Object Detection Algorithms

Vatsal Patel and Aakash Zaveri

patel.vatsal1@northeastern.edu zaveri.a@northeastern.edu

Abstract

The problem dealt with the Tobacco-800 dataset (Doermann) which contained images with letter heads of tobacco companies or a signature. Each image had either a logo, signature, or both in them. We performed object detection on those images and found optimal hyper-parameters to achieve high precision and high mMAP 0.5(one of the accuracy metrics for object detection task) on the images. Some images in the dataset were blank or either had no logo or signature(Zaaboub et al. 2020) in them, so some data pre-processing will have to be performed to clean the dataset. The images of legal documents of tobacco companies, which meant that a lot of these images had a lot of text surrounding signatures and logos, making object detection harder. Overall, the goal was to compare two object detection algorithms on the dataset and after running the YoloV5 and Resnet-50 algorithms on the dataset, we found that YoloV5 was a lot faster and simpler whereas Resnet-50 was much more complex and was very slow.

Introduction

Object detection is an emerging feature in the field of artificial intelligence and machine learning. Self-driving cars and autonomous robots utilize object detection to avoid running into obstacles when completing tasks. Another use for object detection is to detect specific details on images such as a signature or certain keywords in a group of text (Mandal et al. 2014). One of the main motivations for this project is to compare an older algorithm known as Resnet-50 to a quite newer algorithm called YoloV5 and look at the differences in results when training them on the same dataset.

We planned on comparing the two algorithms on multiple different metrics. We compared the Yolov5 and Resnet-50 algorithm under different circumstances. The comparisons we made included a precision and recall metric comparison between Yolov5 and Resnet without performing any data augmentation, Precision and Recall metric comparison between these algorithms with data augmentation performed on these datasets, mMAP 0.5(metric) comparison between the algorithms with and without data augmentation, training time comparison between these two algorithms for the same amount of data and image size, and check the performance of the algorithm on different hyper-parameters such as learning rate, momentum, different anchor box sizes, number of warmup epochs etc.

Background

The two main algorithms being used for this project are YoloV5 and Resnet-50 which are two object detection algorithms. These algorithms are typically used for object detection tasks and the basic structure of the algorithms is already available and open sourced. We modified the data and annotations in a way to have a valid input for datasets and modified the last layers of the algorithms to make them suitable for the datasets.

People generally use the pretrained weights (by training the algorithm on very large datasets) and use these weights for most of the object detection tasks as these algorithms are already trained on very big multi-class datasets and most of the common object detection tasks are sufficed by these pre-trained weights. We trained the model on our dataset, derived the newly trained weights, and used them to make predictions.

Related Work

Pretrained models such as Yolo (Redmon et al. 2016) and Resnet (He et al. 2016) are what makes transfer learning tasks very easy and feasible in machine learning especially in tasks pertaining to computer vision. These models have been studied for a very long time and several iterations of the same model have been trained, each designed to perform a particular size. Both models have been trained on very huge datasets. To make it useful and perform object detection on our dataset, we just need to change a few of the last layers of the network to output detections pertaining to our task and supply it with our dataset and annotations of the datasets to train the model. Both the Resnet model and the yolo model first came out in

2015 and both at the time outperformed all the previous object detection models by about 10% to 15%. Since then, these models, especially the Yolo model have gone through many iterations and most recently, the Yolov5 model was released. This release made it very easier to train on our own custom dataset as well as made the training very fast (this depends on the type of GPU you have). The Resnet model, however, has not been very maintained, and since its release, Tensorflow and PyTorch have trained the model and made it available for use, but it is very difficult to train on your own custom dataset. Nonetheless, these models are still considered pinnacles of the objection detection world, still outperforming several state-of-the art models.

Approach

Resnet was one of the most interesting object detection models to be published after the AlexNet (Krizhevsky et al. 2017) model blew the computer vision world by storm in 2012. The Resnet framework made it possible to train deep, very deep neural networks that contained thousands of layers and still achieve great performance, despite having a very large number of deep neural layers.

The deep neural networks are very good at learning low, mid, and high-level feature representation of the images. The earlier layers of the model are very good at learning low level features and as we deep into the network, it starts learning high level features. Generally, it is considered that stacking more layers on top of one another achieves a higher accuracy on the task at hand. But with this, there comes the problem of vanishing and exploding gradients, whereby the gradients of very deep neural networks would either converge and set the weights to zero or to infinity, there by stop learning and predicting random guesses to the images and saturating the accuracy and then degrading it rapidly, thereby increasing the training loss.

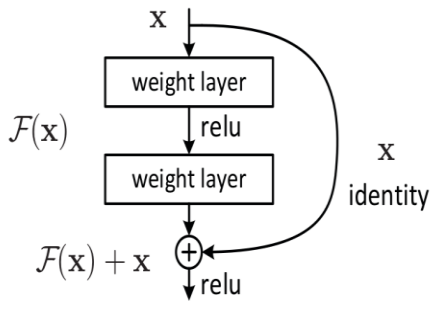

Figure 3: Single Residual Unit

This problem was rectified by taking a shallower model and a deep model that was constructed with the layers from the shallow model and adding identity layers to it, thereby introducing what they called skip connections, addressing the issue of vanishing, and exploding gradients.

The final output of this residual block would be F(x) + x (Figure 2) with the help of the shortcut connection/skip connection with the added benefit of

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

no additional parameters added to the model and thereby keeping the computational time of the model the same.

Figure 1: Model Architecture of Resnet Model

The authors of the original resnet paper published the resnet-50 architecture and that can be seen in Figure 1. The main difference in Resnet 50 model to 18- or 34-layer models is that before this the shortcut connection skipped two layers but now, they skip 3 layers as seen in Figure 2.

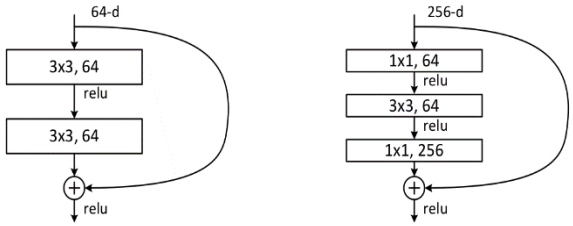

Figure 2: Residual Unit Detailed Architecture

The (x 3) in each convolutional layer depicts that each convolutional layer is made up of 3 of those residual layers shown in the above image. The results from the original resnet paper are shown below and we can see that Resnet 50 offers a sweet spot between the depth of the model as well as the error it achieves, but the selection of the model highly depends upon the task at hand.

Before the advent of YOLO (you only look once) there were models such as Fast-RCNN(Girshick 2015) , RCNN (Girshick et al. 2014) that were two-

stage object detection models. These models repurposed the classifier to detect objects in one go and then in the next go classify those objects. Yolo combines this into one single step, detecting the bounding box as well as classifying the objects in those bounding boxes, all in on go and hence the name. The YOLO algorithm works by dividing the image into *N* grids, each having an equal dimensional region of SxS. Each of these *N* grids is responsible for the detection and localization of the object it contains. Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell. The Yolo model architecture can be seen in Figure 4.

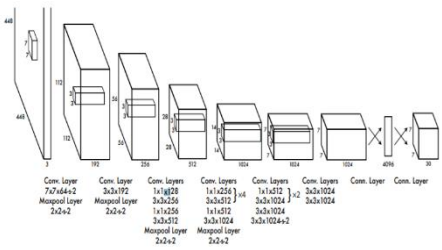

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Figure 4: Yolov1 Architecture

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU (Zenggyu) to provide an output box that surrounds the objects perfectly. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box. This process greatly lowers the computation as both detection and recognition are handled by cells from the image, but it brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions. To counter this issue, YOLO uses Non-Maximal Suppression (Redmon et al. 2016)to limit the number of duplicate bounding boxes. It suppresses all the bounding boxes that have a lower probability score of having an object in them. The architecture of the yolo algorithm is shown in the below image. This network was inspired by the Google LeNet network, and it contains 24 convolutional layers followed by two fully connected layers. The rest of the architecture is self-explanatory.

Experiments and Setup

Both the models were trained on the Tobacco-800 Dataset which contained both signatures and logos on tobacco companies. Our model was only trained for signatures as large number of the images in the dataset di-not contain any logo. Yolo and Resnet models were modified to train on our dataset. Various image augmentation techniques were initially performed on the dataset, and we compare the results of both these models on this augmented dataset. Image augmentation techniques such as horizontal and vertical flip, random saturation and hue changes to the images, randomized cropping of the images and tilting and rotation of images on various angles were performed on the dataset to create more variations in the training data. Both models were trained on Google Colab with Tesla-P100 GPUs. The training time of both models varied, as they are different models, with different architectures. The Yolov5 model was trained for around 450 epochs, which took around two and half hours to train, whereas the Resnet model was trained for 12000-time steps which took around four hours to train.

For the Yolo model, the dataset was already annotated for us, so we just had to modify the yolo model to take in our own custom dataset and train the model. Also, one of the perks of the latest Yolov5 model is that all the weights and training metrics are stored on wandb (weights and biases) online, and hence makes it very easy to track the training progress of the model.

For the Resnet model, we had to annotate the dataset manually, as the format of annotations taken by the resnet are different from those of the yolo model. Hence, some time had to be spent on annotating the images for the Resnet model. We used tensor board, the de facto method to track the training progress of the Resnet model.

Results

Yolov5 and the Resnet model are considered state of the art when it comes to multiple object detections in a frame. Due to recent innovations in more faster algorithms, we can see that Yolov5 is superior to resnet model, both in terms of achieving higher mAP:0.5 and faster training time. So, let us now look at the results of both models individually. Let's first start with the Yolov5 model(Jocher 2020). As we can see from the graphs below, Yolov5 achieves a Mean Average Precision (mAP:0.5, which is the standard evaluation metric for Yolo object detection task) of 81% (Figure 5) . This means that it will correctly

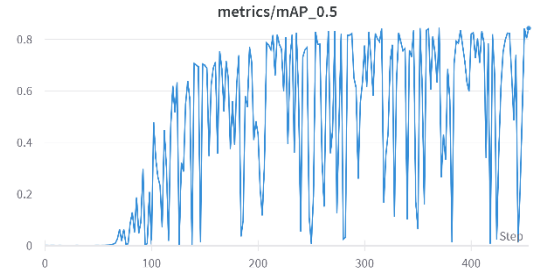

Figure 5: mAP:0.5 for the Yolo Model

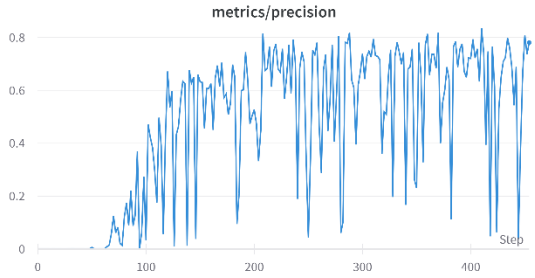

Figure 6: Yolov5 Precision Graph

predict the location of the signature and classify it correctly with a precision of 81%.

Also, over the entire training time, we can see that the object_loss for validation data (the object detection loss, i.e., the model's prediction of the signature location) is also decreasing, and hovers around 0.015 after just 100 epochs. As for the general precision and recall, they both hover around 70% to 80% (Figure 6 and Figure 7), but none the less, they don't matter much, when considering the overall performance of the model. As for the box_loss for validation (how close the box is to the ground truth box) is also decreasing and stays around 0.09, which means that our model is prediction bounding boxes which as very close (in position) to the original annotated training boxes. The predictions of test images can be seen in Figure 8 and Figure 9.

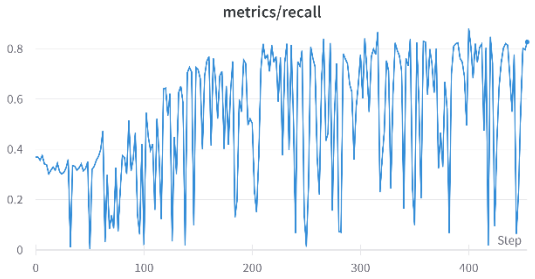

Figure 7: Yolov5 Model Recall Graph

As for the resnet mode, we will also be considering the mAP:0.5. Now, it is not easy to get training and evaluation metrics in a graph for the resnet model, so I am just lay those numbers out. As for the mAP:0.5 for the evaluation dataset after

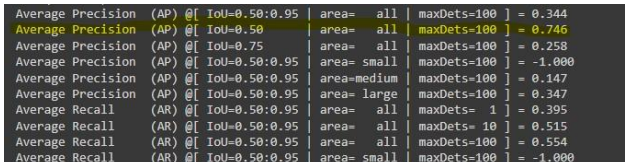

Figure 10: mAP:0.5 for Resnet Model

training for around 12000 steps, it achieves mAP:0.5 of about 73% (Figure 10).

Though there is not much difference in mAP:0.5 value, this is evitable in prediction images given below and the confidence on each of the signatures. The classification loss (Figure 11) (predicting if there is an image in the bounding box or not) is getting very close to 0, which suggests to us that the model is overfitting to the training data. Another evidence of overfitting is given by localization_loss, which is also very close to 0. The problem of overfitting the data with Yolov5 does not arise with the same training data. Prediction on a test image by the resnet model can be seen in figure 12. Now, let's have a quick at the training time of both the algorithms. Both models were trained on Nvidia Tesla P-100 GPU's. The yolo model was trained for around 450 epochs which took around 2.5 hours to train, while the resnet model was trained for around 12000 steps, which took close to 4 hours to train. So, we can see that the Yolo model is about 40% faster to train then

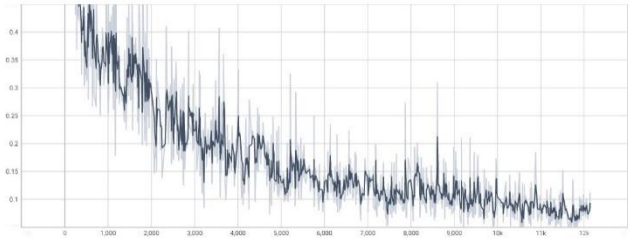

Figure 11: Resnet Model classification_loss

the resnet model and is even more accurate than the resnet model.

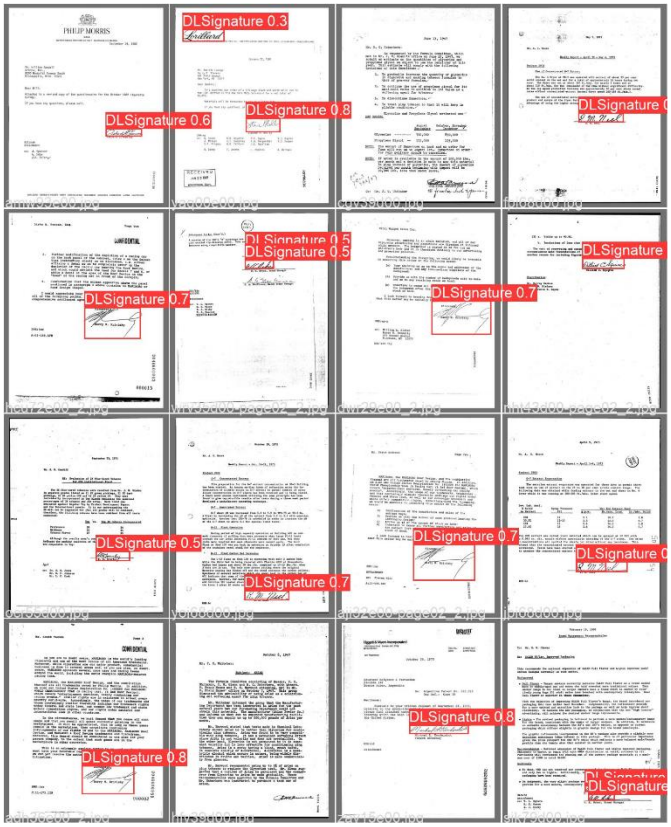

Figure 9: Yolo Model Prediction

Figure 8: Yolo Model Prediction

Conclusion

After viewing the results shown above and analyzing the different relations between the precision and detection of both YoloV5 and Resnet-50, we realized quickly that YoloV5 was more precise and much easier to set up. This is because YoloV5 has continuously been updated and worked on whereas Resnet-50 is newer and much more complicated as it has a lot more warnings and pathways to go through to run it on data. Finally, we concluded that the YoloV5 was also more accurate and faster than Resnet from our results.

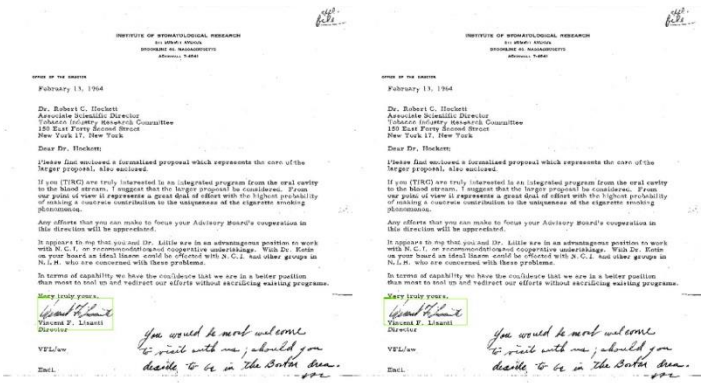

Figure12: resnet Model Prediction

References

Doermann D Tobacco-800 Dataset. http://tc11.cvc.uab.es/datasets/Tobacco800_1

Girshick R (2015) Fast R-CNN. Proc IEEE Int Conf Comput Vis 2015 Inter:1440–1448. https://doi.org/10.1109/ICCV.2015.169

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 580–587.

https://doi.org/10.1109/CVPR.2014.81
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem:770–778. https://doi.org/10.1109/CVPR.2016.90
Jocher G (2020) Yolov5 Github. https://github.com/ultralytics/yolov5
Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60:84–90. https://doi.org/10.1145/3065386
Mandal R, Roy PP, Pal U, Blumenstein M (2014) Signature segmentation and recognition from scanned documents. Int Conf Intell Syst Des Appl ISDA 80–85. https://doi.org/10.1109/ISDA.2013.6920712
Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem:779–788. https://doi.org/10.1109/CVPR.2016.91
Zaaboub W, Tlig L, Sayadi M, Solaiman B (2020) Logo detection based on fcm clustering algorithm and texture features. Springer International Publishing
Zenggyu Yolo Metrics