Domain Adaptation for Object Detection

Janpreet Singh
University of Michigan
janpreet@umich.edu

Clark Arenberg
University of Michigan
carenber@umich.edu

Nihar Bhingaradia
University of Michigan
nbhingar@umich.edu

Abstract

In the modern age of Artificial Intelligence and neural networks, it is crucial for algorithms to adapt to new datasets without deteriorating the accuracy of old/previous datasets. In this paper, we will test the domain adaptation capabilities of the object detection neural network model YOLOv3 [5] on the Laboratory for Intelligent and Safe Automobiles(LISA) [1] Traffic Sign dataset and Russian Traffic Sign Dataset (RTSD) [2]. We will perform cross-data experiments to test how the network adapts to two related but different distributions. In the end, we will modify the model to reduce catastrophic forgetting to keep similar accuracy on the initial dataset while improving accuracy on another dataset.

1. Introduction

The recent advances in Computer Vision have enabled better self-driving capabilities and pushed us toward building a fully autonomous car. Most of the autonomous system in a car relies on neural network models to detect the traffic signals. Our motivation in this paper is to see how good are the object detection models(YOLOv3) for detecting traffic signs on related but different distributions of a Dataset. To simulate this, we used Traffic sign dataset from two different countries, Russia & US and test the transferability of models. Also, we propose a simplistic way to enforce better transferability in the learning process. We observed that using this technique we get a gain in class confidence score as shown in figure 4.

Overall, we will train 2 YOLOv3 models using RTSD and LISA datasets. Then, we will use these datasets to visualize activation maps via Gradient-Weighted Class Activation Mappings, aka. GradCAM mappings, for each. After that, we will be fine-tuning pre-trained models against the other dataset to transfer learning and visualizing the phenomenon of catastrophic forgetting. Finally, we'll attempt to reduce catastrophic forgetting for the YOLOv3 model. [1]

[1] Link to Github repo: repo link

2. Approach

Our work was divided into four main steps. First, we processed the data labels in order to standardize the LISA [1] and RTSD [2] datasets. The second step was to train the YOLOv3 model [4] using this standardized data set and generate the class activation maps using GradCAM [3]. After that, we used transfer learning to see how the network trained on the first data set can adapt to the second data set. Finally, we modified the loss function for YOLOv3 model in a manner that if we are just given weights learned from first dataset, then, we can learn a new model that works efficiently on both, the first dataset and the new dataset.

2.1. Task 1: Training on LISA and RTSD

For this task, our goal was to first train two object detection models on the LISA and RTSD datasets and then visualize the models to better understand how the model is predicting these classes. We used YOLOv3 architecture as our choice of model because of its speed at train and test times. Another upside of YOLOv3 was that, given that it is a single shot network, we could extract activation maps at each layer with more ease when compared to two-stage detectors like Faster R-CNN. Overall, it is a good baseline to do quick experiments and get good metrics in a reasonable time. Figure 1 shows YOLOv3 architecture. As shown, YOLOv3 uses an approach similar to feature pyramids to achieve better learning and results. The final loss function consists of four parts: centroid (XY) loss, width and height (WH) loss, objectness (object and no-object) loss and classification loss.

For investigating the domain adaptation capabilities of YOLOv3 network we first trained the object detector on LISA and RTSD datasets separately. We made a custom configuration file with 15 classes for YOLOv3 and used a batch size of 4. For LISA dataset, we had 4570 images for training and 918 images for validation. For RTSD, we had 3986 number of images for training and 993 images for validation. It took 24 hours of training for each dataset. We started with a learning rate of 0.01 and a final learning rate of 0.0005 with a cosine learning rate scheduler and SGD as the optimizer. After training the network, we used Grad-

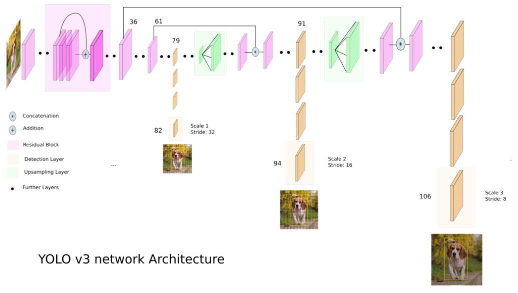

Figure 1. YOLOv3 architecture credits [6]

Dataset	mAP	F1 Score
LISA	0.94	0.82
RTSD	0.8435	0.7024

Table 1. Metrics for Task1

CAM to visualize the class activation maps for each class. We used 105 layers for visualizing the activation maps for YOLOv3.

2.2. Task 2: Investigating YOLOv3 for domain adaptation

Following task1, we took these models and used them as a pre-trained model to fine-tune it on the samples of other dataset. Our aim was to perform transfer learning from the LISA(D1) to the RTSD(D2) dataset and vice-versa. The reason for doing this was to investigate how fine-tuning the pre-trained model on a new dataset would impact the accuracy of the original dataset. The data here was divided into three parts: training images from the new dataset, validation images from the new dataset, and validation images from the original dataset. By doing that, we could monitor how the mAP (mean average precision) changed on the original dataset and the new dataset. Firstly, we took the model trained on the RTSD dataset in the first step and use this as a pre-trained model to fine-tune it on samples of LISA dataset. While fine-tuning, we trained the model on different numbers of samples from the LISA dataset. Through this, we look at the impact of catastrophic forgetting on YOLOv3.

For fine-tuning, we used the last 15 layers as the learnable layers and froze the first 85 layers. For example, the initial model was trained on RTSD dataset and this model was used to fine-tune it on a sample of LISA dataset. Here, we used incremental samples of 100, 200, 300, 500, 600, 800 and noticed that increasing the number of samples increased the mAP score on that data but decreased the mAP on the other data and hence validating the famous phenomenon of Catastrophic forgetting in neural nets. Figure 5 shows the number of samples plotted against the highest mAP and F1 score achieved. We kept the same hyperparameters across all the experiments and trained them on a batch size of 4. Here, again we started with a learning rate of 0.01 and a final learning rate of 0.0005 with cosine learning rate scheduler and SGD as the optimizer.

2.3. Task 3: Modify loss function for better learning

In this task, we made an attempt to overcome catastrophic forgetting for the YOLOv3 model. To explain what we did, let's take a scenario that we have a pre-trained model on RTSD dataset and we are fine-tuning it on LISA dataset. The goal is to produce a model that performs well on both LISA and RTSD datasets after fine-tuning it on LISA dataset. The loss function was then modified in the following manner: we took weights from the last 15 layers from RTSD model, subtracted these weights from the learned weights on the LISA dataset at each epoch and took the mean of absolute value, and added this term as a regularizer in the loss function for YOLOv3. The addition here was a weighted sum between loss from YOLOv3 and the difference of weights. This could also be seen as a regularization of weights at a certain value. Usually, we take L1 norm from the origin but in our case, we took L1 norm from an existing matrix.

3. Experiments

The experiment was performed in a PyTorch 1.4 environment. The models were trained on an Nvidia GTX 1070 GPU. Broadly, the experiments were divided into three categories. The first category consists of training YOLOv3 model on RTSD and LISA datasets. The second category consists of probing the domain adaptation capabilities of YOLOv3 and the third category discusses the change in the loss function we made to enhance the domain adaptation capabilities of YOLOv3.

3.1. Data

There are 32 classes in LISA and RTSD datasets with 6736 images and 7198 images respectively. Figures 5 and 6 show the class distribution of LISA and RTSD datasets. Some of the classes had very few images to train the model so we had to reduce the classes to 15. On top of this, RTSD dataset had numeric encoding for the class names so we had to manually assign the name of classes so that it matches with LISA dataset. We also merge images for some classes since they were very similar traffic signals. After standardizing the datasets we divided images so that the size of the image is 416 * 416. After that, we normalized the pixel value so that it's between 0 to 255.

3.2. Metrics

For task 1, when training the models, we evaluated their mAP(mean Average Precision) and F1 scores on both training and validation datasets where mAP is computed using

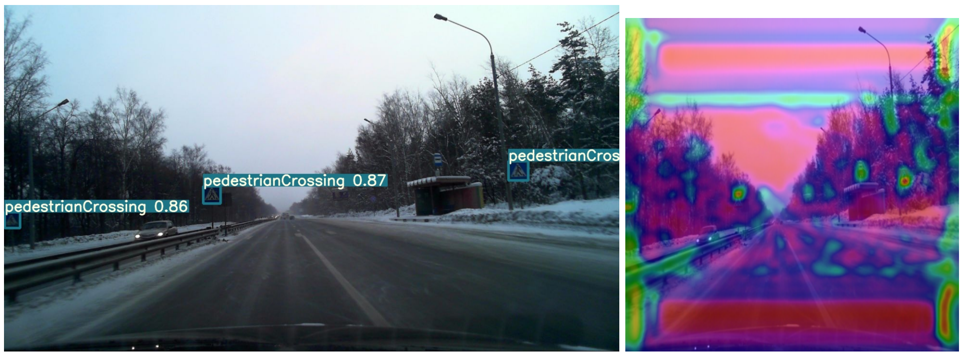

Figure 2. On the left we have the detection result from RTSD model in task1 and on the right we have the GradCAM output

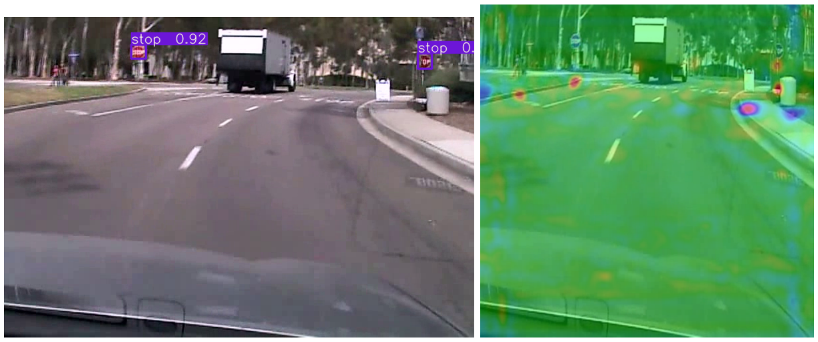

Figure 3. On the left we have the detection result from LISA model in task1 and on the right we have the GradCAM output

precision = True Positive/(True Positive + False Positive). Also, F1 score here is the harmonic mean of precision and recall where recall is defined as TP/(TP+FN). For computing TP, FP and FN, we use the intersection over union(IOU) threshold as 0.5. When performing transfer learning on task 2 and task 3, we evaluated models based on their mean average precision or mAP scores.

3.3. Quantitative Results

For task 1, Table 1 shows the mAP and F1 scores for the validation data when two models were trained on RTSD and LISA datasets. We can see the due to better data distribution in LISA, YOLOv3 has better performance.

For task 2 and task 3, Figure 7 shows the change in mAP values on validation data between the model using the original loss function vs the model using the adapted/modified loss function. Here, on the X-axis we have the number of samples taken from the LISA dataset for fine-tuning and the mAP value on the validation data on the Y-axis. The validation data was kept the same across all the experiments. We can see how the use of the modified loss function tries to do a trade-off between the mAP values. Though we get a jump in the mAP values for RTSD data, on the other hand, the mAP values on LISA go down.

3.4. Qualitative Results

YOLOv3 model on RTSD and LISA were able to detect traffic signs quite well. Additionally, we used GradCAM for a better visual explanation of the trained model. The example shown in figure 2 suggests YOLOv3 model on RTSD detects traffic signs with high accuracy. GradCAM result also showed the heat maps near traffic signal as shown in the figure. Heatmaps were also present around the signal on top of the bus station. This shows our model is looking for features in the image that looks like a traffic signal. YOLOv3 model trained on LISA dataset also gave us better results detecting “STOP” signal as shown in Figure 3. While running GradCAM on the image, we noticed odd heatmaps for stop signs: as often as not, the algorithm would pick up on the base of the sign or the curb around the sign as it would on the sign itself. We suspect that this is because of the placement of stop signs: unlike the other traffic signs, they would be placed more or less exclusively at intersections, and so the shape of an intersection would be a strong signal that the sign in question is a stop sign.

We noticed a significant reduction in accuracy during transfer learning from RTSD to LISA. This is shown in Figure 4, where the confidence score of detection in the image decreased significantly from the model in task 1. Through modification of loss function and regularization, result was

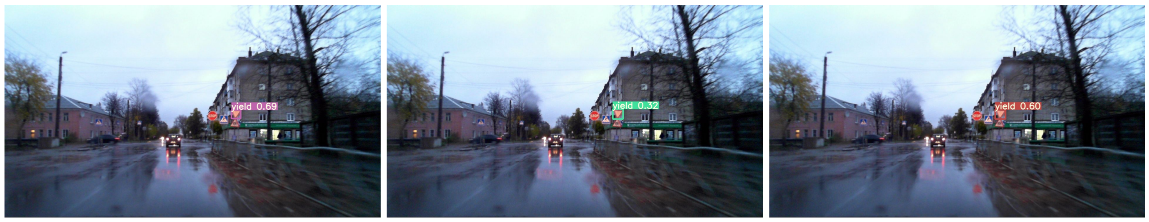

Figure 4. On the left, we have the detection result on RTSD from task1, in the middle we have the detection results from task2(transfer learning) and on the right we have detection results from task3(loss function modification)

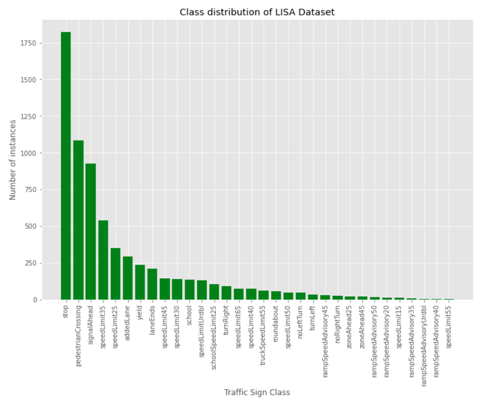

Figure 5. LISA data distribution

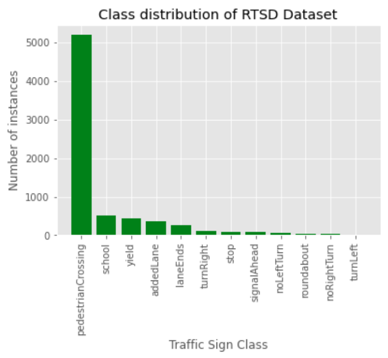

Figure 6. RTSD data distribution

improved as shown in Figure 4. The confidence score on traffic signs was increased as compared to the model from task 2 but not as strong as task 1. This shows we were somewhat able to transfer learning by reducing catastrophic forgetting for the RTSD model but not fully eliminating it.

4. Implementation

In this project, we used two codebases and developed on top of them. The Yolov3 model comes from [4] and the GradCAM comes from [3]. For YOLOv3 implementation, we didn't make many changes to the original codebase. We made the custom configuration files along with configuration for class names and pre-setted our dataset in the format required by the repository. For GradCAM, the initial code was breaking at multiple places and we changed the codebase to get it working with Pytorch 1.4.

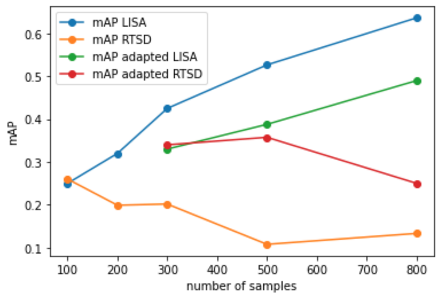

Figure 7. The plot shows change in mAP for LISA and RTSD before and after using the modified loss function on YOLOv3 model. For this, we used the model trained on RTSD data and fine-tuned it only using LISA dataset images. The X-axis represents the number of samples taken for fine-tuning and Y-axis has the mAP value on validation data.

References

[1] Lisa dataset. http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html.

[2] Rtsd dataset. https://graphics.cs.msu.ru/en/research/projects/rtsd.

[3] Gradcam. https://github.com/pifalken/YOLOv3-GradCAM, 2014.

[4] Yolov3 ultralytics. https://github.com/ultralytics/yolov3/tree/archive, 2014.

[5] JRedmon. Yolov3 architecture. https://arxiv.org/pdf/1804.02767.pdf.

[6] A. Kuthuria. yolov3 arch. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b.