Jessica Bhattacharyya
Michael Einhorn
Rohan Patwardhan
Xiaodong Gao

Final Report

Background

An intelligent agent can easily be compared to an animal, as both can discern valuable information and ignore background noise in order to complete their tasks. A predator can track prey over great distances; for instance, a hawk can target a mouse while ignoring the vegetation around it. Given this premise, a neural network that can effectively distinguish meaningful life forms from their surroundings is critical for creating an agent capable of operating independently in the wilderness, as it would have to determine whether there is a creature in its surroundings that may be a threat. This goal can be realized through numerous technologies such as FCNN and U-Net structures.

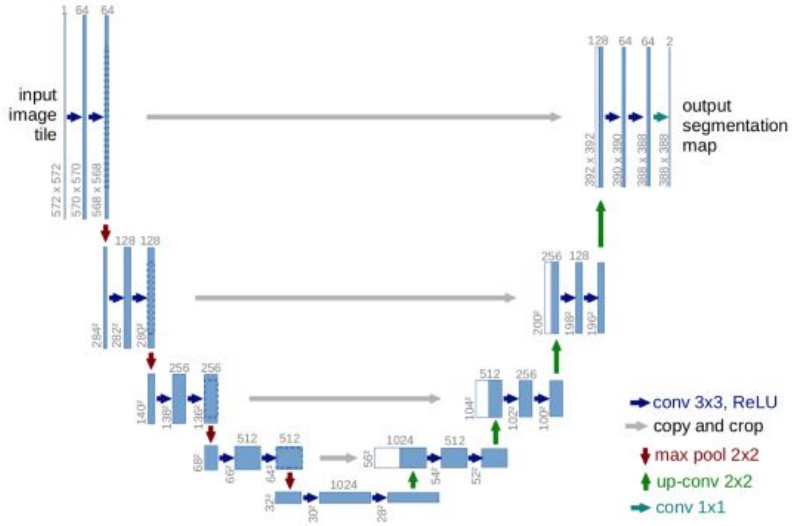


Figure 1. Visualization of the structure of U-Net (Zhang).

Problem Description

The goal of this project is to create a model capable of distinguishing the subject of an image, particularly an animal, from its background. This machine learning model will take an image as input, process the image, and return an image segmented by subject and background. It will be unsupervised, reducing the labor required for data processing.

Data Collection & Processing

Choosing a dataset

This project has a heavy focus on semantic segmentation, so supervised learning techniques will be used to train a model capable of distinguishing animals from background noise. For this, two kinds of data are needed: an unprocessed wildlife photograph and a parallel image that has already been segmented in some way. Having this counterpart allows for cross-validation of the algorithm's results.

Given these criteria, the optimal dataset would contain the following attributes:

A. contains both original and segmented images
B. original and segmented images can be matched with filename, folder structure or both
C. contains a rich set of animals with different features and relatively random background so that the resulting algorithm can be flexible and generalizable.

After an extensive search through datasets available on Kaggle, it was determined that the Oxford's Pet III dataset best matched the project's needs.

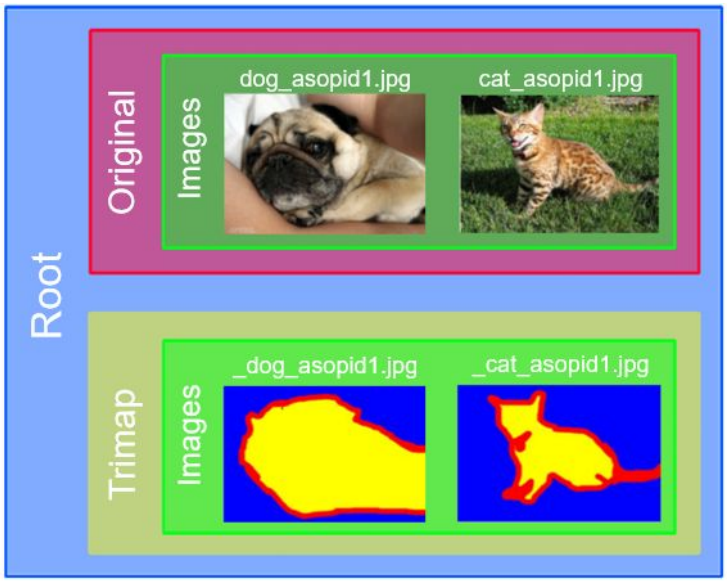


Figure 2. A graphical representation of the approximated structure of Oxford's Pet III dataset.

The structure of this data set readily lends itself to the project's objective, as an unprocessed image can be easily paired with its respective trimap. The trimap can be considered a segmentation ground truth, since it separates animals from background noise.

Data Preprocessing

Although this dataset is composed of two optimal image sets, a potential issue arises in that the images vary in height and width. OpenCV was used to resolve this, as it processes images and converts them into a uniform size. Given this, the model can be agnostic to image attributes as it can treat them as identical inputs.

Prior to the iterative conversion of images to OpenCV arrays, the paths of images had to be gathered. A simple form of regular expression was used to load all files of a certain extension. Also, since OpenCV images have flipped channel ordering (BGR instead of RGB), each image was manually flipped upon import.

There are various methods available for standardizing image sizes in a way that retains features and qualities (Wright). Many involve interpolation. Some use a nearest-neighbor method, which is fast and efficient, but “loses substantial information when resizing” (MusiGenesis, sthlm58). Finally, linear interpolation can be considered an intermediate choice with an average performance.

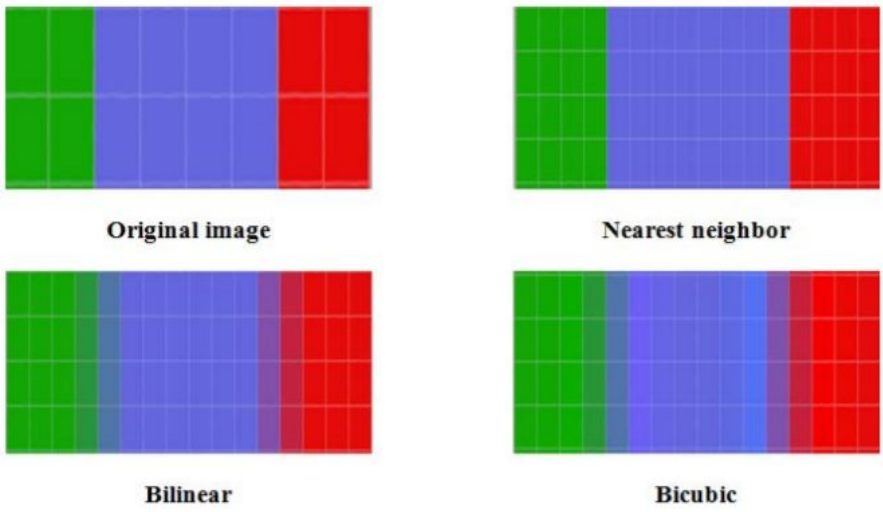


Figure 3. Examples of linear, bicubic, and nearest-neighbor interpolation (Savagave)

These three methods were tested extensively, and eventually the bicubic method was elected. It proved to retain the most information, as was critical for the model.

In addition, random noise was inserted into some images through the use of a Gaussian blur. This was done so that the model could be made more robust against cases with unclear edges, unlike other optimal cases.

Figure 4. An image with an applied Gaussian blur of 10x10 pixels.

Grayscale images were also introduced into the dataset, but yielded poor performance. A possible rationale is that segmentation requires color images and three channels of data to distinguish between background and animal. Technologies such as Deep Dream, capable of visualizing the algorithm itself, could be used to gain further insight on why this issue occurs.

Finally, we resized the trimaps to become three dimensional, with a shape of (256, 256, 3). Originally, the trimaps were two dimensional and while training the model, the images themselves became two dimensional as well. Because of this, loss did not vary and accuracy stagnated at about 33% across all epochs. A possible reason for this is that converting three-dimensional images into two-dimensional images resulted in a loss of image quality. Removing the third dimension of the array damaged the network's ability to match trimaps, as shown by the relatively low accuracy. To combat this, two strategies were tested, first trimaps were resized to three-dimensional images by repeating the same array on itself three times, and second trimaps where each color channel represented one of the three segments. The second option resulted in a higher accuracy on the validation set, and was chosen.

Methods

After cleaning up the data, a fully convolutional neural network model was created based on U-Net architecture (Figure 1). The model took in an image as input and passed it through two two-dimensional convolutional layers of kernel size (3, 3), followed by a dropout layer and a two-dimensional max pooling layer with a pool size of (2, 2) for a total of four layers. The image shape at this point was 16x16x128. The image was then passed through two more two-dimensional convolutional layers and a dropout layer, still with a kernel size of (3, 3). As before this resulted in an array of size (16, 16, 256). The image was then upsized to its original dimensions by passing it through a two-dimensional transposed convolutional layer. The resulting image was then concatenated with the "corresponding image from the downsizing path of the model in order to combine information and get a more precise prediction" (Zhang). This new concatenated image was then passed through two more two-dimensional convolutional

layers and a dropout layer, repeated four times. All convolutional layers up to this point used relu activation with the padding type “same”. With the upsizing portion of the model complete, it had produced an array of size (256, 256, 16). In order to generate a final output, an RGB image array of size (256, 256, 3), the array was passed through a two-dimensional convolutional layer with three filters of size (1,1), which effectively scales the data by a constant factor for each color channel. Using softmax activation, this generated a final output layer sized (256, 256, 3). Upon creation of the model base, the output layer and original image size were passed into the Keras model. The model was then compiled with the adam optimizer at a learning rate of 0.001 and rated using mean squared error.

With a developed model, the original image and trimap datasets were split into a testing and training set. Sklearn’s train_test_split method was used to shuffle the dataset (in order to train across all classes rather than just one) and separate the data in a ratio of 3:1. The Keras model_checkpoint callback was also used to save the most accurate model. Initially, the model had a very high accuracy of 100% in less than 10 epochs, while appearing to output incorrect results. Essentially, the output trimaps consisted solely of arrays of ones, which when compared to the training trimap produced almost perfect results. The reason for this is still uncertain, but it went away after other changes. Additionally, as the trimap pixels were either a 1,2, or 3, a sigmoid activation function was unable to output a correct answer. This was alleviated having each color channel represent a different segment of the trimap (ex (1,0,0), (0,1,0), (0,0,1)), and in the final two-dimensional convolutional layer, the sigmoid was switched with the softmax activation.

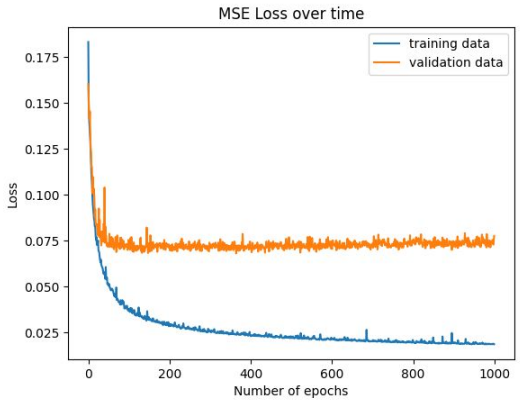
Results & Discussion

When preprocessing images, a few corrupt images within the dataset caused several errors. Even so, the package libpng consistently threw an error regarding the input format. This was resolved by adding various “try-catch” functions to the model. It was found that one JPG image in the vast dataset caused major issues while preprocessing the model. After manually scrolling through over 3700 image names provided by a debugging program, the corrupt images were found to be chihuahua_121.jpg and beagle_116.jpg. With this discovery, these two images and their corresponding trimaps were removed from the image and trimap datasets.

There were two working solutions for how to represent a trimap in the model. One was to have the trimap pixels be (1,1,1), (2,2,2), or (3,3,3) with an activation of relu on output of model. The other was to have the trimap pixels be (1,0,0), (0,1,0), or (0,0,1) with an activation of softmax on the output of the model. After 20 epochs of training, the first model had a validation set accuracy of 51% while the second model had a validation set accuracy of 82%, so the second way of formatting trimaps and a softmax activation of the output in the model was chosen.

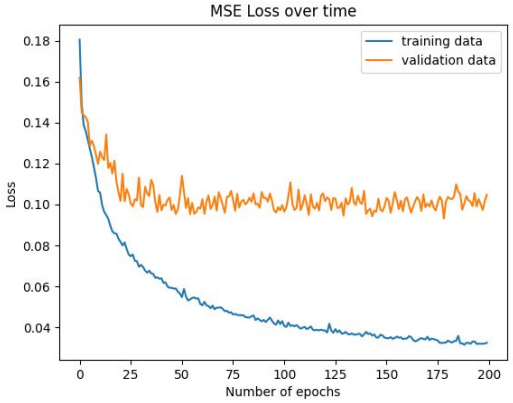
The network was trained with and without Gaussian blur on the images.

Training Progress on Unblurred Images over 1000 Epochs



Validation Accuracy 85.2%

Training Progress on Blurred Images over 200 Epochs



Validation Accuracy 78.9%

Figure 5. Two graphs comparing training loss from running original images over 1000 epochs and running images that have been modified with a Gaussian blur over 200 epochs.

The validation set was used to determine when training was done, and whether the model was with blur without blur. Total training epochs were reduced when training with Gausian blur, because it was seen in the other model that most of the 1000 epochs were not required for the validation set to reach a minimum loss.

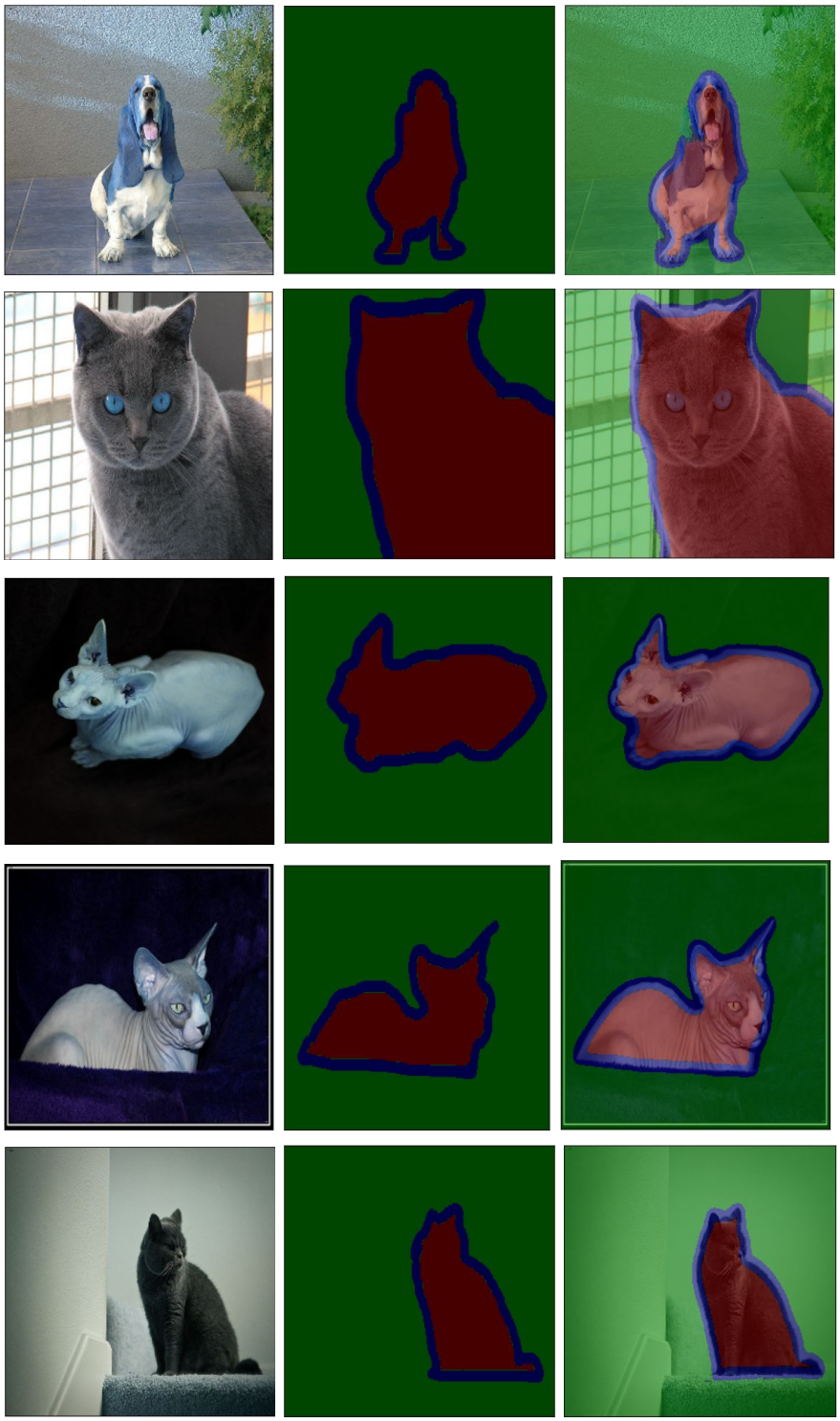
Without a blur the validation set loss stopped decreasing after 100 epochs. At this point, the validation set had an accuracy of 85.2%.

With a Gaussian blur, the validation set loss stopped decreasing after 50 epochs, at this point the validation set had an accuracy of 78.9%.

The model trained for 100 epochs on unaltered images was chosen.

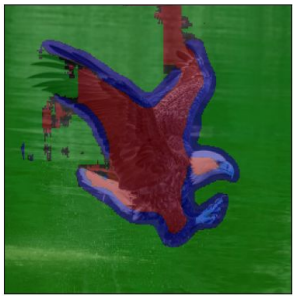
These were 5 images from the validation set that the model performed best on.

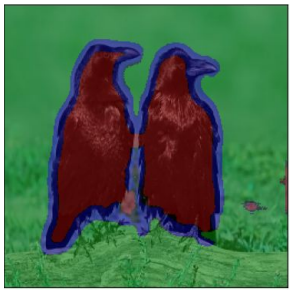
Image | True Segment | Prediction Overlay

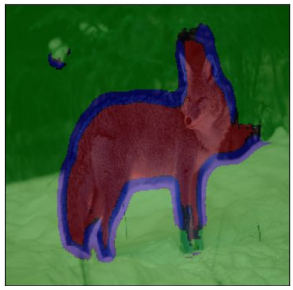
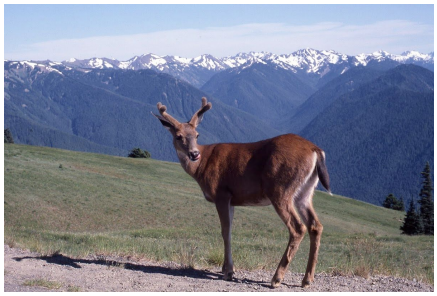


The model was also tested with images of Oregon wildlife. These images were not labeled with trimaps, so visual appearance is the only way to score accuracy.







While this does appear to make more errors than on pet cats and dogs, it is still surprising how well the model predicted trimaps for wildlife in an outdoor setting. Especially for the eagle and ravens, since the model had nothing that looked like a bird in the training set.

For the deer and elk, the model was still able to find the main body, though it did not include the antlers. This suggests that even when the animal has appendages that the model doesn't recognize, it is still able to segment the main body.

Even with a limited dataset of pet cats and dogs that were mostly indoors, the model appears to generalize decently well for different wild animals and settings. This is a very useful feature of this model, as labeled trimaps for a significant number of wild animals would be time consuming to make.

One possibility of future research is to add images of outdoor settings such as mountains, forests and rivers that do not contain animals, labeled with entirely background trimaps to the training set. This may reduce the model incorrectly classifying parts of the background as seen in the images with the deer and elk.

The potential for this model is vast, particularly in the field of machine vision. Using still-frames from video input, the model could be used by an agent to preemptively detect obstacles. Its versatility can easily be extended past basic animal-background detection and can be used to identify changes in depth of field that could signify dangers such as rapid changes in elevation.

Citations

Parkhi, O. M., Vedaldi, A., Zisserman, A., & Jawahar, C. V. (2012). The Oxford-IIIT Pet Dataset [Digital image]. Retrieved from https://www.robots.ox.ac.uk/~vgg/data/pets/

Molina, D. (2018). Oregon Wildlife Wildlife image collection [Digital image]. Retrieved from https://www.kaggle.com/virtualdvid/oregon-wildlife/activity

MusiGenesis. (2010, June 24). Re: How do I choose an image interpolation method? (Emgu/OpenCV). Retrieved from https://stackoverflow.com/questions/3112364/how-do-i-choose-an-image-interpolation-method-emgu-opencv.

Savagave, A., & Patil, A. P. (2014). Study of Image Interpolation. *International Journal of Innovative Science, Engineering & Technology*, *1*(10), 529–534. https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/how_to_model_the_blur_in_Image/attachment/5d8758afcfe4a7968dcdee5d/AS%3A805891544727554%401569150925930/download/Study+of+Image+Interpolation.pdf

sthlm58. (2015, July 20). Re: How do I choose an image interpolation method? (Emgu/OpenCV). Retrieved from https://stackoverflow.com/questions/3112364/how-do-i-choose-an-image-interpolation-method-emgu-opencv.

Wright, Tom. (2010, June 24). Re: How do I choose an image interpolation method? (Emgu/OpenCV). Retrieved from https://stackoverflow.com/questions/3112364/how-do-i-choose-an-image-interpolation-method-emgu-opencv.

Zhang, J. (2019, October 18). UNet — Line by Line Explanation [Digital image]. Retrieved November 5, 2019, from https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5