Machine Learning Engineer Nanodegree
Capstone Project Report
Cassandra Carr Sugerman
August 19, 2018

I. DEFINITION

Project Overview

Object detection is an evolving area in the field of machine learning and computer vision. This project seeks to utilize a known object detection algorithm and fine tune the model for a new dataset. Through this implementation, a deeper understanding of convolutional neural networks (CNNs) and computer vision, as applied to object detection, seeks to be established.

The domain of object detection combines localization and classification through the concept of searching an image for a variable number of objects, identifying the objects, and then classifying the objects. There have been several advancements in the area of object detection between 2012 and today. Originally object detection was completed with exhaustive search, where a sliding window algorithm was used to search the entire image for objects, resulting in high computation times (Rey, 2017). In 2014, R-CNN was developed, which used selective search (Uijlings, 2012) as an improvement to exhaustive search. Selective search used grouping methods to produce a series of object proposals. R-CNN took the object proposals and input them into a CNN which output probabilities for each object class (Ouaknine, 2018). Due to selective search being computationally expensive, a few years later, Fast R-CNN and Faster R-CNN were developed to improve selective search and start leading progress towards real time object detection. Faster R-CNN replaced selective search with a Region Proposal Network (RPN) which generates region proposals within the CNN, see Figure 1 below (Shaoqing, 2016).

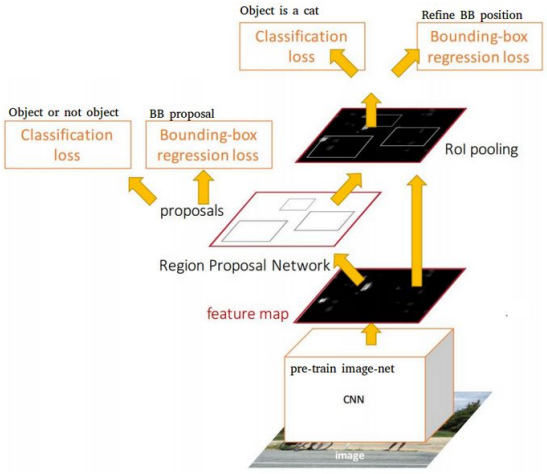


Figure 1: Faster R-CNN architecture

Following Faster R-CNN, other methods were studied to improve speed and accuracy of object detection. One worth noting is YOLO (You only look once), which simplified the model, allowing

for real time object detection, but at the expense of accuracy. The model divides the image into a SxS grid and identifies B bounding boxes for each cell of the grid, with each box having a confidence value. Each cell of the grid is also assigned a class probability. Non-maximum suppression is then used to combine multiple bounding boxes which are identifying the same object (Ouaknine, 2018 & Redmon, 2016).

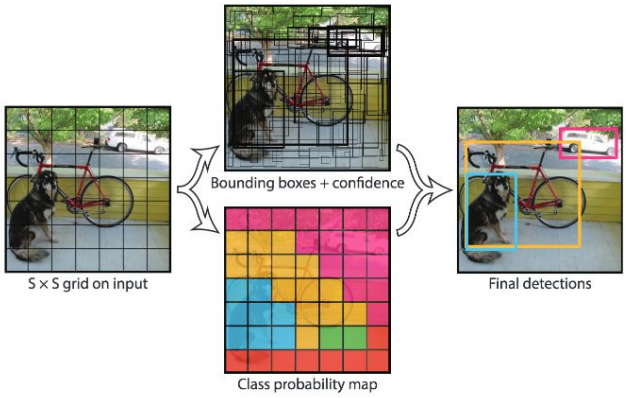


Figure 2: YOLO architecture

Work is still being done in the area of object detection to further improve speed and accuracy. This project aims to take the YOLO architecture and utilize it for a single class, people, in order to see if accuracy can be improved. The classification and detection of people in a video stream is useful in applications such as surveillance and autonomous driving. If successful for one use case, the architecture could then be applied to other objects.

Resources:

Arthur Ouaknine. “Review of Deep Learning Algorithms for Object Detection”. Feb 5, 2018. https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Javier Rey. “Object detection: an overview in the age of Deep Learning”. Aug 30, 2017. https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. May 9, 2016. https://arxiv.org/pdf/1506.02640.pdf

J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. “Selective Search for Object Recognition”. 2012. http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. Jan 6, 2016. https://arxiv.org/pdf/1506.01497.pdf

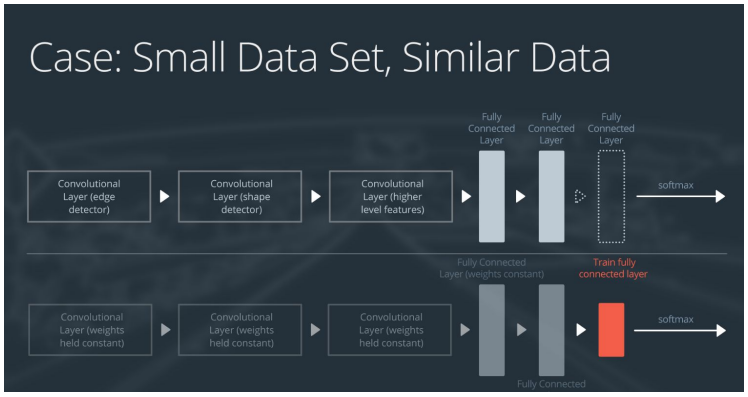
Problem Statement

As described above, accurate and fast object detection is a domain which is continually being improved due to the many machine learning applications where it is applied. These areas encapsulate autonomous driving, video surveillance, and much more. This machine learning capstone project seeks to use object detection to identify people in a video stream. The goal is to develop an understanding of the current work being done in the field and implement the YOLO architecture on a specific problem, people detection. YOLO was documented with a frames per second (FPS) speed of 155 and mean average precision (mAP) of 52.7%. The goal of this project is to improve accuracy when implementing on one class, people, and show that the model can be retrained for a variety of different classes and / or objects.

The dataset used for training was originally noted to be the INRIA Person Dataset (http://pascal.inrialpes.fr/data/human/). This dataset is made up of positive images, which include people, and negative images without people. For all the positive images, there is an associated text file with annotations in Pascal Challenge Format which indicates the bounding box for each person in the image. Unfortunately, the Pascal Challenge Format was found to not be common and it was determined that using Pascal VOC 2007 data (https://pjreddie.com/projects/pascal-voc-dataset-mirror/) would be easier to use initally, as the format and dataset has been widely used.

After the model was successfully trained and tested on the Pascal VOC 2007 dataset (only images labeled with “people” were used), video data was used to test if the model could detect people in a video stream. The stock videos used can be found here:
https://pixabay.com/en/videos/lake-park-people-walking-summer-11571/
https://pixabay.com/en/videos/city-skyscraper-bokeh-blur-6388/
https://pixabay.com/en/videos/street-walk-urban-city-walking-5025/

The model retraining process follows the transfer learning case where we have a small data set with similar data. The pretrained YOLO model weights were loaded, the last fully connected layer removed, and a new fully connected layer was added to create a new model. The model was then retrained, keeping the weights associated with the other layers constant. (Udacity Machine Learning Module & Chu, 2017)



Resources:
Udacity Machine Learning Module: Lesson 2: Convolutional Neural Networks.

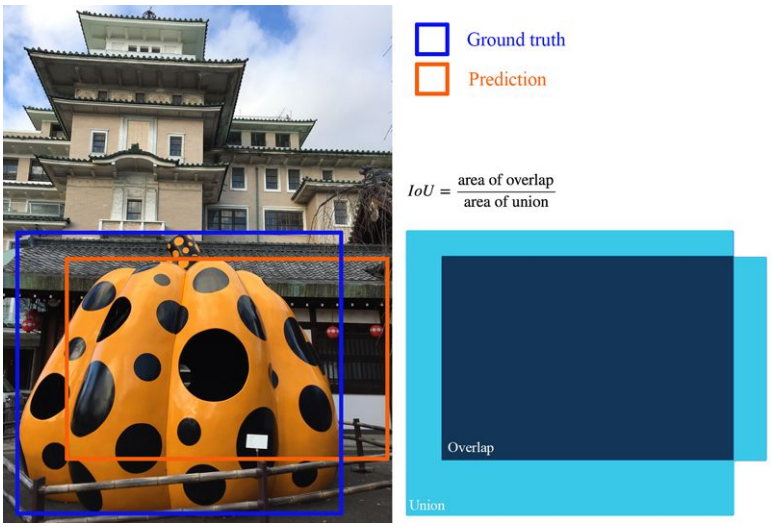
Greg Chu. “How to use transfer learning and fine-tuning in Keras and Tensorflow to build an image recognition system and classify (almost) any object”. May 1, 2017. https://deeplearningsandbox.com/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2

Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. Dec 20, 2010. https://hal.inria.fr/file/index/docid/548512/filename/hog_cvpr2005.pdf

Metrics

The solution for this project was developing an object detection algorithm for finding and detecting people in a video stream. The model developed will take images from a video stream and output bounding box(es) of all people in the images, then overlay the bounding boxes, generating a new video stream. The classification and localization used to identify the bounding boxes was taught at once through a deep neural network and evaluated with the metric mAP (mean average precision). mAP is a widely used metric in the area of object detection and gives a good evaluation of how well the model’s predictions match ground truth bounding boxes.

Mean average precision was calculated through a series of steps. First, the validation data set was evaluated to find all of the model’s predictions. Following this, the number of true positives and the number of false positives were identified. A true positive is defined as a prediction where the intersection over union (IoU) was greater than or equal to 0.3. The image below shows that IoU is the area of overlap between the predicted and ground truth bounding box divided by the area of the two bounding boxes not consumed by the overlap.



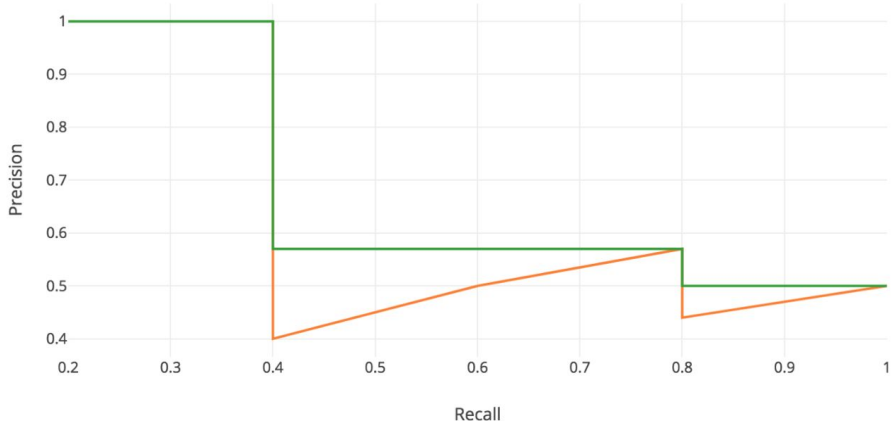
With the number of true positives and false positives defined, precision and recall can then be calculated as shown below.

$$TP = \text{True positive}$$
$$TN = \text{True negative}$$
$$FP = \text{False positive}$$
$$FN = \text{False negative}$$

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

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

The predictions were ordered by confidence value, then precision and recall were calculated appropriately. With these results, the area under the precision recall curve could then be calculated, giving the average precision. An example of an approximated precision recall curve is shown by the green line below.



mAP is the mean of all average precisions for each class. Since in this case we are only considering one class, average precision is equivalent to mean average precision.

Resources:

Hui, Jonathan. “mAP (mean Average Precision) for Object Detection.” Mar 6, 2018. https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

II.ANALYSIS

Data Exploration

It was determined that images containing people from the Pascal VOC 2007 dataset would be used for training. The dataset contains 1070 images for training and 1025 images for validation containing people. The images came in a variety of sizes but were all resized to 416 x 416 for

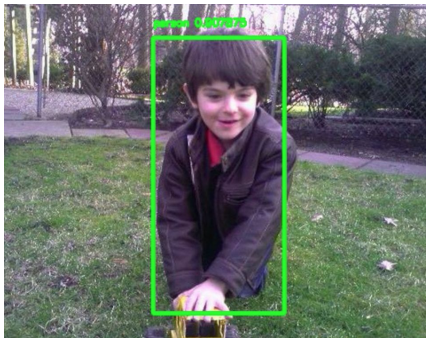
training. There were no other abnormalities found in the dataset. Below is an example of some of the images found in the dataset.

Exploratory Visualization

For each image in the Pascal VOC 2007 dataset, there is an annotation for that image which contains the bounding box information for all items in that image. The first section of the annotation file contains information about the file, including the file name and size of the image. The next section lists the bounding box for each labeled item in the image. Below is an example from an annotation file with an object labeled “person”. There could be a number of different objects within the file with the same or different labels.

```
<object>
        <name>person</name>
        <bndbox>
                <xmin>399</xmin>
                <ymin>219</ymin>
                <xmax>486</xmax>
                <ymax>371</ymax>
        </bndbox>
</object>
```

The bounding box information for each object (xmin, ymin, xmax, ymax) indicate the ground truth bounding box for that object. A plotted bounding box is shown below. This is the intended output for the model, developing bounding boxes that correspond to the ground truth bounding box. How this is done is explained in the following sections.

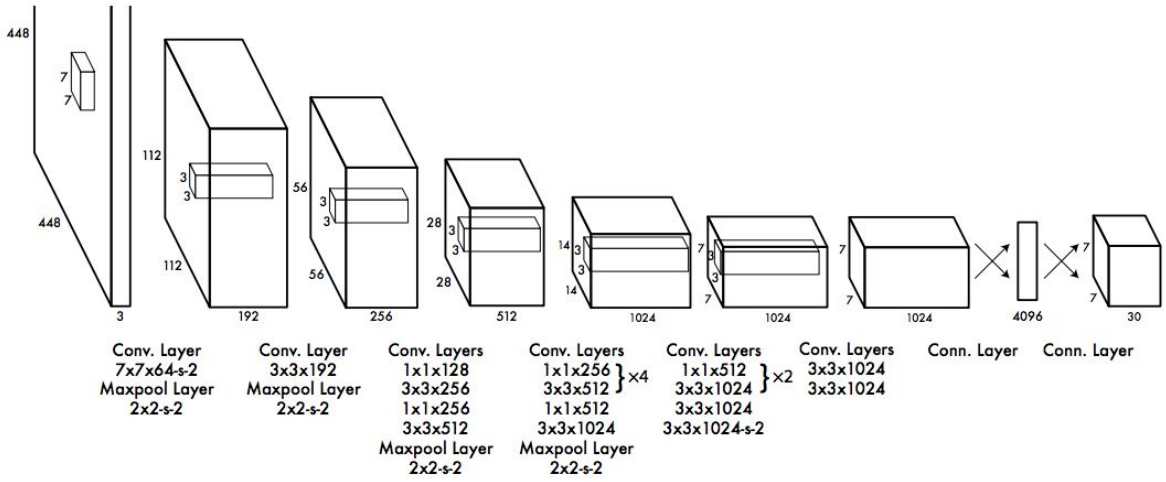


Algorithms and Techniques

As described in the Project Overview section above, a convolutional neural network (CNN) is the best choice for object detection problems because it is optimized for images as inputs. The CNN used in this project follows the version 2 YOLO algorithm (Redmon). The YOLO model is one of the leading object detection algorithms and has been frequently used to achieve positive and fast results for object detection of various classes.

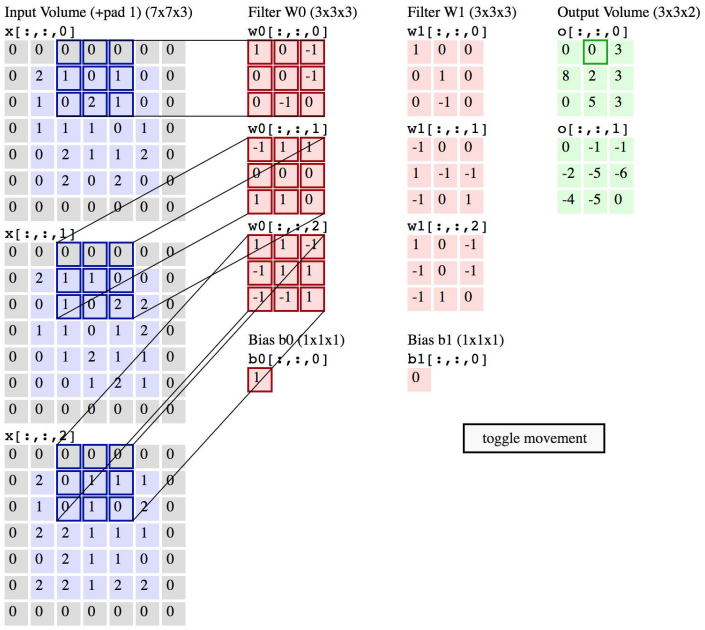
The YOLO model consists of 23 convolutional layers, and for this project, pretrained weights were loaded from the original model. For the first 22 layers, the weights were kept frozen, while the last layer was randomized and retrained. This follows the transfer learning case where we have a small data set with similar data.

The YOLO model CNN has a general pattern of the following layers: convolutional 2D layer, followed by a batch normalization layer, followed by a leaky relu activation layer, and occasionally, a max pooling 2D layer. The diagram below shows a visual of the YOLO version 1 model. Note that some differences exist between this model and the one used for this project, as YOLO version 2 was implemented. The output of the model from the YOLO version 1 paper is a 7x7x30 matrix. However, this is dependant on the number of classes (C), the grid size used when searching the image (S), and how many bounding boxes are set to predict in each grid cell (B). Since each bounding box prediction has 5 values (x, y, h, w, and confidence), the formula used for the output is (S x S x (B*5+C)). So, in this project, since we have a grid size of 13 x 13 (S x S), 5 predictions per cell (B), and 1 class (C); our output is (13 x 13 x 26). (Redmond)



The convolutional 2D layers of the YOLO model work to identify the key features of the image by searching the image with filters. A visual of this can be seen in the image below. A filter of a set size slides across the image capturing the feature information. Multiple filters can be applied in each convolutional layer producing an output with a depth of the number of filters applied.

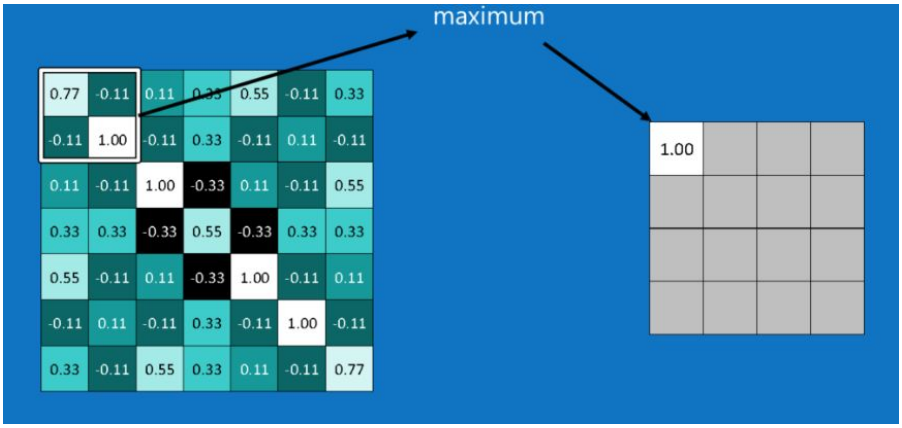
The weights and bias of each filter are randomly defined for the first pass, but are adjusted through back-propagation as learning takes place.



The batch normalization layer is then applied after the convolutional layer, which normalizes all feature values to be between 0 and 1. This reduces the amount the hidden layer weights shift, allowing increased learning rates and shorter training time. It also has some regularization effects which reduces overfitting. (Doukkali)

After the batch normalization layer, an activation layer is applied. In this model, the Leaky ReLU activation function is used. This takes all feature values, and for each value that is negative, it turns it into a small negative number (close to 0).

The YOLO model also has some max pooling layers at various locations towards the beginning of the network. The max pooling layers shrink the number of features by preserving the most important features. A defined window slides across the feature map and takes the largest value in the window. This helps speed up training without losing important feature information.



Resources:

Stanford CS231N. “Convolutional Neural Networks (CNNs / ConvNets).”
http://cs231n.github.io/convolutional-networks/#add

“How do Convolutional Neural Networks work?”
https://brohrer.github.io/how_convolutional_neural_networks_work.html

Doukkali, Firdaouss. “Batch normalization in Neural Networks.”
https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c

Joseph Redmon, Ali Farhadi. “YOLO9000: Better, Faster, Stronger.”
https://pjreddie.com/media/files/papers/YOLO9000.pdf

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. May 9, 2016. https://arxiv.org/pdf/1506.02640.pdf

Benchmark

As mentioned above, the benchmark model used for this project was the YOLO version 2 model. The goal was to take the same deep neural network used for object detection and fine tune the model for people identification. After fine tuning, mAP (mean average precision) was evaluated in an attempt to improve upon what was achieved with the YOLO model, 52.7%. Since FPS (frames per second) was difficult to optimize due to computing power available, this will not be evaluated against the original YOLO metrics.

III.METHODOLOGY

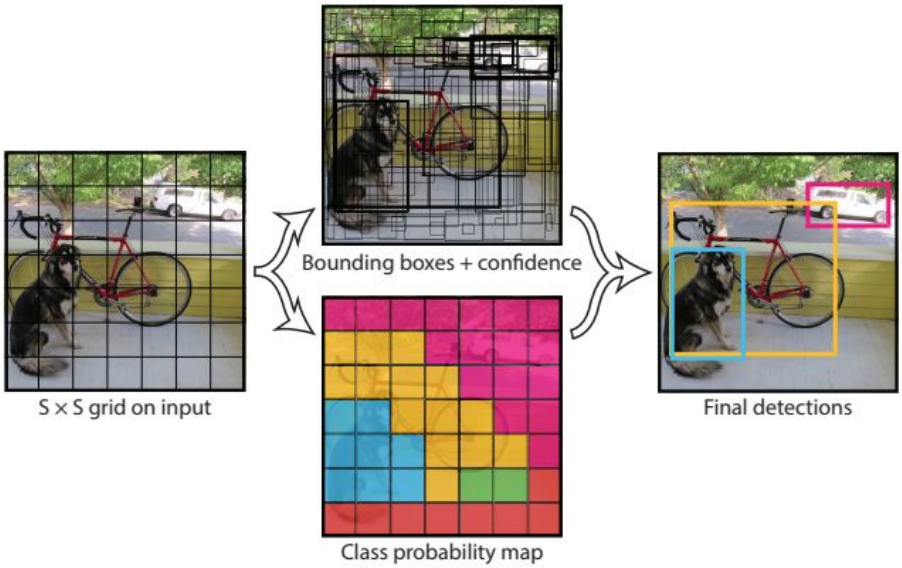
Data Preprocessing

First, it is important to note that the keras-yolo2 repository was used to lay the groundwork for this project (https://github.com/experiencor/keras-yolo2). It was a similar project that used keras to recreate the yolo model and retrain the model for a new dataset.

For data preprocessing, the images were first normalized by dividing the pixel arrays by 255. The images and associated annotations (xml files) were then appropriately parsed. A batch generator was then used during training. The batch generator first applied an augmentation algorithm to each image in the batch, which performed random scaling, translation, and flip to the image. The object information, parsed from the annotations, were then looped through in order to get the bounding box information for each object labeled in the image. The image and true bounding boxes are then passed to the model.

Implementation

The model design for this application is based on the YOLO version 2 deep learning model. For this project, the model was built in Keras and precisely followed the YOLO version 2 architecture, which consists of 23 convolutional layers. Thanks to experiencor, the Keras code for this was already developed. The only changes made to the code were to the parameters GRID_H, GRID_W, BOX, and CLASS. GRID_H and GRID_W are the defined grid size (S x S as seen in the image below). This was set to 13. BOX is the number of predictions for each grid cell. This was set to 5. CLASS is the number of classes the algorithm is searching for. Since in this project we were only concerned with “people”, there was only one class.



The next step was to download the pretrained weights from the YOLO website. The pretrained YOLO model weights were loaded into the Keras model and the weights for the first 22 layers were kept frozen. The 23rd layer was changed to output the correct size based on the parameters GRID_H, GRID_W, BOX, and CLASS defined above. The weights for the 23rd layer were then randomized so they could be retrained.

For training, Keras `model.fit_generator` was used. A batch generator, as described in the Data Preprocessing section above, was used to generate the training and validation data for the fit generator. The BATCH_SIZE was set to 16. A number of Keras callbacks were also utilized. `EarlyStopping` was implemented in order to stop training after the model stopped improving. In addition, `ModelCheckpoint` saved only the best performing model weights based on validation loss. The optimizer for the fit generator was first defined as an Adam optimizer with a learning rate of 0.5e-4. The number of epochs were first set to 2 until the model was working and then increased to 10. These parameters were later changed as described in the Refinement section below.

Refinement

The evaluation for this model was mostly done qualitatively by looking at how accurate the model performed on a set of test images containing people. On the first iteration of training, it was found that the model produced a large number of false positives. As an attempt to correct this, a various number of epochs were trialed, eventually settling on 20. This was when the validation loss was no longer improving. However, the false positives still remained. After some research, it was determined that changing the parameters NO_OBJECT_SCALE and CLASS_SCALE could help address the false positives. These parameters affect how much to penalize wrong predictions of non-objects and how much to penalize wrong class predictions. NO_OBJECT_SCALE was changed from 1 to 7, and CLASS_SCALE was changed from 1 to 3. This didn't result in much improvement, so increasing each value to 10 was attempted. This again resulted in little improvement. BATCH_SIZE was then also adjusted to see if it would have any effects on the results. The BATCH_SIZE was changed from 16 to 30. Again, there was little change in how the model performed detecting people in the test images. NO_OBJECT_SCALE, CLASS_SCALE, and BATCH_SIZE were then set close to their original values of 5, 3, and 16.

With these parameters not causing significant changes in model performance, more research was done. It was then determined that a different optimizer may develop better results. The optimizer was changed from Adam to RMSprop. It was this change that resulted in a model which correctly identified the people in the test images.

IV.RESULTS

Model Evaluation and Validation

Overall, the model developed showed to perform well. A set of trial images were ran through the model and the predicted bounding boxes identifying people in these images were accurate. In addition, three videos containing people were evaluated by the model with great results. One of these videos can be seen here. The bounding boxes produced by the model in these videos are accurate and show that minimal overfitting exists in this model because the images in the videos are not similar to what is found in the training data set. This is also justified by the validation loss observed during training, as training was stopped when the validation loss stopped improving.

The mean average precision (mAP) was calculated on the validation set and was found to be 65.96%.

Justification

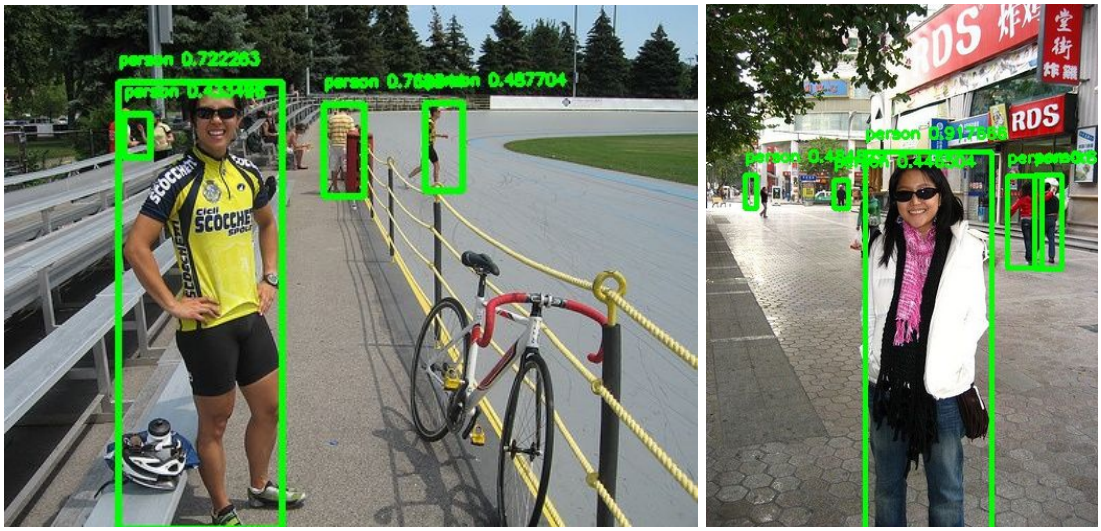
As noted above, the model developed in the project to identify people showed a mean average precision of 65.96% on the validation set. This is an improvement over the original YOLO model, which achieved 52.7%. It is important to note that the YOLO model identified multiple

classes, where in this case, only one class was trained. However, overall, we can state that the model is satisfactory compared to the benchmark.

V.CONCLUSION

Free-Form Visualization

The model developed in this project was first tested on a series of images containing people and it was found to identify people in the images accurately. See below for examples.



The model was also used to find people in multiple video streams. One of the results can be found here: https://youtu.be/6I3_GQn-evQ.

Reflection

This project aimed to develop an object detection algorithm for identifying people in a video stream. Based on object detection research, it was determined that the YOLO model would be a good candidate for this application. The YOLO deep neural network was recreated and weights from the original model were loaded. The last layer of the model was then modified, weights randomized, and the model was retrained on a new dataset containing only people. Originally the model gave too many false positives, but after refinement, the model proved to be successful in identifying people in a video stream.

During the project, a number of barriers were overcome. It was concluded that keras would be an acceptable platform to recreate and run the model, so a benchmark repository was found which used keras to recreate the YOLOv2 model (https://github.com/experiencor/keras-yolo2). This code was the baseline for this project and was modified for the dataset at hand. This was the first major challenge faced, as many iterations were performed before a working model was

established. This ended up being a good learning experience because the code was deep dived and better understood. After the code was working and giving results, there was still fine tuning that needed to occur in order to develop an accurate model. This was the second major challenge faced. AWS (Amazon Web Services) was used to speed up the training process and allow for an appropriate number of iterations, however, training time was still very long. Multiple parameters were changed and eventually an accurate model was developed.

Improvement

There is still room for improvement to be made on this model. However, due to the long training time, it was difficult to do a lot of experimenting. It would be beneficial to change the model to the YOLO tiny model and see if similar, or even better, results can be accomplished. The tiny YOLO model has fewer layers and would take less time to train. In addition, improvement could be made to the model by looking at other optimizers, increasing the number of epochs, adding more pre-processing, adding dropout, and changing the optimizer parameters such as learning rate. In addition, it would be interesting to make adjustments to the code to easily train on a variety of other datasets and see how it performs at identifying different objects. The YOLO model has great potential for being a useful object detection algorithm in many instances.