DS 4400 Final Project: Analysis of Machine Learning Algorithms for Image Segmentation

Fall 2020

By Noah Lee and Jason Chen

Problem Statement

Image segmentation is both a common and complex technique in computer vision research. The task involves identifying the location of one or more target objects in an image, and separating them from the background of the image. For our project specifically, we attempt to segment a single entity from the background by predicting a pixel-wise mask over the image. Image segmentation techniques are commonly used in robotics, autonomous vehicles, medical image analysis, and geospatial image analysis. Our project aims to compare three different approaches to image segmentation: a logistic regression model, a feed forward neural network, and a fully convolutional neural network.

References

Dataset: https://www.robots.ox.ac.uk/~vgg/data/pets/

U-Net: https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/

ROC Curves: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

PyTorch: https://pytorch.org/docs/stable/index.html

Feed Forward Neural Network:
https://colab.research.google.com/drive/10lNeB3SqFBQLDHdf-2h3pnQKC-cs1nY2?usp=sharing

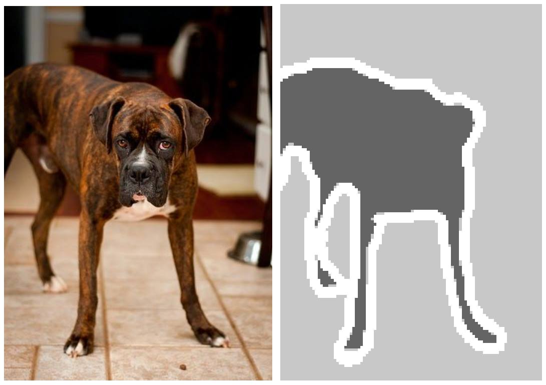
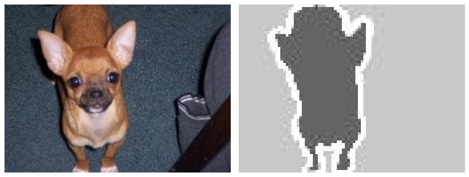
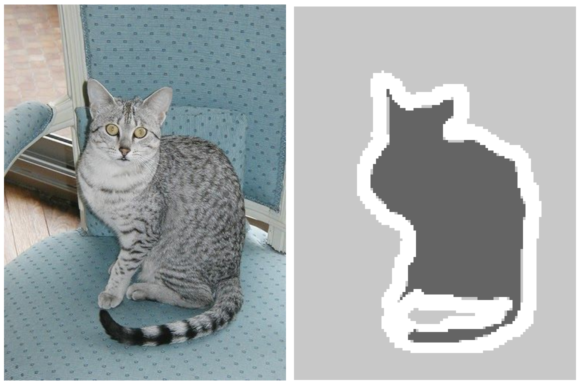
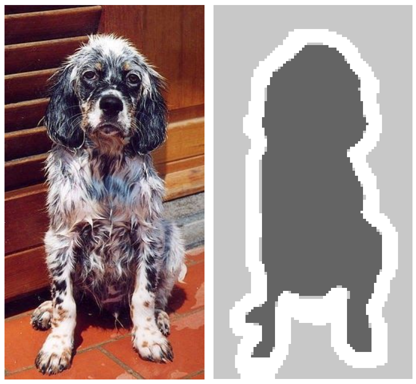
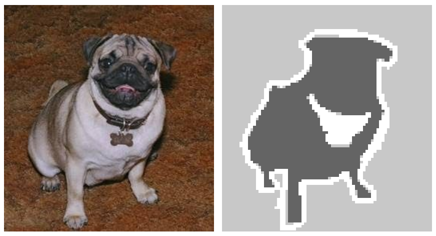
Logistic Model and U-Net:
https://colab.research.google.com/drive/19zM24VZMsD7dVGFfvrTW2ko7yqrhLqIM?usp=sharing

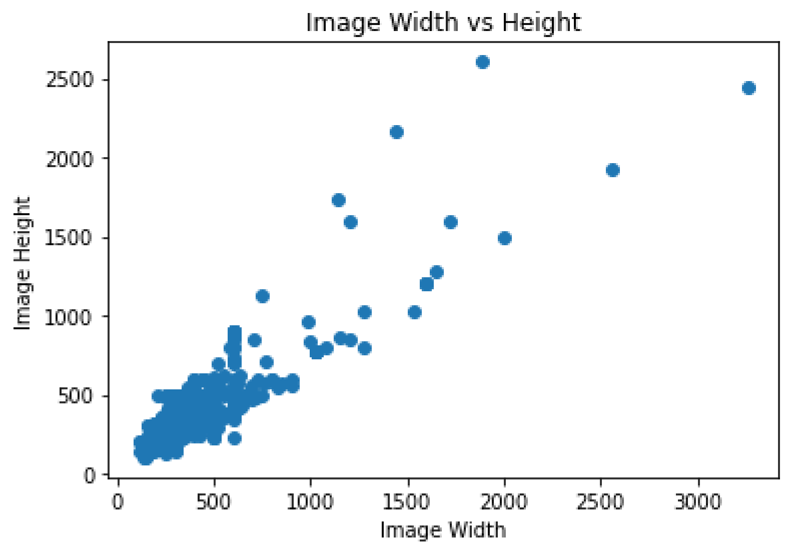
Presentation Slides:
https://docs.google.com/presentation/d/1bj9wxS4nGoPmToAmJt1Bw4Rl0jCLVtcyRWSdTlx-_-M/edit?usp=sharing

Dataset

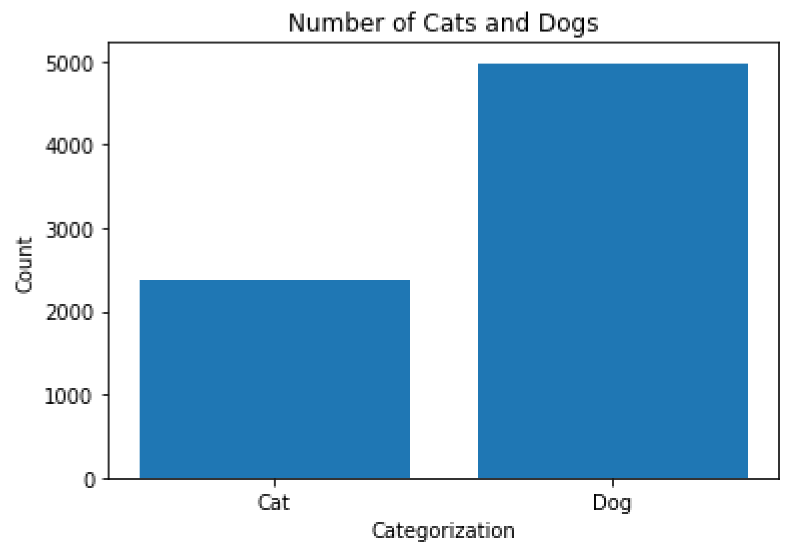
The dataset we decided to use was the The Oxford-IIIT Pet Dataset, which is available at https://www.robots.ox.ac.uk/~vgg/data/pets/. This dataset consists of 7349 pictures of cats and dogs. The image size varies between different images. Accompanying each image is a ground truth annotation for several different machine learning tasks, including breed, head bounding boxes, and pixel level trimap segmentation.

Several sample images from the dataset appear on the following page, as well as their trimap segmentation masks. The images that appear below have been resized to be smaller, while maintaining their relative dimensions.

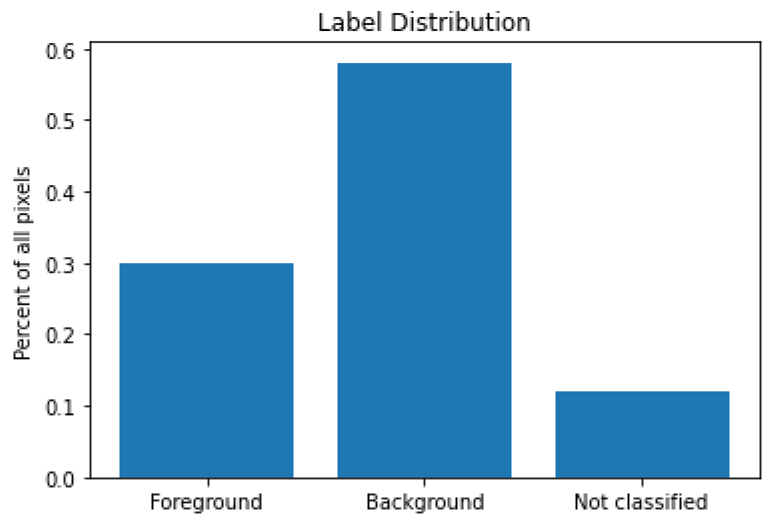








A brief glance at the images (and also some exploratory analysis) shows that the images vary in size, with a majority of images having a size between 100x100 and 1000x1000.



In addition, for a categorization task, the number of cat and dog images is slightly skewed to favor the dogs, there are approximately twice as many dog images as cat images.



Finally, each trimap image has three pixel classifications: “foreground”, “background”, and “not classified”. The label distribution of the trimaps over all the training images is heavily skewed in favor of background pixels. Over 50% of all pixels in the trimaps consist of background pixels, 30% are classified as foreground, and only about 10% fall under the “not classified” label.

After viewing some of the sample images and their trimaps, it appears that the “not classified” label is actually used for the outlines of pets and for a pet’s accessories, such as a collar.

For this task, we are only looking at the original images and the corresponding trimaps. Each trimap mask is sized identically to the image itself, and consists of pixel values of 1/255, 2/255 and 3/255. When these pixel values are multiplied by 255, these values are 1, 2, and 3. The corresponding mappings of these trimaps is as follows:

1: Foreground

2: Background

3: Not classified

Features

The features we used consisted of each individual pixel's value. Each pixel value consists of 3 channels with RGB values between 0 and 1. These values can be multiplied by 255 to get integer RGB values between 0 and 255 inclusive.

The targets for this task are the trimap segmentation masks. For each pixel in the original image, we are attempting to predict the corresponding trimap value: 1, 2, or 3.

Data Preprocessing

As all the images are different sizes, we resized all the images to square images. The exact dimensions are dependent on which model was being used.

To augment our dataset, we chose to randomly flip our images horizontally and vertically, as well as augmenting the brightness and contrast of the images randomly. Finally, the images were normalized based on the means and standard deviation of the pixel values from the untransformed images. Whenever an image is pulled from the training set, the image will be randomly transformed according to these criteria.

Models

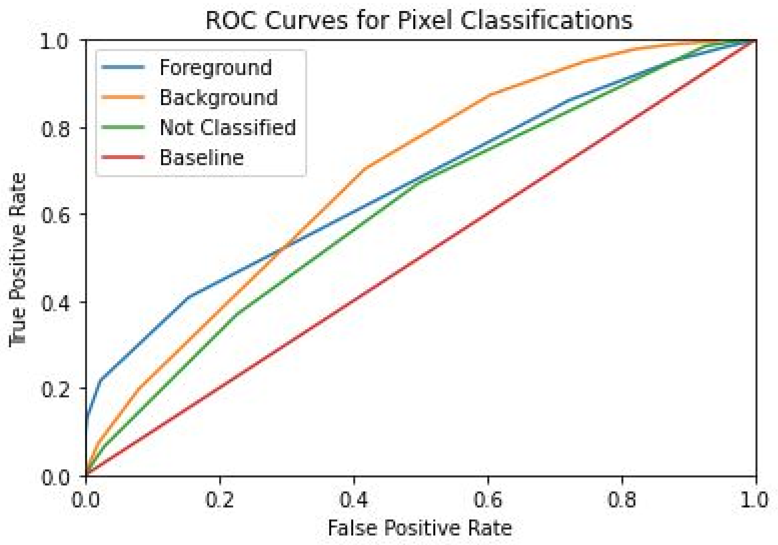
Model 1: Neural Network with a Single Convolutional Layer

This neural network was designed to emulate a single logistic regression classifier passed over the entire image to output a 256x256 trimap segmentation mask. Each output pixel was predicted individually using the corresponding input pixel and the surrounding input pixels as features.

As this model only consists of one convolutional layer and a sigmoid activation function, it is only able to learn extremely simple regional features. It relies solely on regional features of each image, and is unable to learn the features of the image as a whole.

This model takes in a 256x256 image with three color channels representing red, green, and blue, for a total of 256x256x3 = 196608 input features and outputs a 256x256 segmentation mask to match the size of the image.

The ROC curves and AUC metrics for this model are shown below:



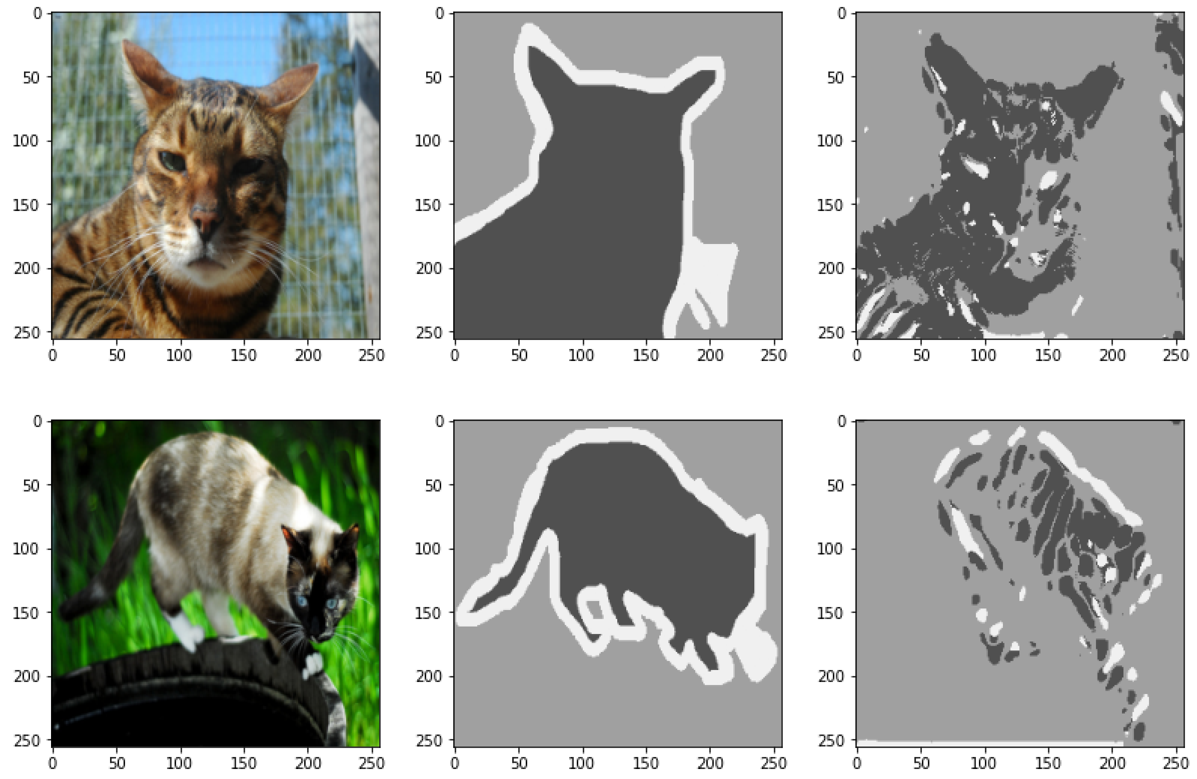
```
AUC Foreground: 0.6653770559410274
AUC Background: 0.6322525727472978
AUC Not Classified: 0.6131485505350576
```

The precision, accuracy, recall, and F1 score for each pixel classification averaged across all images appears below:

	Precision	Recall	Accuracy	F1
Foreground	0.734	0.800	0.656	0.764
Background	0.487	0.884	0.561	0.626
Not Classified	0.884	0.9997	0.884	0.938

The most surprising metric here is the precision of background classifications, which is only slightly above 50%. This indicates that this model is misclassifying a lot of pixels as background when they should not be background.

Some sample outputs from this single convolutional model appear below. The input image appears in the right column, the ground truth segmentation mask appears in the middle column, and the output from the model is in the leftmost column. This model is able to capture the rough shape of the pet, but is unable to accurately fill in the pet itself, nor construct the outlines around the pet.



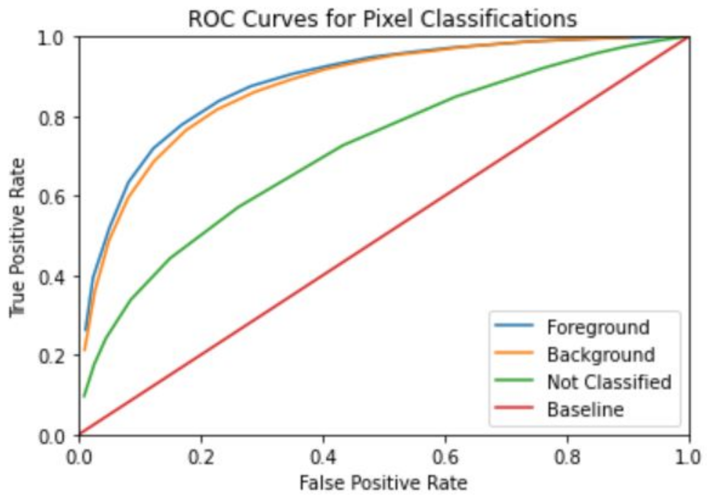
Model 2: Feed Forward Neural Network (Multilayer Perceptron)

This model is a fully connected feed forward neural network with two hidden layers. The input to the model comes as 64 by 64 pixel RGB images, and each hidden layer has 48 neurons. The output of the model is the same shape as the input, but each of the three channels represents the pixel class rather than RGB. The reason for the smaller input size was to keep the model within a 15 gigabyte RAM limit as required for Google Colab. The size of the model was quite large due to the number of weights between layers.

A major difference between this model and the prior model is that all of the output pixels are computed simultaneously rather than individually. This means that each pixel classification can receive information from the entire image instead of the immediate surrounding pixels.

The model was trained for 25 epochs on mini batches of 64 images with a learning rate of 1e-4. During training, the model tried to minimize the loss according to the cross entropy classification score of the pixels in the output. A final testing accuracy of 68.74% was achieved.

Below are ROC curves for the three pixel classes. The ROC curves were computed by performing one vs. all classification, where each class was compared to a combination of the other two classes.

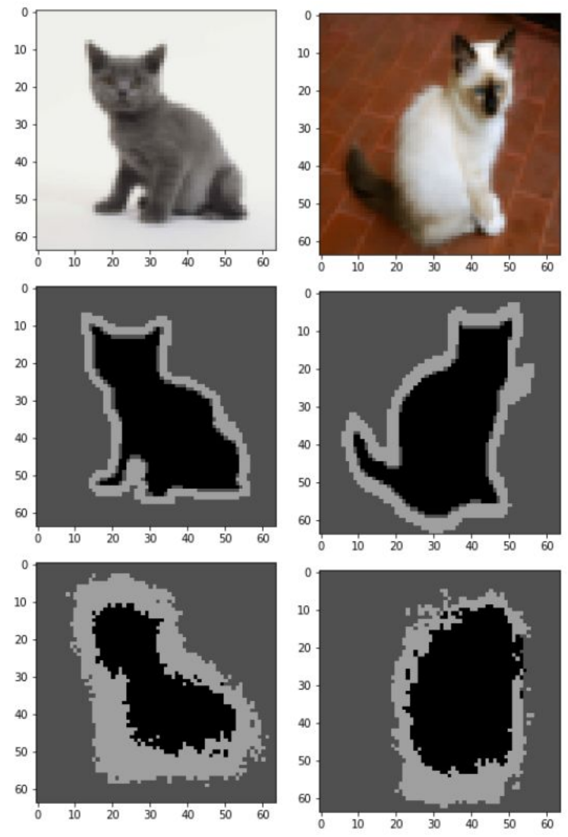


```
AUC Foreground: 0.8344886575068686
AUC Background: 0.7758952583613531
AUC Not Classified: 0.7143721767945775
```

It is evident that the model is best able to predict foreground pixels, then background pixels, and finally unclassified pixels. This is likely a function of the class weighting that we did to offset the class imbalance in our dataset.

	Precision	Recall	Accuracy	F1
Foreground	0.845	0.932	0.829	0.886
Background	0.597	0.919	0.723	0.723
Not Classified	0.905	0.956	0.871	0.930

We believe that the reason for an improvement in performance over the first model is due to the fact that pixel information is no longer completely localised; now, values from across the entire image are able to contribute to the classification. This prevents the model from only learning edges of the object.



In the images, we see that the model is able to identify the main body of the object in the image and create a rough outline. However, it is still limited because it is unable to classify correctly the appendages like the legs and tails of the pets.

Model 3: Fully Convolutional U-Net

After doing some research into previous neural networks designed for image segmentation, we found a network architecture called a "U-Net". This architecture consists of a "downsizing" branch and an "upsizing" branch. The downsizing branch consists of four blocks of convolutions and max pooling, each block consists of two convolutional layers followed by a max pooling layer. The upsizing branch consists of four blocks of convolutions and "up-convolutions", each block consists of an up-convolution followed by two convolutional layers. Additionally, the output of each second convolutional layer in the downsizing block is saved and appended forward to the corresponding upsizing branch, which eventually allows the model to output a 256x256 segmentation map.

Due to only having convolutional layers, this model is able to have a much larger number of layers than a traditional feed forward neural network, which allows the network to learn more regional information from the image. In addition, these convolutional layers also act as a sort of feature regionalization, which allows the model to learn regional features and group together pixels of the same final categorization. In addition, to provide the network more data to work with (to avoid reconstructing a 256x256 image from a 16x16 image, the outputs of the convolutional blocks in the downsizing branches are passed forward to provide more data for the upsizing branch to reconstruct the image from.

This model takes in a 256x256 image with three color channels representing red, green, and blue, for a total of 256x256x3 = 196608 input features. This model outputs a 256x256 segmentation mask to match the size of the input image.

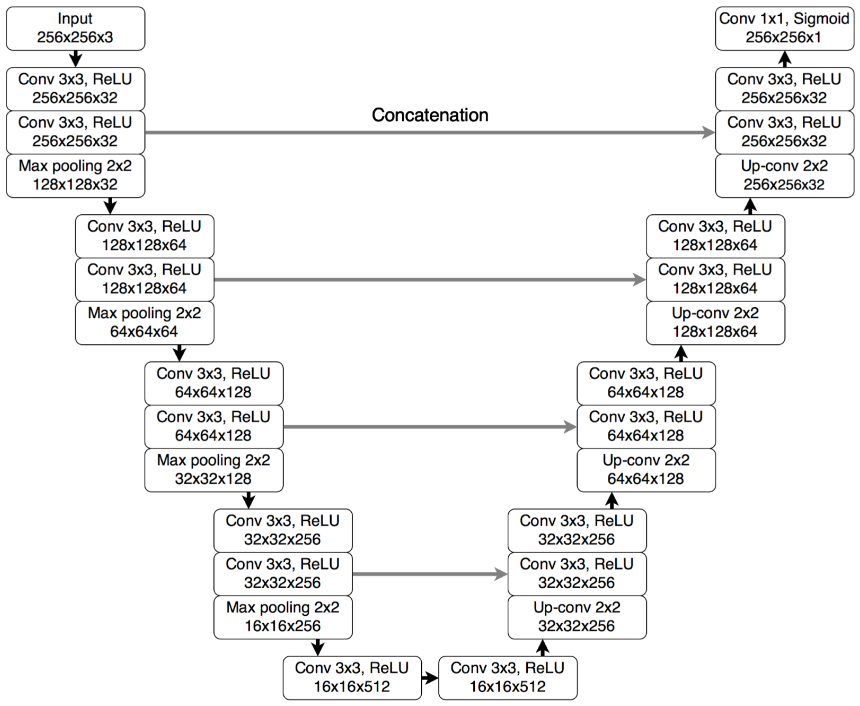


Image from: https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/

On the fully convolutional U-Net, we performed 5-fold cross validation. The training and testing loss and accuracy are as follows after training for 25 epochs:

Fold 1: Train Loss: 0.326 | Train Acc: 88.88% | Val. Loss: 0.334 | Val. Acc: 88.54%

Fold 2: Train Loss: 0.356 | Train Acc: 87.88% | Val. Loss: 0.355 | Val. Acc: 88.15%

Fold 3: Train Loss: 0.331 | Train Acc: 88.66% | Val. Loss: 0.380 | Val. Acc: 88.48%

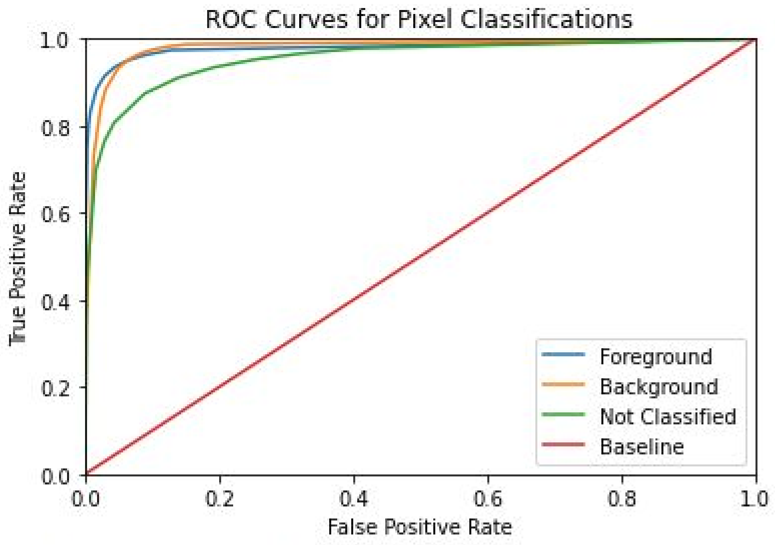
Fold 4: Train Loss: 0.334 | Train Acc: 88.63% | Val. Loss: 0.333 | Val. Acc: 88.80%

Fold 5: Train Loss: 0.333 | Train Acc: 88.71% | Val. Loss: 0.333 | Val. Acc: 88.90%

In addition, we trained another model on an increased number of epochs (approximately 50), and achieved a maximum validation accuracy of 89.91%. The metrics for this model are as follow (pixelwise accuracy):

Train Loss: 0.279 | Train Acc: 90.57% | Val. Loss: 0.314 | Val. Acc: 89.91%

Using the final model, the ROC curves and the AUC's are as follow:



```
AUC Foreground: 0.9793468679426613
AUC Background: 0.9802448336580172
AUC Not Classified: 0.9539136039599037
```

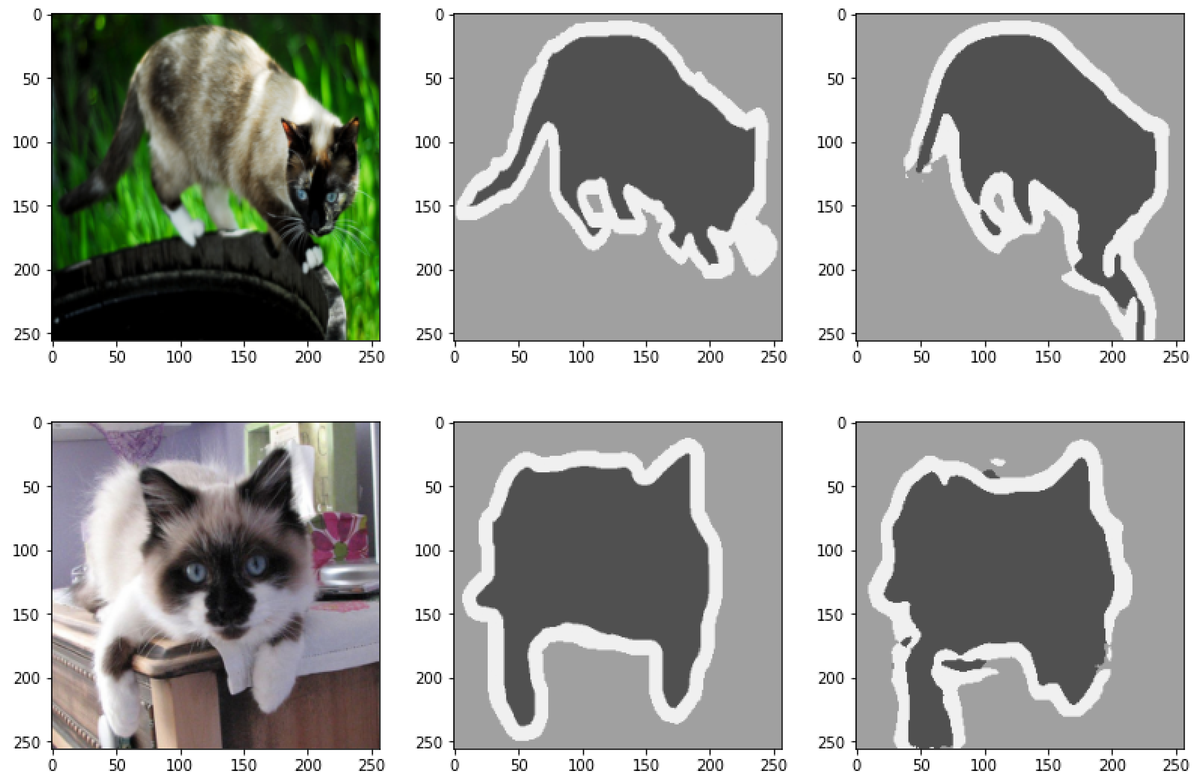
The precision, accuracy, recall, and F1 score for each pixel classification averaged across all images appears below:

	Precision	Recall	Accuracy	F1
Foreground	0.963	0.944	0.935	0.953
Background	0.875	0.956	0.925	0.913
Not Classified	0.957	0.957	0.924	0.957

According to the ROC curves, this U-Net seems to perform slightly better on foreground and background pixels, while performing slightly worse on unclassified pixels. This may be due in part to the percentage of unclassified pixels being smaller than the percentage of foreground and background pixels. According to the other evaluation metrics, it appears that classifying background pixels suffers from poor precision, indicating that there are a lot of false positives for that class, which also caused the F1 score to drop.

Initial iterations of this U-Net had less convolutional layers, and were lacking the "pass forward" portion of the network, meaning the network had to create a 256x256x3 segmentation map from a downscaled 16x16x512 image. These previous versions of the model did not perform as well as the final version of this U-Net.

Some sample outputs of this model appear below. The left column is the input image, the middle column is the ground truth segmentation mask, and the right column is the output of the U-Net model.



Discussion

One of the first difficulties that we encountered upon starting this project was learning the PyTorch library and other image-related libraries. In regards to this, it took a little bit of time to figure out the libraries, read in images, and display them back to the user in Python. Another hurdle we faced was working with the data from the images, which was formatted as a PyTorch Tensor, including transforming the data and converting it into a format usable by the models we created. The final obstacle regarding the libraries themselves was figuring out how to create models, get outputs using a forward pass through the model, and finally backpropagating to train the model.

Another difficulty we encountered were problems with the size of models. Since the inputs to our models are color images of a decent size, there are a total of 256x256x3 = 196608 input features into the models. Additionally, the output of our model must be an image of equal size (256x256), and we must be able to build a trimap from the output of the model. This means the output from the model would have to have three channels as well, representing the probabilities that a given pixel belongs to any of the three classes. This was a very big issue in the fully connected feed-forward neural network, as a single fully connected layer between 196608 input features and a hidden layer of 10000 output features is already 2 billion weights. However, if we use smaller fully connected layers in the middle of the network, then we are reducing the number of features we are working with, and then increasing them again to reconstruct the full size trimap. This led us to two different solutions: reducing the number of input and output features (such as in the fully connected feed forward network), or not using fully connected layers at all (such as in the fully convolutional U-Net).

Research Materials

In order to build more robust classifiers after seeing the performance of the baseline model, we did some research on existing neural networks for image segmentation. From this research, we found the Fully Convolutional U-Net, the implementation we found was designed to be used for biomedical image segmentation to detect abnormalities in a brain MRI. (https://pytorch.org/hub/mateuszbuda_brain-segmentation-pytorch_unet/) Using this U-Net as a guide, we designed our fully convolutional U-Net after this example, and found considerable success with this model architecture.

Conclusion

When training up the U-Net, we went through multiple different model architectures before selecting the one detailed in this report. Our first iterations of U-Nets were significantly smaller than the current version, and struggled to fit the training dataset. Even after expanding the U-Net to encompass more layers, the model was still struggling to fit the training dataset. We then found it quite surprising that after implementing the “forward append” to the model, it was able to perform significantly better than without the forward append. (Approximately a 10% increase in accuracy).

One thing we were working on implementing was a mean Intersection over Union loss metric for our models. The aim was to be able to train the models using this alternative loss function and compare the results to the Cross Entropy loss function. However, we ran into several challenges in the implementation - foremost being that the basic form of intersection over union is non-differentiable, and so we are unable to use the metric for training models. However, if we had some more time I believe that it would have been possible to implement a differentiable version of the IOU metric.