Evaluation of Generic Deep Learning Building Blocks for Segmentation of 19th Century Documents

By

Evan Segaul

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Computer Science

May 14, 2021

Nashville, Tennessee

Approved:

Douglas C. Schmidt, Ph. D.

Jesse Spencer-Smith, Ph. D.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Image segmentation is the process of partitioning an image by assigning a label or class to each of its pixels to represent the image meaningfully [1]. For example, an automated driving system may find it helpful to label objects in its environment, like street signs and pedestrians, to assist with the driving process. Other examples of image segmentation may exist in content-based image retrieval systems [2], medical imaging [3], object detection [4], surveillance [5], and biometric security systems [6].

This paper will explore methods and tools for image segmentation, specifically in the context of paper documents and especially with handwritten records from before the twentieth century. For the previously mentioned applications of image segmentation, there are many distinguishing features between different objects like color and brightness. However, in historical documents, there are no color or contrast differences between parts of the document. There only exist logical differences that can be inferred from markings on the page, which are created inconsistently between records over time and are often degraded [7]. Thus, the segmentation of paper documents may necessitate different methods than the segmentation of other types of data. Furthermore, the range of desired analysis on paper documents is extremely expansive, so it is important to consider the specific dataset used for this paper and the problem that this analysis was originally intended to help solve.

1.1 Motivation

The motivation for this paper comes from the Slave Societies Digital Archive (SSDA), which is hosted at Vanderbilt University. Slave societies are defined as civilizations that slave labor and/or trade is an essential part of their economies, politics, and lives as a whole. The SSDA preserves documents related to African people and their descendants

in slave societies, mostly in the Iberian New World. According to the SSDA website [8], "SSDA holdings include more than 700,000 digital images drawn close to 2,000 unique volumes dating from the sixteenth through twentieth centuries that document the lives of an estimated four to six million individuals."

The majority of the documents in the SSDA are Catholic Church documents, which mandated the baptism of African slaves and their descendants. With baptisms additionally comes the eligibility for marriage and burial with the Catholic Church. Since the Catholic Church is a centralized, hierarchical organization, there is a large level of consistency between documents that have been created in different parts of the world and at different periods in time. Although the quality and the layout of documents may be different between record keepers, there are a base set of facts that remain consistent throughout the documents. These facts include names, locations, dates, and the names of family members [8]. These common facts between documents create a structure that lends itself to algorithmic analysis rather than attempting to analyze each of the 700,000 images manually. However, it is not yet feasible to simply extract the characters from the page using OCR technology [7] and subsequently analyze the text. Thus, one must try to derive information from other features to create a meaningful analysis of this archive.

The ultimate goal of this project is to develop a model for the procedural creation of family trees based on the images in the SSDA. Using each record of baptisms, marriages, and deaths, it may be possible to match the names, locations, and dates to create a story that follows the genealogical progression of the descendants of African slaves in the Iberian New World. Ideally, this tool may be used by a descendant of African slaves who knows or can infer the name of an ancestor who appears in these documents, opening up a new chapter of his or her ancestral history that wouldn't have been uncovered otherwise.

To perform the analysis, the following approach was planned. First, the records must be separated from each image. Every image in the SSDA contains at least one record, empty page space, and extraneous parts of the image that do not provide useful information

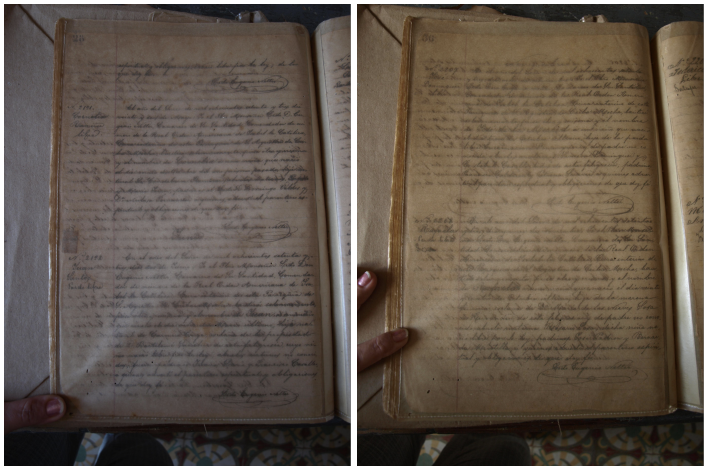

Figure 1.1: Images from the SSDA

These are two example images found in the Slave Societies Digital Archive [9]. These images display extraneous objects such as fingers and glimpses of the surface that the book is resting on. Additionally, there are differences in lighting and page orientation between these two pages. Finally, smudging and bleed-through between pages are also clearly evident.

such as parts of a table or fingers. Thus, it is necessary to isolate the records as blocks of text from the SSDA images to analyze the data more efficiently. Since OCR technology is not yet sufficiently advanced to directly glean character information from such degraded images [7], another approach must be taken to match the names. Most likely, this approach would have involved treating the written names as patterns and matching them with other names that appear similar to surpass the idiosyncrasies of handwriting between different record keepers. Finally, after matching the records by name, a family tree structure may begin to emerge and unlock the ancestral history of millions around the world.

The first leg of the approach, separating records from the rest of their images, requires using image segmentation. While segmentation research has grown tremendously in recent years, research revolving around the segmentation of documents has not seen expansion. However, the dhSegment [10] model and toolkit have demonstrated promising success in this domain using generic computer vision models rather than document segmentation-specific tools.

1.2 Contribution

The contribution of this paper will build on the idea from the dhSegment paper that generic computer vision models can effectively perform document segmentation. We will do this by evaluating generic computer vision models other than the ResNet50-based model used in dhSegment and explore what other models may help advance this domain further. We will do this based on the success of these models on the images found in the SSDA. We hypothesize that the success of ResNet50 in classification tasks will allow it to be the most successful generic building block used for constructing segmentation models compared to other common convolutional neural networks.

While this paper does not propose a comprehensive model that solves the first leg of the eventual family tree problem, it will lay the foundation for future attempts of this problem and many others that lie adjacent to it. In Figure 1.1, it is evident that there are many difficulties in the quality of historical document data. Between the confounding issues like extraneous objects and page bleed-through, more research must be conducted to advance this domain of analysis further. Thus, we will explore new avenues of research and potential analyses that may help to further the success of historical document segmentation. Finally, the appendices of this paper will introduce a comprehensive review of the state-of-the-art techniques that are most successful in the segmentation of ancient documents and how they work at a fundamental level, such that anyone who wishes to pursue research in this domain can use this paper as a high-level overview.

Chapter 2

Related Work

2.1 Convolutional Neural Networks

Since image processing is a ubiquitous field with extreme variation between different types of images and tasks, there is a significant number of segmentation methods that one can use for any given task. We will focus primarily on segmentation using convolutional neural networks (CNNs), although a review of classical segmentation methods and other artificial intelligence methods can be found in the appendix.

One of the main CNNs that first made it to the academic community was AlexNet [11, 12]. AlexNet's main innovation was its unique implementation of a CNN that allowed it to be trained more efficiently, bringing the cost down and raising the amount of data learned. Some of these techniques include using Rectified Linear Units (ReLUs) as activation layers, custom GPU optimized algorithms for convolutions and training, and pooling outputs together [12].

Building on top of AlexNet was the advent of the Visual Geometry Group (VGG) [13]. The VGG model incorporated the smallest possible convolutions to the earliest layers of the model, allowing for quicker training than its predecessors. Convolutions allow feature maps to be created from input data [14], so using the smallest possible convolutions allows for the parameters of the convolutions to be more easily determined. Thus, using smaller convolutions may have allowed for the most efficient creation of useful feature maps.

The next significant model discussed is GoogLeNet. GoogLeNet used the insight that the way to theoretically reduce parameter count and use of resources while training is to go from densely-connected networks to sparsely-connected networks. However, the underlying hardware for modern computations is extremely inefficient in working with sparse calculations. This paper was able to approximate sparse structures with existing dense

components, allowing for good results with less expensive networks [15].

An incredibly significant model to discuss is ResNet [16]. ResNet provided breakthroughs in training extremely deep networks by adding skip connections between layers. These skip connections allow the training of the model to ensure that skipped layers are performing meaningful tasks, creating a more efficient training process. This is an extremely promising approach that has advanced the field of computer vision significantly.

Next, we have SqueezeNet [17]. SqueezeNet's authors sought out AlexNet level accuracy with their network but wanted to use significantly less space. They did this with the idea that convolutions that operate on every input channel use extremely large amounts of space, so they created 1x1 convolutions to squeeze all the input channels into larger convolutions that now require fewer parameters. Additionally, they used compression techniques like Deep Compression [18] to make their model even smaller while still maintaining accuracy.

Finally, we have DenseNet [19]. DenseNet uses the same principle the ResNet operates with to add skip connections between layers. However, DenseNet adds from one layer to every subsequent layer, rather than adding connections between every other (or third) layer. Additionally, DenseNet uses fewer convolutional filters per layer due to the large amount of information being passed between layers. The authors' analysis on DenseNet shows that early features that are extracted are still used by layers closer to the end of the model, which is an extremely interesting result and poses questions about how low-level features can be combined with higher-level features.

2.2 Using CNNs for Segmentation

The main architecture we discuss to adapt CNNs for segmentation is the U-Net [20]. The U-Net builds on top of the encoder-decoder architecture, which uses convolutional layers to encode low-resolution maps of fundamental features in images and subsequently decodes the feature maps to labels for each pixel using upsampling operations like pooling

and deconvolution [21]. U-Net uses this architecture but adds skip connections between corresponding downsampling and upsampling layers. So, during the decoding process, the upsampled data is combined with data that is never fully convolved, thus maintaining information about high-level features in a given image. This allows the model to combine the low-level knowledge about how to classify pixels with the high-level data such as where these pixels may be located in the image. This base model architecture is heavily built upon and significantly advanced the field of image segmentation.

The most interesting approach found specifically for the domain of historical document segmentation is called dhSegment [10], which applies common deep learning architectures and standard image processing techniques to perform pixel-wise segmentation. The dhSegment architecture implemented consists of a model that is extremely similar to a base U-Net except that it uses a ResNet50 model as the encoder and utilizes standard upsampling and concatenation of encoder features as its decoder.

The main insight from the dhSegment paper is that the authors built a highly successful model for document segmentation using a generic, pretrained encoder-decoder structure. This model can be trained on a variety of different tasks regarding document segmentation such as page extraction, layout analysis, and line detection. Since the variety of tasks that can be performed on documents is so expansive, the authors additionally used many different types of image processing techniques to further improve the accuracy of their model. These image processing techniques such as thresholding or shape vectorization are standard processes that do not require machine learning analyses. Thus, the task-specific application of post-processing techniques on a general model opens the door for a large, generalizable tool for document segmentation that does not require much training, but rather requires the domain knowledge to construct accurate results using simple processing techniques on its output. The success that comes from dhSegment using only generic deep learning models as building blocks is extremely impressive, so we will further evaluate the viability of other generic building blocks in their applications to the segmentation of historical documents.

Chapter 3

Model Implementation and Training

3.1 Dataset

The dataset used for this analysis is a collection of documents from the SSDA that consists of the baptismal records of people of color from the Iglesia de San Agustín in Ceiba Mocha, Cuba from 1872 to 1892 [9]. Each image in the dataset is a photograph of a page that contains one or more records. As seen in Figure 1.1, these images also contain extraneous information, such as a hand that is holding the page flat or the table that the record book is resting on. The images are labeled such that the two categories of records to be extracted are different colors from the rest of the image, including the blank page space, table, and other extraneous information. This data archive is available publicly on the SSDA website; however, there are no labels for the data. For the sake of time and cost, a dataset of only around 100 images was created to train and evaluate the performance of a model.

3.2 dhSegment

We use the pre-trained dhSegment model to form a baseline measure of success by which we can evaluate other models. The dhSegment model implementation is included in the paper and can be found on GitHub [10]. The model is implemented as an API wrapped around a deep learning model built with TensorFlow and Keras. Since the provided dhSegment model was trained for document-specific segmentation tasks by its authors, its pre-trained weights allow for quick and efficient training. Using the built-in training method from the dhSegment API, the model can predict the correct pixel value 92% of the time. However, the mIoU accuracy, which is more reflective of the success of the model, is only about 70.7%. Given the fact that the training dataset is only around 80 images, this per-

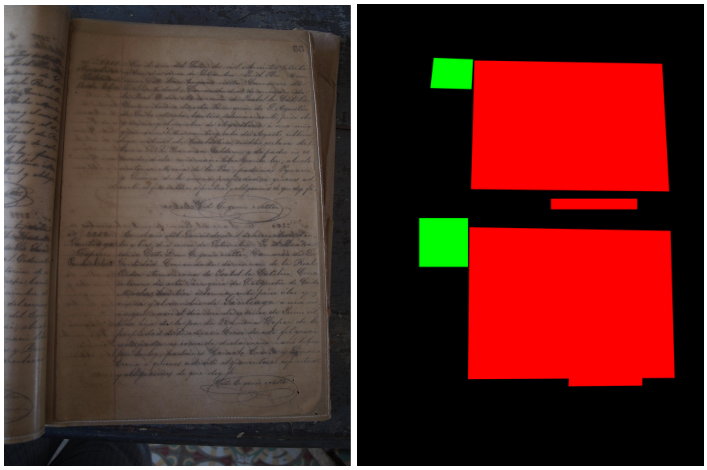

Figure 3.1: Training Data

This is an example of the data used in the training and test set: an image from the SSDA (left) and its corresponding segmentation mask (right). The red masks represent the main-body blocks of text while the green masks represent column blocks of text.

formance is extremely impressive and demonstrates that the power of this architecture is quickly learning how to analyze ancient documents.

After the initial round of training, the out-of-the-box results were extremely promising. However, with about 70% mIoU accuracy, there is room for improvement. By the nature of how the dhSegment toolbox is constructed, the ability to look inside the model and make improvements is restricted. There is a wrapper class that is created around pure Tensorflow. However, this wrapper did not include functionality such as built-in GPU-optimized data augmentation or the ability to experiment with different loss functions. Thus, dhSegment provides an extremely solid proof-of-concept that a convolutional neural network can label these historical documents sufficiently well, so we construct other generic models in the fastai framework and evaluate their performance.

3.3 Fastai

3.3.1 Implementation

FastAI [22] provides much of the custom functionality necessary to experiment and augment the results that dhSegment did not provide. For example, it gives ways to easily change the loss function of the model during training. Another benefit is the ability to add data augmentation when loading the dataset and perform it on the fly with GPU optimizations. These additions allow for the relatively quick tuning of hyperparameters that can allow the model to perform better.

One extremely important feature of fastAI is the function 'unet_learner'. This function allows the user to pass in an architecture of a standard, pre-trained convolutional neural network for use as the encoder of a U-Net model that can then be trained and tested. Additionally, fastAI provides a custom implementation of the cross-connections among the encoding and decoding passes of the U-Net such that it can operate with any encoder architecture that is passed in.

One of the notable discussion points of the dhSegment is that the architecture that they created was a combination of generic building blocks. The authors of dhSegment used a ResNet50 architecture for training and evaluation, but their work opened the door to say that many other generic architectures can work for similar functions. The fastAI library is compatible with any of the models available in the torchvision [23] library. Thus, the dhSegment architecture can be recreated with any CNN as its encoder and its performance evaluated. These torchvision models include the ones discussed before such as AlexNet, VGG, GoogLeNet, and ResNet. Additionally, the torchvision library includes many slightly different alterations of these models that can also be evaluated. We will now use fastai to construct these altered U-Net models and evaluate them on the SSDA dataset.

3.3.2 Training

To train and evaluate different generic building blocks in place of a ResNet50 as the encoder of a U-Net, we used the built-in torchvision models that were available in fastai. These models are different variations of ResNet, SqueezeNet, DenseNet, VGG, and AlexNet. After the U-Net architecture was created using these pretrained models as the encoder, we decided to use the cross-entropy loss function [24]. In the initial round of training, we performed segmentation into the three classes seen in Figure 3.1: main text, column text, and not text. However, we found that the training process resulted in models that minimized the amount of "not text" that was labeled incorrectly, rather than labeling it correctly as "main text" or "column text". With a larger dataset, we could have used a weighted cross-entropy loss function to account for the imbalance text classes, but with the limited amount of data we elected to combine "main text" and "column text". Thus, we performed a binary classification on "text" or "not text" with relatively balanced classes, so a cross-entropy loss function was an appropriate function to minimize.

The hyperparameters were selected to be either the default or through cross-validation schemes. fastAI's built-in function to find the optimal learning rates, 'find_lr', was used. Additionally, each training epoch was performed with the built-in 'fine_tune' function, which includes training defaults specifically used for transfer learning.

The actual training process consisted of training each model between 26 to 30 epochs and evaluating their accuracy on the testing dataset at several checkpoints. This large amount of epochs relative to the amount of training data was done to train each model to its best performance and to observe how quickly its performance may degrade due to overfitting. To train each model, we began by solely training its last layer in the first set of epochs. Then, the entire model was unfrozen and trained. Subsequently, we progressively froze the earlier layers and continued training until either the models overfit or around 30 epochs were completed.

Chapter 4

Results and Discussion

Following are the results of our evaluation of different the different base encoders of the UNet on the given segmentation task. Figure 4.1 represents the mIoU accuracy over each checkpoint epoch for the different ResNet models evaluated. Figure 4.2 represents the mIoU accuracy for each of the SqueezeNet and DenseNet models evaluated. Figure 4.3 represents the mIoU accuracies achieved for the VGG and AlexNet models throughout their training. Finally, Figure 4.4 displays the same data between the best of each type of model.

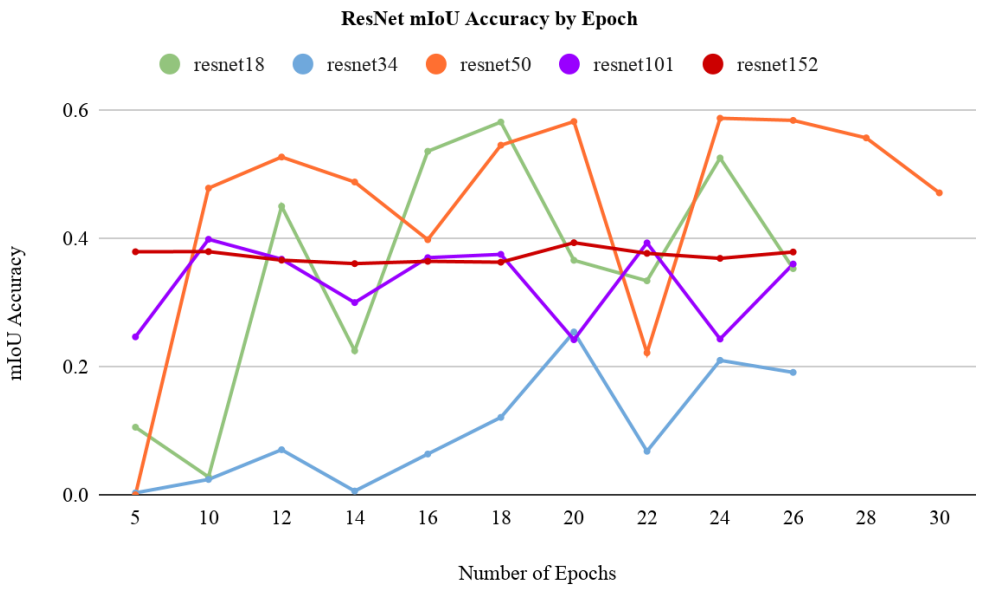

Figure 4.1: ResNet-Encoder Comparison

The testing accuracy for each U-Net with a ResNet encoder at given epochs. The accuracy was measured at epoch 5, epoch 10, and every other epoch until the model was determined to sufficiently overfit. The ResNet101 base seems like it does not overfit, but it never fit in the first place and its relative success was deemed to be random by visual inspection.

In Figure 4.1, it can be seen that the most successful models used resnet50 and resnet18

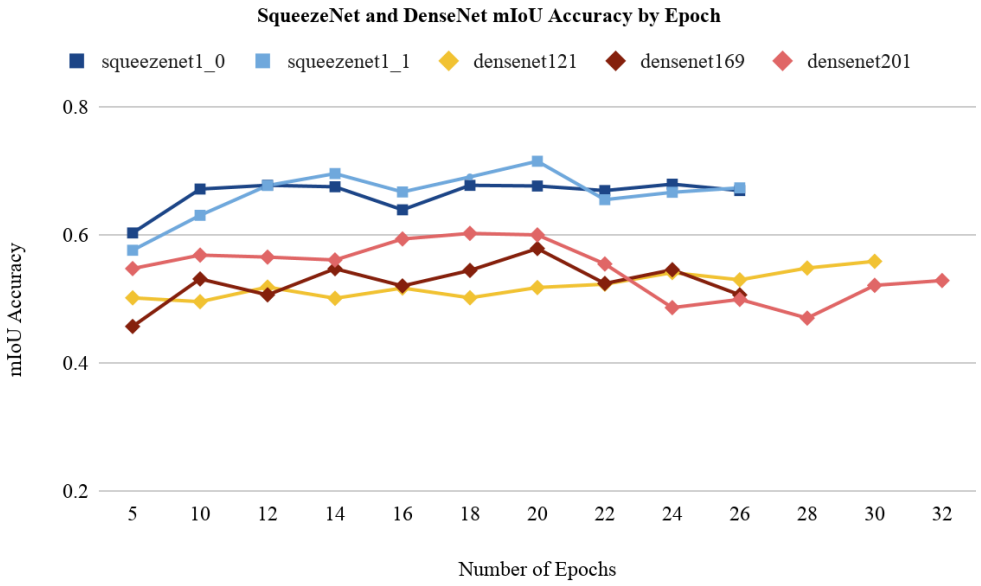

Figure 4.2: SqueezeNet-Encoder and DenseNet-Encoder Comparison

The testing accuracy for each U-Net with either a SqueezeNet or DenseNet encoder. The accuracy was measured at epochs 5, 10, and every other epoch thereafter. The models, after learning what the targets for segmentation were, did not improve or worsen significantly after many epochs.

as their encoders. The model with resnet34 performed the worst by far, whereas the models with resnet101 and resnet152 performed relatively well initially but did not improve very much. This demonstrates how a CNN can experience tradeoffs as a result of increasing the number of layers. When there are more layers in a CNN, there are more extracted features as the result of convolutions, but the question remains whether these features can be used efficiently and effectively. As the size of the model grows, the complexity and thus the number of parameters used also increases. Thus, training the models requires changing more parameters. This increases the possibility of overfitting. This seemed especially true with resnet101 and resnet152, as they both had very high training accuracies, but had essentially random guesses on the testing set. The most interesting part of this tradeoff is how it operated when going up from 18 layers to 50 layers. Since 18 layers are relatively few, the tendency to overfit was lesser, which allowed the model that is based in resnet50

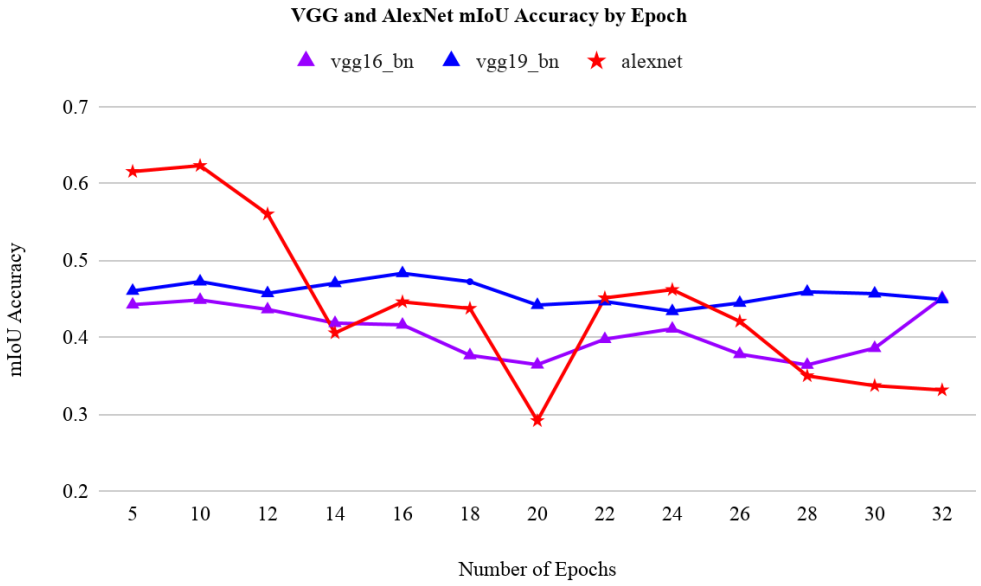

Figure 4.3: VGG-Encoder and AlexNet-Encoder Comparison

The testing accuracy for each U-Net with either a VGG or AlexNet encoder. AlexNet had very high accuracy in the beginning but fell off after training. This is not due to chance, as this happened multiple times when training from scratch.

to become accurate very quickly without deriving newer features in its layers. Then, the model containing a resnet34 encoder may have had too many layers such that it overfit, but not enough layers to derive any high-level features that may have helped its performance. Finally, the model with a resnet50 encoder had enough parameters such that it may easily overfit, but the added layers gave an extra level of depth that allowed it to work out newer features and gain accuracy with more training than resnet18 required.

Figure 4.2 shows how both variations with SqueezeNet encoders had the best accuracy out of all of the out-of-the-box models available on fastAI. Additionally, DenseNet encoders performed extremely well. It is interesting to note how both of these base encoder architectures created models that performed very accurately without much training but did not significantly increase their levels of accuracy or begin to overfit after excessive training. One such explanation for why these models perform better than ResNet is that they both

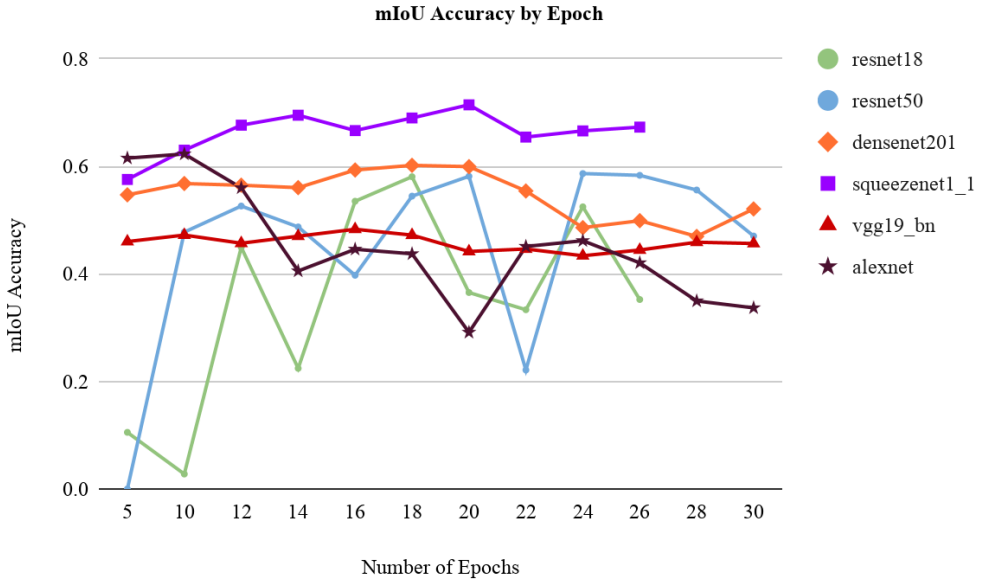

Figure 4.4: Inter-Model Comparison

The testing accuracy for the best from each model type. Two types of ResNet models were included since they performed nearly equally, which is notable and will be discussed later. This comparison of different model types allows for easy visualization of relative accuracy and variability.

have significantly fewer parameters than ResNets, which is consistent with the same logic that allows the model based on the resnet18 encoder to perform well.

Figure 4.3 demonstrates how AlexNet required very little training to perform well while VGG did not change its performance even after training extensively. AlexNet, being one of the more successful yet extremely early CNNs, is composed of very few layers compared to the CNNs that have been created since. The fact that it is much smaller and has fewer parameters can help to explain how it quickly picked up the important features in training but also how further training without the depth required to make complex features allowed for overfitting.

Figure 4.4 provides an overview of how all the types of models compare against each other. It is very interesting to see how ResNet-based encoders provide much more variability in the model. The model may be basing its decision-making on features that, when

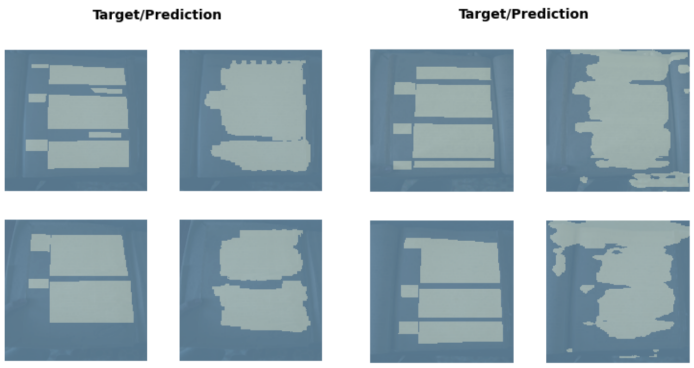

Figure 4.5: SqueezeNet and ResNet Outputs

The real and predicted output of a test set image for both squeezenet1_1 (left) and resnet50 (right) encoder-based models. By visual inspection, the SqueezeNet based model encapsulates the logic of where text blocks may occur better than the ResNet-based model.

slightly altered, provide very different results. Additionally, another very notable result in this chart is how several of the tested CNN encoders create models that do not perform much better or worse after training than before. Finally, the fact that SqueezeNet and DenseNet encoders perform better than the other models for a majority of the training and evaluation process opens the door for further research into these architectures and their applicability in document analysis. It is especially interesting to note that the SqueezeNet output in Figure 7 seems to be more block-based than the ResNet, which may help to explain how SqueezeNet can encapsulate blocks of text with success.

Chapter 5

Conclusion

Due to the nature of the historical documents, their segmentation is a different type of task than what most of the pre-trained state-of-the-art CNNs are built to do. While there is not an extensive amount of literature on document segmentation since the rise in popularity of CNNs, some research is beginning to push the envelope. The dhSegment paper and library provided an extremely interesting approach and toolkit that can be used for the segmentation of historical documents. They demonstrated that it is possible to use general computer vision models and standard post-processing techniques to outperform specifically dedicated systems that were built for document segmentation. The dhSegment authors did this by inserting a ResNet50 encoder into the U-Net architecture. We build on their discoveries by training and evaluating other general-use computer vision models as the encoders in the U-Net architecture.

With a small amount of data, we were able to train and evaluate several important CNNs such as other ResNets, SqueezeNet, DenseNet, and more. Our initial hypothesis that a ResNet50 encoder would perform the best on segmentation tasks rather than other generic building blocks is not supported by our experimental evidence. Due to the lack of diversity within the dataset and the small number of images analyzed, our results are not conclusive that any of these given models work better than ResNet. However, we demonstrate that other models like SqueezeNet and DenseNet perform better on our specialized dataset and should be considered targets for further research in the context of document segmentation. In summary, the most notable lessons learned are listed:

- Deep learning may outperform mature, classical methods of document analysis. The lack of ubiquity of a single or subset of classical methods used for image-based document analysis demonstrates that the existing tools are not sufficient to create

meaningful analyses. Deep learning is a relatively new technique and is already making significant progress in its ability to create successful document analysis [11].

- Generic deep learning techniques, rather than specialized document analysis systems, are very successful in historical document segmentation. With the advent of the dhSegment toolbox [10], it was shown that the combination of generic deep learning building blocks, ResNet and U-Net architectures, shows promising results in historical document analysis.

- Other unspecialized deep learning building blocks have the potential to improve on dhSegment's original architecture. We show that other generic CNN-based architectures, especially using SqueezeNet [17] and DenseNet [19], were able to outperform ResNet50 on our specific dataset. This result is not definitive due to the limited size and scope of the data used, but it is nonetheless an interesting outcome.

In the future, we would explore the performance of these architectures on larger datasets and incorporate them into a toolbox with the post-processing techniques mentioned by the dhSegment authors. Additionally, we would explore the intricacies of the training process concerning the variability seen in training the ResNet models and the lack thereof within the training of the DenseNet models and others. Finally, the use of the transformer-based multidimensional long-short-term-memory [25], another type of artificial intelligence model, is a promising technique for document analysis that we would also like to explore.

Chapter 6

Appendix A: Alternate Segmentation Tools and Methods

6.1 Classical Image Segmentation Techniques

Classical image segmentation techniques can be defined as processing techniques that are either algorithmically based or that are based on classical machine learning or statistical methods. Some examples of these techniques are edge-based, partial differential equation (PDE) based, threshold-based, region-based, clustering-based, or fuzzy theory-based [26].

Edge-detection can be defined as labeling pixels as either on-edge or off-edge, where an edge is a boundary between two adjacent parts of an image [27]. Edges are useful in a variety of diverse applications, so it is difficult to create a formal definition of an edge. Thus, there are many different ways to go about discovering edges based on the different requirements of an edge. One such algorithm that is considered the benchmark for edge detection is the Canny edge-detection algorithm [28]. The Canny algorithm uses Gaussian convolutions to reduce noise, finds edge strength and direction using the Sobel operator, and uses further statistical methods to clean up and connect edges. The Sobel operator is loosely defined as a convolutional operation on an image that uses two kernels to calculate the vertical and horizontal derivatives at a given pixel and determine if it is an edge [29]. While there are other edge-detection algorithms, we only discuss the Canny to provide an introduction into how edges may be found programmatically. After edges are discovered, they can then be used to separate the images into discrete categories to fulfill the goal of segmentation.

Threshold-based segmentation is one of the simpler kinds of image segmentation. Using either intensity or the greyscale value, one can create a binary partition of the image based on a selected threshold value. A threshold can be calculated with a histogram-based technique, where each pixel value is inputted into a histogram. The threshold value can

then be determined from an average of the histogram peaks, a minimum between peaks, or other attributes that may be found in the chart. This can be done globally, where one threshold is used for the entire image. This can also be done locally by splitting the image into sub-images and calculating a threshold value for each of these sub-images. This is because different parts of the images may have different amounts of lighting, so a single threshold may not effectively capture the correct labeling schema. Thus, under the assumption that smaller image regions are more likely to have a more approximately uniform light distribution, splitting up the image will help to determine more effective thresholds [30]. After each pixel is compared to its threshold, the resulting image is segmented into its predicted categories.

Partial differential equation-based segmentation methods are mostly used for quick calculations and image enhancement. They can be used to enhance the edges of an image and remove the noise [26]. They are based on topological theory, where the intensity or color values of pixels are converted into curves, and these curves are analyzed by topological tools to determine their homogeneity and other properties, allowing pixels to be categorized and thus segmented [31].

Region-based segmentation methods are those that separate the image into different regions based on similar characteristics. There are two types of region-based methods: growing and merging. Region growing methods begin with an initial seed in one location of the image. Then, all of the neighboring pixels that belong to the categorical group of the seed, based on a specific condition or thresholded value, are added to the seed, forming a region. Then, the process is repeated at the edges of the region until the region can no longer grow. This final region represents a segment that may be derived from the image. On the other hand, region merging methods start by partitioning the image many times and merging the individual partitions until they satisfy some conditions [32].

Clustering-based segmentation methods split the image into clusters of pixels that have similar characteristics. One example of a method that does this is the k-means method [33].

This method selects k initial clusters center and categorizes each pixel by which cluster center it is closest to. In this algorithm, distance can be defined by any characteristic of the pixel, such as color or intensity. Then, the cluster centers are all recalculated to be the average of all of the pixels in its cluster. This recategorization and reclustering continue until pixel labels no longer change. This method thus leaves the image in k segments, distinguished by a chosen characteristic, rather than by their region. Fuzzy theory-based methods build upon the previously mentioned cluster methods by realizing that decision-boundaries are not always clearly defined. This uses statistical methods to calculate the probability of pixels belonging to each class, rather than solely labeling them as a single class [34].

Many of the mentioned methods operate based on similar characteristics of individual pixels or groups of pixels to perform segmentation. While this may be effective for images that have a distinctly defined foreground and background or have high contrast between classes, this may not be suitable for operating on documents. Since documents are usually monochrome, locating different parts of the document will rely less on individual pixel values and more on the logical significance of certain markings.

6.2 Using CNNs for Segmentation (cont.)

Due to the success of CNNs for classification tasks, their applications to segmentation were explored [11]. The first significant development in this application comes from using Fully Convolutional Networks (FCNs) [35]. FCNs are similar to CNNs, except that they only contain convolutional layers. Using FCNs allowed models to create feature hierarchies that can utilize the underlying indicators of a feature to produce a per-pixel classification schema [11]. Building off of the original FCN for segmentation, the encoder-decoder architecture for segmentation was created.

The encoder-decoder architecture first uses an encoder to create low-resolution maps of fundamental features that exist in the image processing task [21]. It does this with multiple

sequences of convolutional layers, batch normalization, and ReLUs. After the encoding of underlying features is completed, the decoder maps the underlying features to labels for each pixel. This is called upsampling, which can either be done with pooling functions or deconvolutions.

6.3 Document Segmentation

Document segmentation is growing ever more important as data moves from the physical world to the virtual world. Volumes of paper documents will be uploaded digitally, and while humans can read them easily, machines have a much harder time doing so. The ability to automatically extract and process regions of documents can revolutionize both current workforce processes and the analysis of documents made before the era of digital data.

Before the recent advancements in artificial intelligence, classical algorithms had been used to try to tackle this problem of document segmentation. Documents had either been segmented with a top-down or a bottom-up approach [36]. A top-down approach begins with the whole document and divides it into logical subsections until no further divisions can be made, while a bottom-up approach begins with small elements of the document such as pixels. For more defined and clearer documents than the ones of the SSDA archive, some algorithms focus on a few geometric features such as straight lines or by using filtering techniques such as a run-length-smearing-algorithm [37]. Then, other algorithms may use classical machine learning techniques like clustering or function analysis. These algorithms discover features in the dataset but need to be trained to make sense of the data and how to interpret what features may mean.

Finally, recent advances in artificial intelligence and computer vision have allowed neural networks to attempt the segmentation problem on documents. They approach this through a pixel-by-pixel approach of classification into one of the segmented classes. Documents, especially historical ones, are extremely difficult to segment due to their degraded

nature and inconsistent layouts [38]. For a complicated visual task like this, it has been found that convolutional neural networks and U-Net style architectures have been very effective [10]. Another type of architecture that can be used for document segmentation is the multidimensional long-short-term-memory model (MDLSTM). A long-short-term memory model (LSTM) is extremely good at sequence processing [25], and one can treat an image as a sequence of pixels. An MDLSTM is similar to an LSTM except that it can process data that exists in multiple dimensions, such as an image. Thus, it can still reap the benefits of an LSTM's sequence processing but can do so in a way that treats an image as a two-dimensional sequence. This allows the model to notice patterns between pixels that may not be adjacent to each other. This model can help capture the dependencies of pixels on indicating features that may be far away on the page [39]. For example, a column marker can begin on the left side of the page and the column can extend to the right side, but one can only know that this column can still be classified as such by noticing the marker on the left. Experiments with MDLSTMs have shown that even with a relatively simple model, one can still derive success in segmentation tasks on old documents, even when they may include a lot of noise and degradation.

Chapter 7

Appendix B: Tuning Deep Learning Segmentation Models

7.1 Transfer Learning

While huge advances have been made in the architecture of deep learning models, they still require massive amounts of labeled data to train [40]. To train a computer vision neural network from scratch is to teach a computer to "see" from scratch, which is an extremely time-consuming and costly process. Thus, deep learning networks often employ transfer learning, which uses the domain knowledge of a model trained for one problem and applies it to another problem [41]. For example, a model that is trained to classify a dog may also be retrained to classify a cat much more efficiently than training a cat classifier from scratch. The original model's underlying domain knowledge, from low-level features such as lines and edges to high-level features like fur and limbs, can be efficiently reutilized to train the model to perform a new task.

When performing transfer learning, it can be efficient to only train the parameters of the later layers since the knowledge used in the earlier layers may not change. However, it has been shown that training can be accelerated by freezing and unfreezing layers of parameters, respectively disallowing and allowing them to change during model training [42]. A practice we employ is unfreezing the entire model's parameters and then progressively freezing the layers until almost all the layers besides the last few are frozen. This allows the model to alter some of its underlying layers, and thus its underlying knowledge, to be slightly more suitable and specialized for the given task at hand. Since earlier layers need to change less, they are progressively frozen to make training more efficient than changing every parameter.

7.2 Data Augmentation

As stated before, deep learning networks require a massive amount of data to train [40]. To overcome this issue when the size of the dataset is limited, data augmentation can be used to increase the amount of training data available. This can be done by either warping or oversampling [43]. Warping is altering existing data through a series of transformations including but not limited to rotations, mirroring, cropping, noise injection, and color space transformations. Oversampling is artificially creating new data points by either mixing existing ones or generating fake data by other deep learning methods. Any number of augmentations can be combined to vastly increase the size of the dataset, but one must consider the task at hand to create reasonable augmentations. For example, in a dog classifier, it makes sense to flip existing data horizontally but not vertically, as dogs standing upside down does not naturally exist and thus are not in the problem space. However, with geospatial or satellite data, the orientation usually does not matter so any sort of reflection or rotation is consistent with the range of possibilities in the data. Oversampling is especially interesting since creating fake images is counterintuitive to the goals of data augmentation. By mixing two images, the resulting image will look incorrect to a human observer, but through experimentation, it has been shown that it reduces classification errors, since mixed images can help the model isolate the important identifying features that persist between different original images [43].

7.3 Loss Functions

As important as neural network architecture is in attaining good model output, one must also use the right loss function to serve as a target to optimize in training [44]. There are several different types of loss functions that one can choose from in a segmentation problem. First, there are cross-entropy loss functions [45]. Cross-entropy loss functions measure the difference between the probability distributions of the real output and the predicted

output. Cross-entropy loss functions can be further altered to account for imbalances between class sizes. Additionally, cross-entropy loss functions can be altered to weigh "hard" examples more heavily.

A different family of loss functions is based on Dice loss. This type of loss function uses the Dice coefficient, which calculates how similar the prediction and truth images are, and returns a loss based on that value [24]. Another type of loss function is Tversky loss. Tversky loss is similar to Dice except that it especially weighs against false positives and false negatives. The Tversky loss can also be extended such that it weights harder examples more heavily. Finally, we have Sensitivity Specificity Loss [46]. This type of loss is calculated as a proportion of positives predicted to all real positives and a proportion of negatives predicted to all real negatives. This allows for one to account for class imbalances as well.

7.4 Evaluation Metrics

There are five main evaluation metrics for an image segmentation problem [11]. First is Pixel Accuracy, which only consists of the ratio of properly labeled pixels to all pixels. Next is Mean Pixel Accuracy, which averages the Pixel Accuracy of each class. Then, there is the mean Intersection over Union (mIoU). mIoU is the most common metric used in segmentation and is defined as the per class ratio of true positives to the sum of true positives, false negatives, and false positives, which is then averaged between classes. Finally, we have the Frequency Weighted Intersection over Union, which is similar to mIoU but allows for considerations to be made based on the relative frequency of appearance for each class.

BIBLIOGRAPHY

[1] Yanhui Guo and Amira S Ashour. 11 - neutrosophic sets in dermoscopic medical image segmentation. In Yanhui Guo and Amira S Ashour, editors, *Neutrosophic Set in Medical Image Analysis*, pages 229–243. Academic Press, January 2019.

[2] S Belongie, C Carson, H Greenspan, and J Malik. Color- and texture-based image segmentation using EM and its application to content-based image retrieval. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, pages 675–682, January 1998.

[3] D L Pham, C Xu, and J L Prince. Current methods in medical image segmentation. *Annu. Rev. Biomed. Eng.*, 2:315–337, 2000.

[4] Spyros Gidaris and Nikos Komodakis. Object detection via a multi-region and semantic Segmentation-Aware CNN model, 2015.

[5] M Nilsson, A H Herlin, H Ardö, O Guzhva, K Åström, and C Bergsten. Development of automatic surveillance of animal behaviour and welfare using image analysis and machine learned segmentation technique. *Animal*, 9(11):1859–1865, November 2015.

[6] Zhaofeng He, Tieniu Tan, Zhenan Sun, and Xianchao Qiu. Toward accurate and fast iris segmentation for iris biometrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(9):1670–1684, September 2009.

[7] A S Kavitha, P Shivakumara, G H Kumar, and Tong Lu. Text segmentation in degraded historical document images. *Egyptian Informatics Journal*, 17(2):189–197, July 2016.

[8] Slave societies digital archive. https://slavesocieties.org/home. Accessed: 2021-2-10.

[9] Libro 7 de bautismos de pardos y morenos, 1872-1892, June 2020.

[10] S Ares Oliveira, B Seguin, and F Kaplan. dhsegment: A generic Deep-Learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12, August 2018.

[11] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. April 2017.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.*, 25:1097–1105, 2012.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for Large-Scale image recognition. September 2014.

[14] J Wang, J Lin, and Z Wang. Efficient convolution architectures for convolutional neural network. In *2016 8th International Conference on Wireless Communications Signal Processing (WCSP)*, pages 1–5, October 2016.

[15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. September 2014.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. December 2015.

[17] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size. February 2016.

[18] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep

neural networks with pruning, trained quantization and huffman coding. October 2015.

[19] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. August 2016.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. May 2015.

[21] Saeid Asgari Taghanaki, Kumar Abhishek, Joseph Paul Cohen, Julien Cohen-Adad, and Ghassan Hamarneh. Deep semantic segmentation of natural and medical images: a review, 2021.

[22] Jeremy Howard and Sylvain Gugger. Fastai: A layered API for deep learning. *Information*, 11(2):108, February 2020.

[23] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, MM '10, pages 1485–1488, New York, NY, USA, October 2010. Association for Computing Machinery.

[24] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 240–248. Springer International Publishing, 2017.

[25] Marijn F Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel Multi-Dimensional LSTM, with application to fast biomedical volumetric image segmentation. June 2015.

[26] Dilpreet Kaur and Yadwinder Kaur. Various image segmentation techniques: a review. *International Journal of Computer Science and Mobile Computing*, 3(5):809–814, 2014.

[27] H G Kaganami and Z Beiji. Region-Based segmentation versus edge detection. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1217–1221, September 2009.

[28] J F Canny. A computation approach to edge detectors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8:34–43, 1986.

[29] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[30] K Bhargavi and S Jyothi. A survey on threshold based segmentation technique in image processing. *International Journal of Innovative Research and Development*, 3(12):234–239, 2014.

[31] Xin Jiang, Renjie Zhang, and Shengdong Nie. Image segmentation based on level set method. *Phys. Procedia*, 33:840–845, January 2012.

[32] S Angelina., L P Suresh, and S H K Veni. Image segmentation based on genetic algorithm for region growth and region merging. In *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, pages 970–974, March 2012.

[33] Siddheswar Ray and Rose H Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*, pages 137–143, 1999.

[34] Daniel Gómez, Javier Yáñez, Carely Guada, J Tinguaro Rodríguez, Javier Montero, and Edwin Zarrazola. Fuzzy image segmentation based upon hierarchical clustering. *Knowledge-Based Systems*, 87:26–37, October 2015.

[35] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, April 2017.

[36] Sébastien Eskenazi, Petra Gomez-Krämer, and Jean-Marc Ogier. A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognit.*, 64:1–14, April 2017.

[37] N Priyadharshini and M S Vijaya. Genetic programming for document segmentation and region classification using discipulus. March 2013.

[38] Yue Xu, Wenhao He, Fei Yin, and Cheng-Lin Liu. Page segmentation for historical handwritten documents using fully convolutional networks. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 541–546, 2017.

[39] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555, 2015.

[40] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[41] Karl Weiss, Taghi M Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):9, May 2016.

[42] Piotr Chudzik, Somshubra Majumdar, Francesco Caliva, Bashir Al-Diri, and Andrew Hunter. Microaneurysm detection using deep learning and interleaved freezing. In *Medical Imaging 2018: Image Processing*, volume 10574, page 105741I. International Society for Optics and Photonics, March 2018.

[43] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, July 2019.

[44] Shruti Jadon. A survey of loss functions for semantic segmentation. June 2020.

[45] Ma Yi-de, Liu Qing, and Qian Zhi-bai. Automated image segmentation using improved PCNN model based on cross-entropy. In *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, pages 743–746, October 2004.

[46] S R Hashemi, S S Mohseni Salehi, D Erdogmus, S P Prabhu, S K Warfield, and A Gholipour. Asymmetric loss functions and deep Densely-Connected networks for Highly-Imbalanced medical image segmentation: Application to multiple sclerosis lesion detection. *IEEE Access*, 7:1721–1735, 2019.