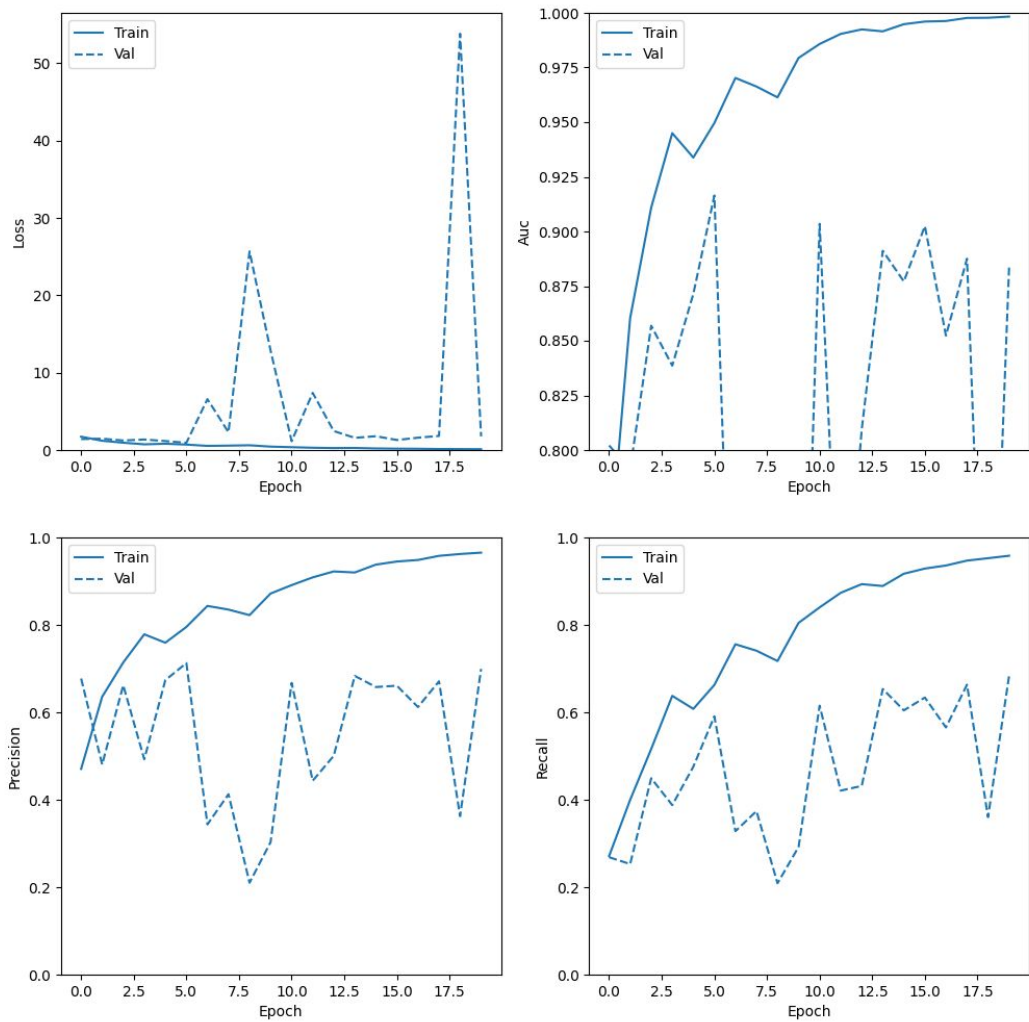
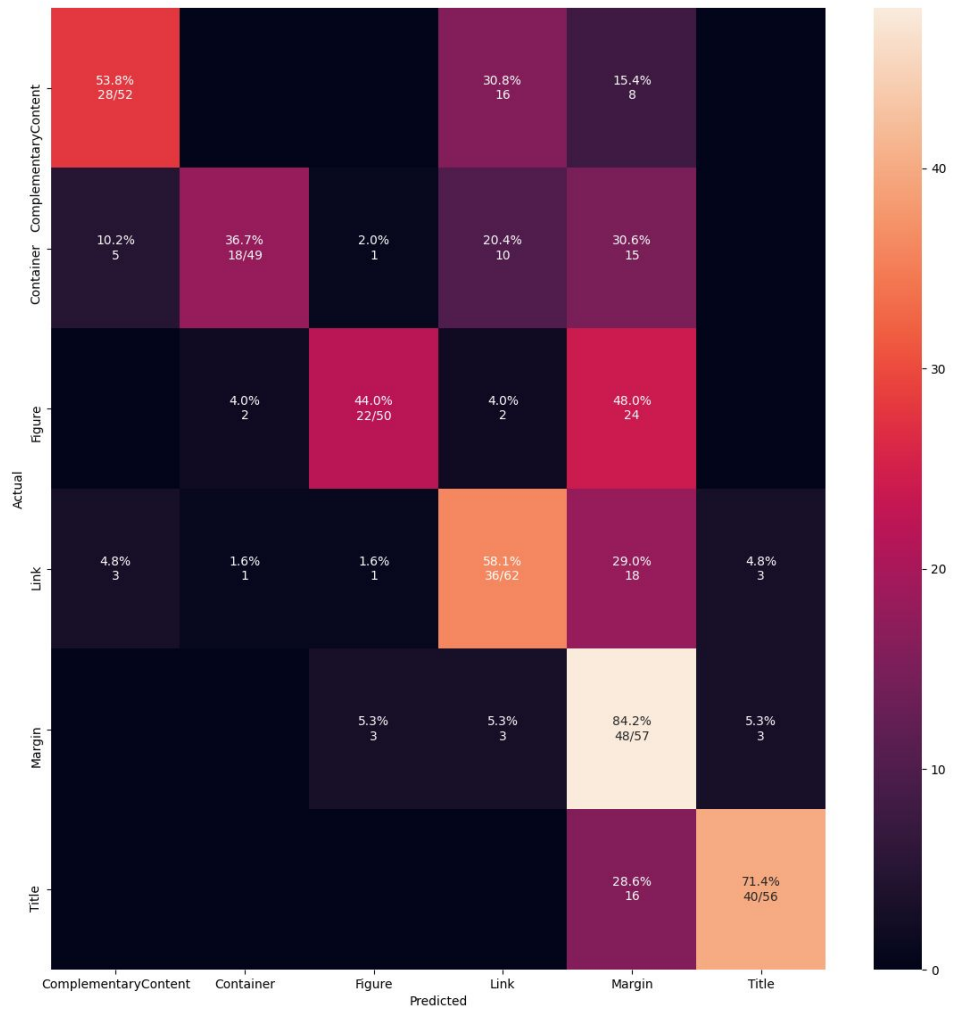


Figure 2.8: Training graphs from one instance of the CNN. On the training set, the curves are relatively smooth, but on the validation set, the performance varies significantly, especially the loss (which is the metric we are monitoring for the training). This indicates that there is instability of performance in the training process, and depending on which iteration of the model is picked (the one with the lowest loss), we can anticipate variance in the test results. A larger dataset would stabilize the training, as we would more reliably converge on a global optimum.



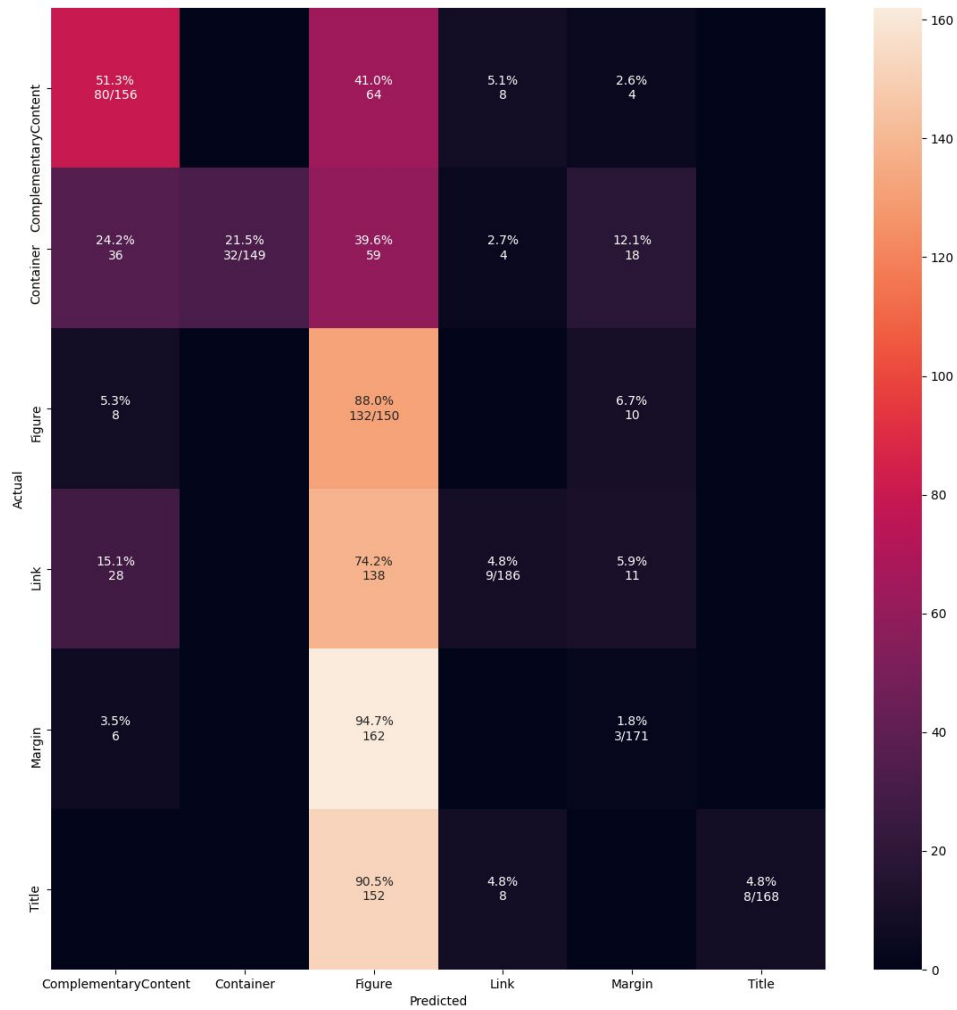


Figure 2.9 and 2.10: Confusion matrices for two instances of the CNN model trained on a 90/7.5/2.5 and on a 70/15/5 train/val/test split, respectively, with a random reallocation of images between the two runs. The confusion matrices demonstrate that the errors in the CNN can be starkly dissimilar. Significant differences also occur when we use the same data allocation and split size, but simply reinitialize the starting weights of the CNN.

To recap, we have demonstrated that the CNN approach was, as anticipated, challenging, and that the challenges occurred at all levels, from the dataset size, image size, and class separation, to documentable variation in results due to data reallocation and weight reinitialization. Therefore, while at around 55% the CNN approach showed decent results (given the constraints), it is difficult to assess exactly the results and their generalizability. To address these concerns, and to better understand our results with deep learning, we proposed a different approach, using classical machine learning. We turn to this next.

2.4 Classical Machine Learning

To implement classical machine learning, we decided to use data about the image, such as its position on the page, area, log aspect ratio, and length of the sides, to ascertain class membership, rather than using the image itself. This allowed us to employ many simpler machine learning models. We tried out-of-the-box solutions from the Scikit-learn library [24], particularly a Random Forest Ensemble Classifier, ExtraTrees Classifier, Adaboost Classifier, Multi-Layered-Perceptron (MLP) Classifier, Gaussian Naive Bayes (GNB), and Support Vector Classifiers (SVCs). The accuracy of these models in the minimum 250 instance members per class scenario is shown in figure 3.1, and we use those numbers for our discussion here, since they match the training scenario of the CNN. The range of models was chosen to cover a diverse hypothesis space, and to provide insight into the nature of the problem. For instance, linear SVCs did not perform well (35% accuracy), indicating that the problem is not linearly separable, even when projected on an n -dimensional hyperspace (see [30]). The MLP classifier also had moderate performance with some variability, indicating that the problem was not well modelled by a neural network when working with the data provided. The Adaboost classifier performed moderately better than the CNNs (at about 60%), while the Random Forest classifiers and the ExtraTrees classifier performed much better, at around 73%. This suggests that ensembles, particularly sets of decision trees, had the best performance. Given their performance, we focus on the Random Forests for the rest of the paper.

Although the range of performance across the above models sometimes overlapped the CNN's (35-75% as opposed to 18-65%), the variability of each model was far less than that of the CNN. For instance, the Random Forest almost always obtained at least 70% accuracy over a 100-fold cross-validation. Additionally, the approach had the advantages of requiring less computational power, having better validated code due to pre-existing implementations, and having the ability to randomly split the data each time the algorithm was trained, rather than at certain intervals when the images were split and separated into the training, validation, and test sets.

The downside of machine learning is that it's difficult to ascertain why certain models performed better, but we can venture to infer that the reason sets of decision trees performed well is that the input data contained information amenable to decision-based separation (if the image is positioned on the top left of the page, it is definitely not a footer, and may be a headline or header). With enough of the right kind of data, and with enough decision trees, a vote amongst them may be well suited to successful classification. This is what the results suggest. The ExtraTrees Classifier works similarly to the Random Forest, and so lends support to our hypothesis that decision trees work well on this problem.

To validate our results, we ran cross-validation, which involves splitting the dataset into K-folds, one of which is assigned to the testing, and the rest are allocated for the training. The procedure iterates, so all K-folds are part of the testing exactly once, and are in the training the rest of the time. This helps eliminate variance in performance based on which subset of the data was assigned to the training set, and which to the testing. The picture so far describes K-fold cross-validation. In our case, since we have imbalanced classes, we also use Stratified K-fold, which ensures that the same proportion of each of the classes is assigned to the training and test dataset across runs, rather than some classes being allocated more heavily in the training in one run, and the

testing in another. This ensures even more consistency when imbalanced classes are present. In figure 2.11 we report the results for our Stratified K-fold method.

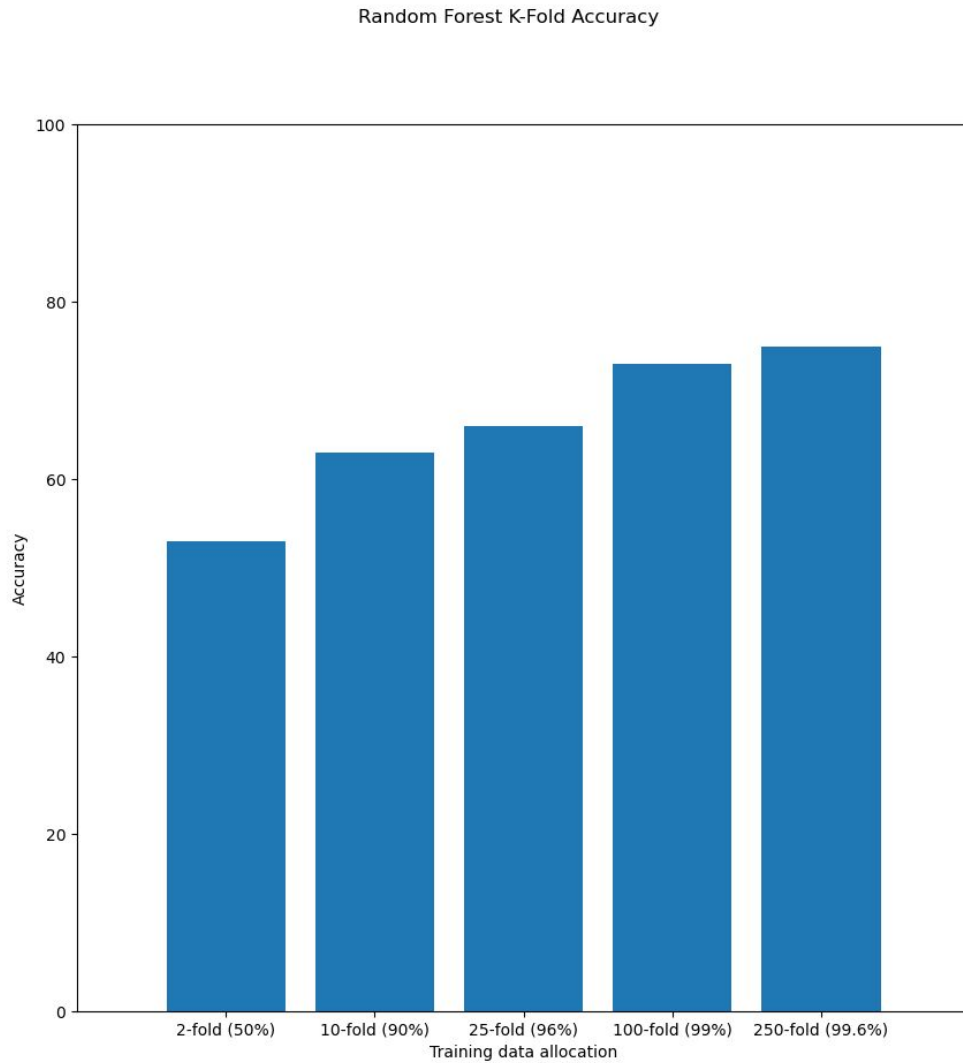
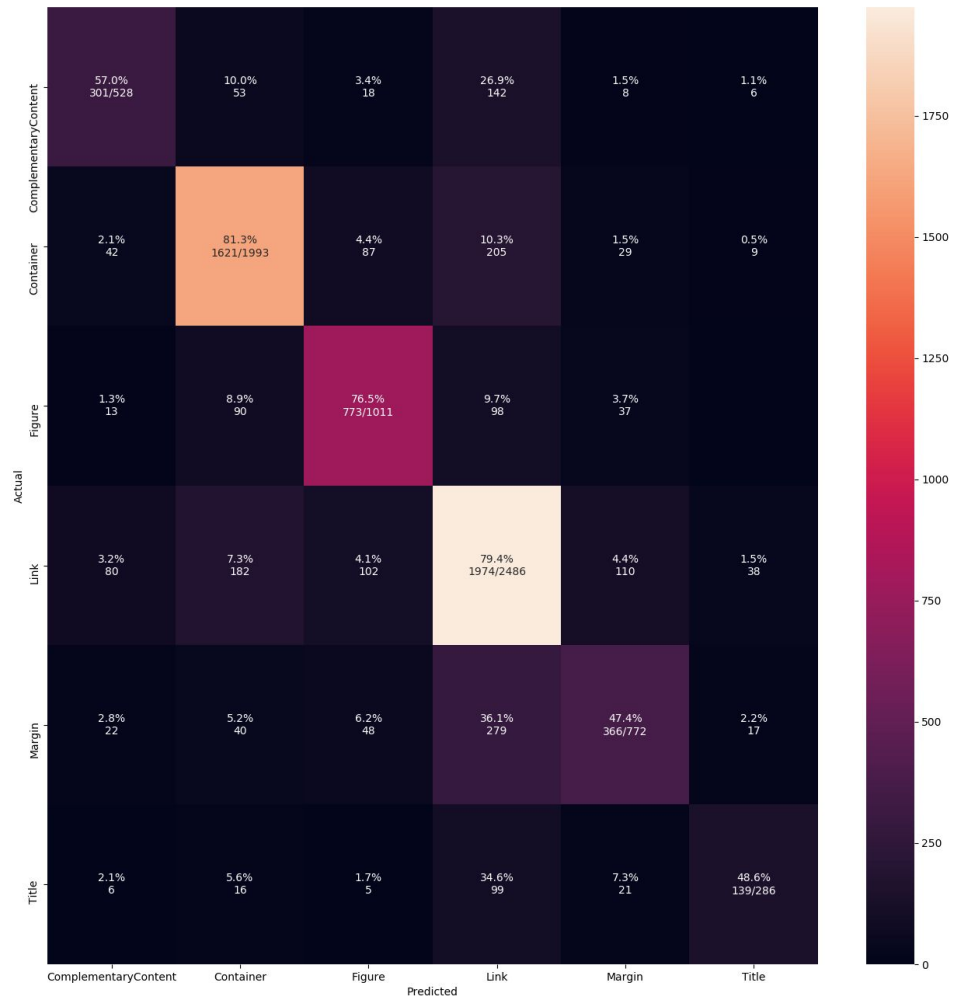


Figure 2.11: Changes in Random Forest accuracy as we increase the number of folds, which assigns proportionally more data to the training set per run. We see that even marginal increases in test size improve accuracy.

Figure 2.11 shows that performance increases as more data is allocated to the training set in comparison to the test, which means we have not yet plateaued in performance with our given dataset. Performance will likely improve if more data is incorporated.



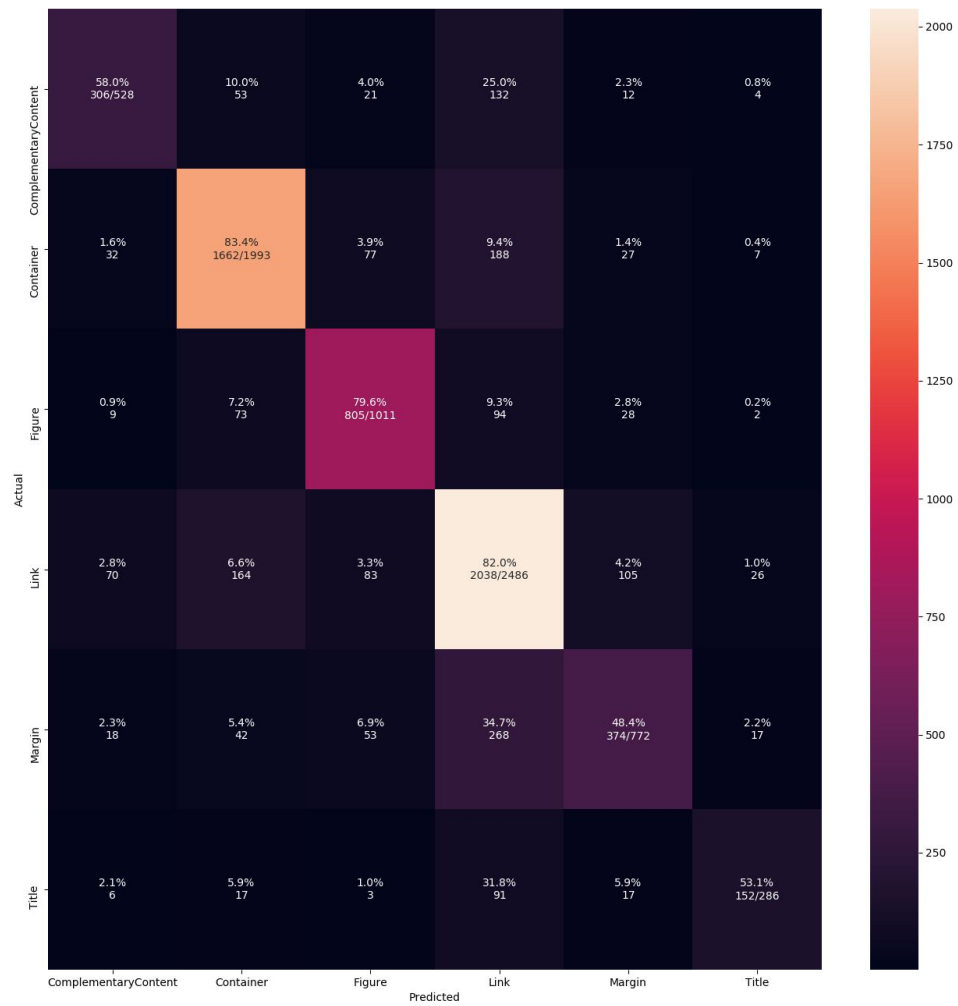


Figure 2.12 and 2.13 : Confusion matrices for the Random Forest from classes with at least 250 members with 100 and 250 splits, respectively. The important results here are that despite the data being reallocated, and despite the models being initialized again, the confusion matrices show a very high degree of similarity. This shows us that on average, the Random Forest makes consistent inferences.

3. Contributions and Future Work

3.1 Comparing our Results

Figure 3.1 shows our accuracy across all models for the 250 minimum class members scenario. For all but the CNN, a 100-fold stratified cross-validation was used to obtain these numbers (we refrained from doing a 250-fold version for reasons of computational constraints). As we've seen, in general, performance increased as the size of the training data was increased relative to the validation and test sets, indicating that we are not at our asymptotic performance peak, and that further improvements to performance should be anticipated with more data. This was separately reflected in cross-validation and stratified cross-validation results, where performance increased with the number of folds, as a larger portion of the data was allocated to training for each fold.

For the CNN, training graphs and confusion matrices indicate that instability is present throughout the training process, and that the class of errors can vary widely across different trainings of the model. This suggests that the CNN has a relatively poor apprehension of the problem space, and does not adequately map out the domain to consistently identify classes with more similar images and classes with less similar images. In the future, we may couple our results with a distance similarity measure for images, to ascertain how image similarity relates to the kinds of errors observed in the case of the Random Forest and the CNN models, to determine, in part, whether pairing the two in an ensemble setup will improve performance or simply result in overfitting. The goal, essentially, is to create models that map the problem space well, have the ability to generalize to other images from the same problem space, and provide insight into the problem space that can yield further models. While the models show a range in their performance, one promising scenario is to combine multiple models together in

ensemble learning, which would allow us to use complementary models to obtain both better accuracy and better generalizability.

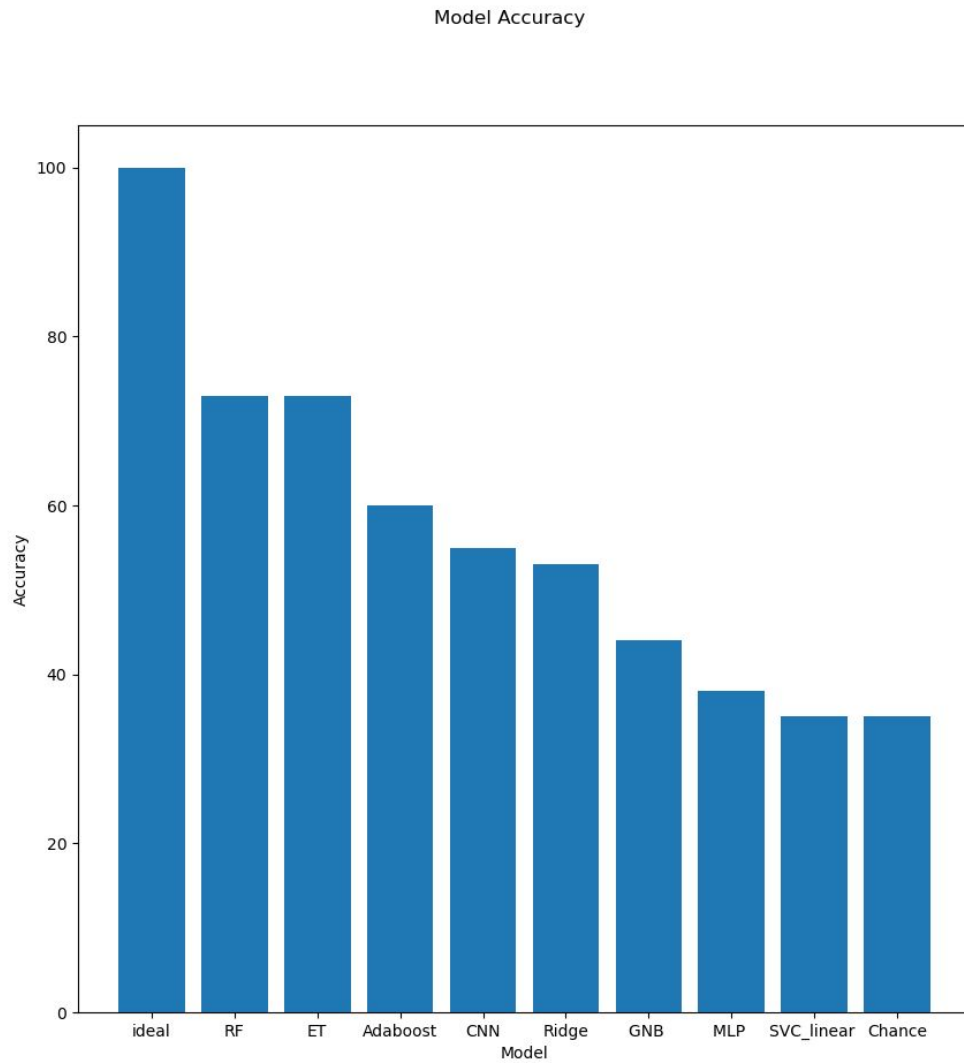


Figure 3.1: Results for the main models used.

In comparison to earlier versions of the SiteSeer system, our results are encouraging. As figure 3.2 shows, the system attained much higher accuracy as images were pruned further. If we compare our results to the ones below, our CNN obtained an accuracy of about 55% with 5K pruning, and the Random Forest and ExtraTrees Classifiers obtained upto 75% accuracy. This compares very favorably to the 20% obtained by the earlier system; however, since we only used classes with more than 250 members, these numbers would be lower if we considered the entire dataset. Still, since we retain 90% of our dataset, this would at most reduce our accuracy to 67.5% (and to 49.5% for the CNN), which is a very substantial improvement. When we set a minimum class threshold of ten members, and conducted a 10-fold procedure with the Random Forest, we obtained a 59% accuracy, which is only marginally worse than the 63% obtained in the 10-fold procedure with the 250-member class scenario. To further compare our performance to earlier versions of the SiteSeer system, we look at a different pruning scenario.

In the instance where we use the 100K pruning (for the scope of our project, the intermittent prunings were omitted because they were not converted to .csv format), we obtain an 89% accuracy with our Random Forest. This is also a very favourable result compared to the 60% obtained earlier. The CNN was not run, since there were only about 900 images remaining, which is too small for a CNN. Since the pruning also reduced class diversity, resulting in the container class having roughly 800 of the 900 images, we favored less pruning as it demonstrates a greater challenge and more robust results. The corresponding confusion matrix is shown in figure 3.3. Due to these considerations, we decided to use the 5k pruning throughout this project. In the future, we can obtain results for different levels of pruning, to better understand its effects on our current approach.

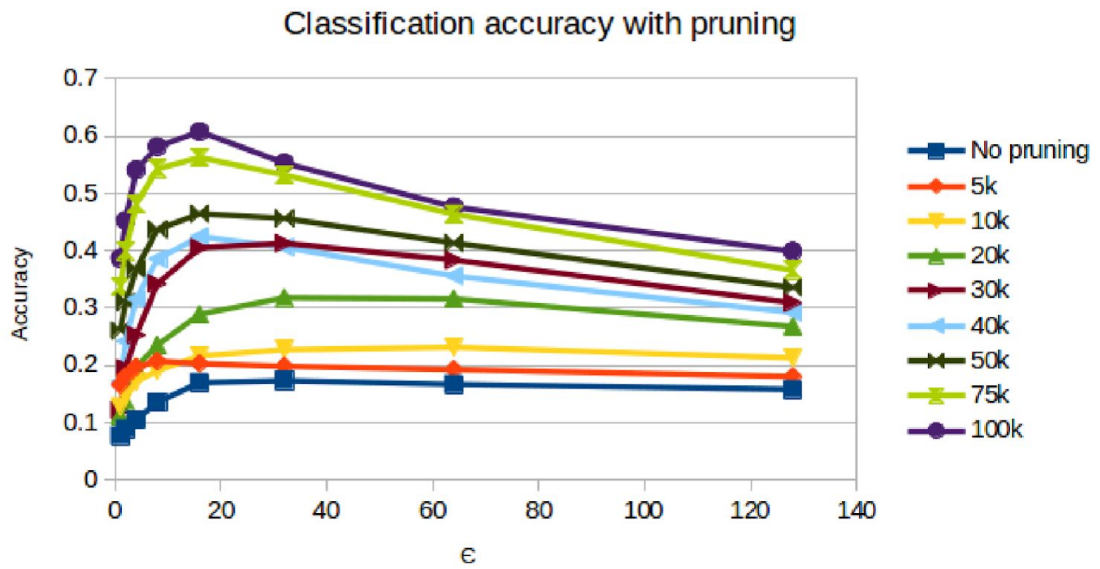


Figure 3.2 (from [3]): Old classifier accuracy with varying degrees of pruning, and a varying epsilon parameter. Here, we are considering results at the best epsilon value, at around 20.

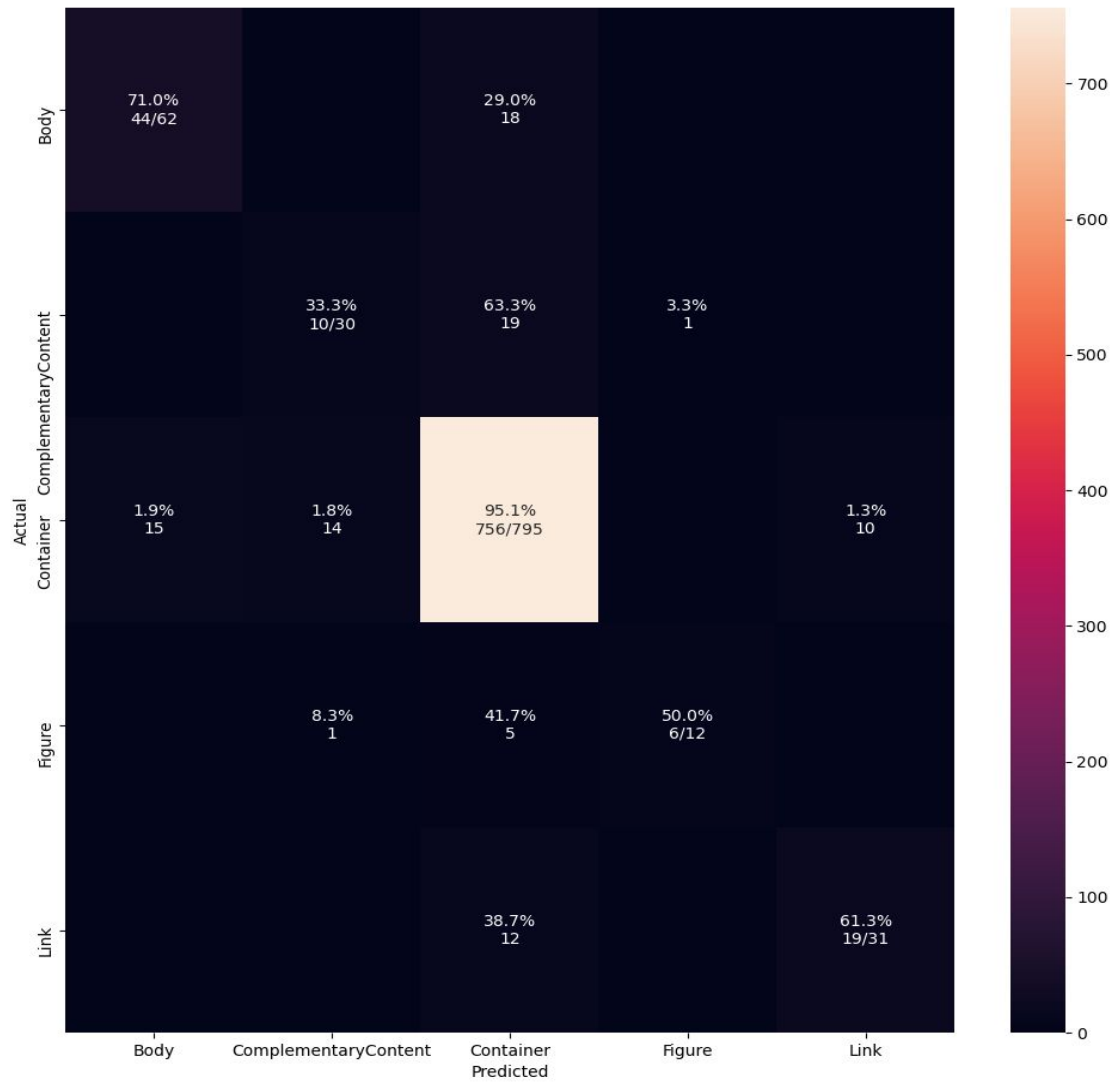


Figure 3.3: Confusion matrix from the 100k pruned Random Forest with 10 class members min and 10-fold stratified k-fold. It is important to note that it did not always predict the most probable class.

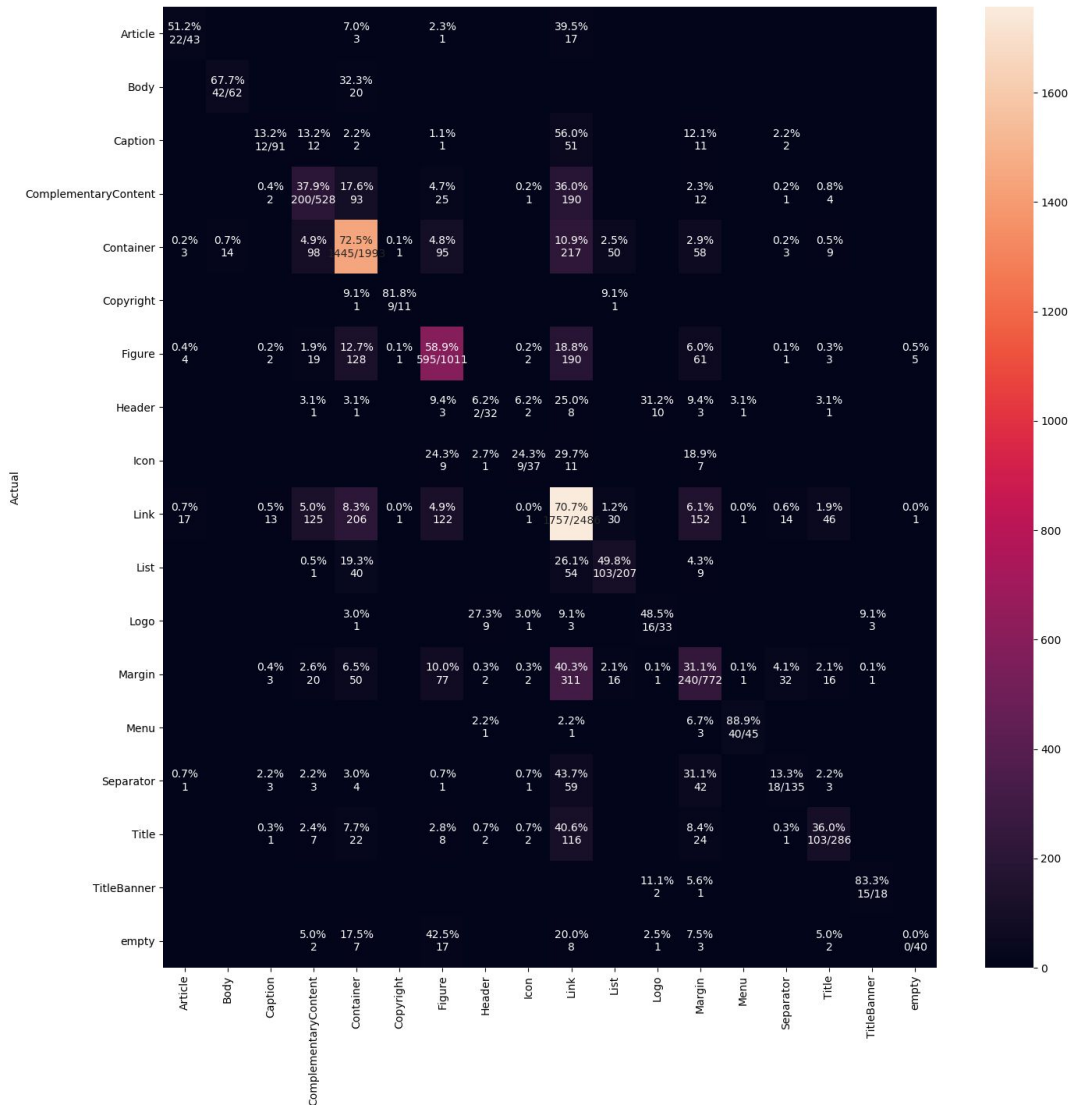


Figure 3.4: Confusion matrix for Random Forest with 5k pruning, 10 min class instances and 10-fold train. This is clearly a much more difficult classification problem, and the 59% accuracy which was obtained here still exceeds the performance of the CNN, despite there being many more classes.

One approach to circumventing the data limitations would be finding ways to first collect large sets of images within a web page category (for instance, advertisements can be identified by trawling web pages for links associated with adblocker lists, which would contain images of advertisements). Social media feeds and comment sections (categories not included in our ontology) can similarly be identified by finding common variations of social media platform logos, and by collecting large sets of images of comment sections, which can be segmented on a per comment basis. However, some elements will not be distinguishable on either a visual or a DOM basis, and will not yield a set of similar images from which to create a dataset. For these, we could collect data from users who are using assistive screen reading technology by occasionally asking them to verify if we identified a section correctly (and to indicate which it is, if we did not). In this way, techniques (such as the DOM), which are not directly part of the SiteSeer system, could be used as part of the data collection. This would allow us, in a way, to have the best of both worlds.

Even with more data, and the inclusion of an HMT potentially in conjunction with an HMT, the task of successfully segmenting and classifying web pages is difficult because of the ontology. Websites vary significantly, and finding or developing an ontology that works well on all websites and is reasonably succinct is difficult. However, in the age of machine learning, we see the utility of a different ontology that not only has finer distinctions than the 27 available, but also better aligns with the visual appearance of a web page, while preserving its correspondence to the functional and semantic role of each class. Such an ontology may come about in the future, and could significantly improve the quality of work in this area.

While it would be useful for the model to know what elements underlie certain sections of a page (headlines often have big text, and a link to the article, along with an associated image), this presents many challenges indicated earlier when we discussed the DOM. What could help is contextual cross referencing (loading headlines every day

from a news aggregator) combined with OCR and NLP on the segments of the image, to better associate happenings around the world with the news web pages (which were the sample from which our data was selected). This approach would apply to many kinds of websites, since most websites, even when they are not traditional news sites, are topical. And such a model would be the beginning of an intelligent agent which is able to traverse the internet to collect information, make inferences, and highlight findings. It could, for instance, identify news trends in a country speaking a foreign language, or provide insights to investors about news stories throughout the day. Investors, governments, and social media influencers could use this system to see trends developing ahead of time (if certain websites consistently mirror more reliable sources) and leverage them. Thus, there is potential for a lot of practical synergy between our present approach and a purely NLP approach, and our approach and a purely HMT approach (since we will still want to know where things occur on a page to best identify them). There is, as well, plenty of room to dream for the future.

At the core, we have two challenges—data collection, and the integration of multiple information processing modalities that humans possess, namely sight, language, and spatial contextualization. More data helps generalize, while underdetermination and ambiguity of small images can best be resolved by integrating multiple kinds of information. This is sometimes not a challenge in natural images, or in identifying the language of a speaker, because a cat (or a person's speech) is an organic whole, and reveals itself even in small sections: the fur of the cat, or the prosody of the voice. But man-made images are composed of similar, discrete parts, which when segmented, are similar, and may belong to a variety of owners.

3.2 Conclusion

Despite numerous challenges in the dataset which may have limited the effectiveness of machine learning on our problem, we obtained results that substantially improved on earlier work, and on a baseline of picking the most common class. A 75% accuracy, as opposed to a 35% that would have been obtained by picking the most probable class, is encouraging, but we suspect that with our current approach, and current dataset size, we are approaching the limit of possible performance, due to the challenges already described.

To further disambiguate classes, we plan to incorporate results from the machine learning into the Hidden Markov Tree, combine our models with ensemble learning, and also increase the size and quality of our dataset by improving our segmentation algorithm, and gather more data. This should also help stabilize the results of our CNN models.

We hope that our findings here contribute to an understudied area of computer vision, and that eventually this work will be useful in broader contexts for interpreting and manipulating man-made images intended to convey semantic content, of which the web page is a fine example.

3.3 GitHub Repository

All data files and code related to our work can be found at the following GitHub repository: https://github.com/ShekharDewan/fully_convolutional_network

References

Main Text

1. W3C. Accessible rich internet applications (WAI-ARIA) 1.0. w3c recommendation. <https://www.w3.org/TR/wai-aria-1.0/> (accessed Nov 29, 2019), 2014.
2. M Elgin Akpinar, Yeliz Yeşilada, Discovering Visual Elements of Web Pages and Their Roles: Users' Perception, *Interacting with Computers*, Volume 29, Issue 6, November 2017, Pages 845–867, <https://doi.org/10.1093/iwc/iwx015>
3. Michael Cormier (2018). Computer Vision on Web Pages: A Study of Man-Made Images. UWSpace. <http://hdl.handle.net/10012/13523>
4. Jonathan Robie, Lauren Wood, Steven B Byrne, Philippe Le Hégarret, Arnaud Le Hors, Mike Champion, and Gavin Nicol. Document object model (DOM) level 2 core specification. W3C recommendation, W3C, November 2000. <http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113/>.
5. Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: A vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft, 2003.
6. M. E. Akpinar and Y. Yesilada. Vision based page segmentation algorithm: Extended and perceived success. In *Lecture Notes in Computer Science: Current Trends in Web Engineering*, pages 238–252. Springer International Publishing, 2013.
7. Omer Barkol, Ruth Bergman, Ayelet Pnueli, and Sagi Schein. Semantic automation from screen capture. Technical Report HPL-2009-161, HP Laboratories, 2009.
8. Jiuxin Cao, Bo Mao, and Junzhou Luo. A segmentation method for web page analysis using shrinking and dividing. *International Journal of Parallel, Emergent and Distributed Systems*, 25(2):93–104, 2010.

9. Justin Romberg, Hyeokho Choi, Richard Baraniuk, and N Kingbury. Multiscale classification using complex wavelets and hidden markov tree models. In *Image Processing, 2000. Proceedings. 2000 International Conference on*, volume 2, pages 371–374. IEEE, 2000.
10. Justin K Romberg, Hyeokho Choi, and Richard G Baraniuk. Bayesian tree-structured image modeling using wavelet-domain hidden markov models. *IEEE Transactions on image processing*, 10(7):1056–1068, 2001.
11. Yi Yang, Sam Hallman, Deva Ramanan, and Charless C. Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.
12. Ke Li, Bharath Hariharan, and Jitendra Malik. Iterative instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3659–3667, 2016.
13. Anurag Arnab and Philip HS Torr. Bottom-up instance segmentation using deep higher-order crfs. *arXiv preprint arXiv:1609.02583*, 2016.
14. Michael Cormier, Karyn Moffatt, Robin Cohen, and Richard Mann. Purely vision-based segmentation of web pages for assistive technology. *CVIU special issue on Assistive Computer Vision and Robotics*, 2016.
15. Michael Cormier, Richard Mann, Robin Cohen, and Karyn Moffatt. Classification via hidden markov trees for a vision-based approach to conveying web pages to users with assistive needs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence*, 2016.
16. Michael Cormier, Richard Mann, Karyn Moffatt, and Robin Cohen. Towards an improved vision-based web page segmentation algorithm. In *Proceedings of the 2017 Conference on Computer and Robot Vision (CRV)*, 2017.
17. Cormier [in review] - Validation of an improved vision-based web page parsing pipeline

18. He, Kaiming et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." *Lecture Notes in Computer Science* (2014): 346–361. Crossref. Web.
19. Jinlin Chen, Ping Zhong, and T. Cook. Detecting web content function using generalized hidden markov model. In *5th Int. Conf. on Machine Learning and Applications*, pages 279–284, Dec 2006.
20. Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
21. Piramanayagam, S.; Saber, E.; Schwartzkopf, W.; Koehler, F.W. Supervised Classification of Multisensor Remotely Sensed Images Using a Deep Learning Framework. *Remote Sens.* 2018, 10, 1429.
22. J. Long, E. Shelhamer and T. Darrell, "Fully convolutional networks for semantic segmentation," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 3431-3440.
23. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (May 2017), 84–90. DOI:<https://doi.org/10.1145/3065386>
24. Scikit-learn: Machine Learning in Python, Pedregosa et al. *JMLR* 12, pp. 2825-2830, 2011.
25. X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):1943–1955, 2016.
26. Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. *Insights Imaging* 9, 611–629 (2018). <https://doi.org/10.1007/s13244-018-0639-9>
27. Jürgen Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks*, Volume 61, 2015, Pages 85-117, ISSN 0893-6080, <https://doi.org/10.1016/j.neunet.2014.09.003>.

28. Altman, N., Krzywinski, M. Ensemble methods: bagging and random forests. *Nat Methods* 14, 933–934 (2017). <https://doi.org/10.1038/nmeth.4438>
29. Billah SM, Ashok V, Porter DE, Ramakrishnan IV. Ubiquitous Accessibility for People with Visual Impairments: Are We There Yet?. *Proc SIGCHI Conf Hum Factor Comput Syst.* 2017;2017:5862–5868. doi:10.1145/3025453.3025731
30. R. Berwick, "An Idiot's Guide to Support Vector Machines (SVMs)", 2003, Massachusetts Institute of Technology, Cambridge, MA.

Figures

- A. The Document Object Model used to access objects in web pages with eg. javascript, retrieved from <https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/DOM-model.svg/642px-DOM-model.svg.png> Used under Creative Commons Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0) (<https://creativecommons.org/licenses/by-sa/3.0/>)