Effects of Multi-headed Self Attention on Object Detectors

1st Samuel Okhuegbe
Department of Energy Science and Engineering
University of Tennessee
Knoxville, United States
sokhuegb@vols.utk.edu

2nd Vijaysrinivas Rajagopal
Department of Electrical Engineering and Computer Science
University of Tennessee
Knoxville, United States
vrajago2@vols.utk.edu

***Abstract*—Object Detection is one of the main challenges in computer vision. Convolutional Neural Networks (CNN) have been used traditionally to solve image classification and object detection problems. To further improve performance, integrating CNN with Transformers, specifically Multi-head self attantions (MSA), is a prominent area to explore. In this paper we investigate the effects of integrating MSA with the YOLOv3 object detector. A new network YOLOv3 MSA is developed and applied to the Oxford IIT Pet and Visual Objects Challenge (VOC) dataset. The effects of augmentation on the YOLOv3 and YOLOv3 MSA was also investigated.**

From the results obtained, MSA modules showed significant improvement in accuracy when applied to a classifier (MSA-Darknet53) but when applied to the object detector (YOLOv3 MSA), the MSA modules marginally reduced the mAP performance.With augmentations removed for all networks, the YOLOv3 MSA had a better mAP performance compared to the traditional YOLOv3. In general, MSA modules showed a reduction in localization and objectness, which further reduced the mAP peformance. We posit this is because MSA do not perform well on non-convex function, and the L2 distance and logistic regression (which form the objective function for localization and objectness) are not guaranteed to be convex.

We suggest that increasing the dimension of the MSA heads would improve the overall mAP performance for object detection. Although this would require large amount of computational power.

***Index Terms*—deep learning,multi-head self attention, transformers, , object detection, YOLOv3**

I. Introduction

With the introduction of Transformers into deep learning, tasks related to natural language processing (NLP) have improved tremendously compared to previous types of networks and architecture. Transformers calculate "self-attention" in order to contextualize the sequence of input data they are given. The Transformer architecture is now being utilized in computer vision (CV) tasks, such as classification and object detection. With the introduction of ViT for image classification [9], Transformers seem to comparable or even better performance compared to convolutional neural networks (CNNs). Naturally, there has also been work done on combining the Transformer's self attention mechanism with convolutional layers.

Particularly, the work done by Park et al. ([9]) has shown a significant improvement in classification accuracy of a blend of convolutional layers and multi-headed self attention (MSA) modules. For residual-based network, [9] empirically established that replacing a residual block with MSA modules at the end of each stage in an alternating fashion helped those networks improve classification accuracy.

In this paper, we extended the work done in [9] to object detection and attempted to apply the same MSA modules toward object detection networks. More specifically, we modified a real-time, single stage object detector called YOLOv3 with differing layouts of MSA modules and observe the detection performance. Through our analysis, we conclude that MSAs **could** increase mAP performance, but directly applying them to any object detection network is not a straight-forward process. Our experiments show that the majority of the YOLOv3 MSA configurations **decrease** performance. We attribute that to non-optimal augmentations, non-optimal hyperparameters within the MSA modules, and non-convex loss surfaces for localization-based functions that are present in YOLOv3.

II. Previous Work

With the success of Transformers in natural language processing, various researchers have attempted to apply Transformers to image classification. Transformer are based on self-attentions that enable learning long range relationships.

Researchers in [2] where the first to show how transformers can arguably replace traditional CNNs for image classification tasks. This architecture called Vision Transformer (ViT) was used to classify a large dataset. The Transformer involved long computational time. Although the vision transformer performed well on large datasets, it does not perform well on medium/small sized datasets when compared with CNN. The authors of ViT posited that the reason was because CNNs have some form of inductive bias that reduces the need for large datasets. To address the issues with ViT, researchers in [12] developed a Data-efficient image Transformers (DeiT) that performed well on medium sized datasets and also had shorter training time. Transformers have also been combined with CNN to improve image classification accuracy. LeVit was developed by researchers in [6], LeVit improves local feature modeling in ViTs by adding convolution blocks.

With the success of combining of Transformers with CNNs for image classification, a logical next step would be to

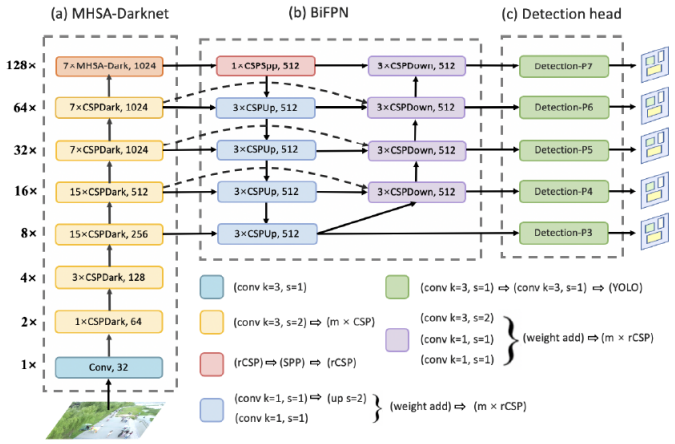

Fig. 1. **ViT YOLOV4 entire network structure**, (a) Adds transformers to CSP-Darknet backbone. (b)Weighted Bi-directional Feature Pyramid Network (BiFPN) aggregates features from different backbone levels for different detector levels (c) Yolov4 detection head to predict the boxes [13]

investigate how well this approach would work for object detection. One key inspiration for this work is from research in [9], where they explored in details how Transformers work for vision task. They were able to affirm that Transformer success is not just because of its ability to capture long dependencies. The researchers also highlight a key differences between convolutional blocks and MSAs. They assert that convolutional (Conv) blocks are "higher-pass" filters while MSA blocks are "low-pass" filters. The research further introduces some framework on how to combine MSA's with Conv blocks, and they showed over 5% improvement on Imagenet dataset. The architecture developed was called AlterNet, which basically combines ResNet50 with MSA blocks. In this paper the frameworks used by [9] for adding MSA's to Conv would be replicated for object detection using YOLOV3 and MSA's.

Some researchers have attempted similar tasks. Works, such as [13], developed a Transformer-based YOLO (ViT-YOLO) network for object detection. Here a Vision Transformer was combined with YOLOV4. The architecture replaced a late Cross Stage Partial (CSP) block in with 128 large Multi-Head Self-attention Darknet (MHSA-Darknet) containing 1024 heads, a Weighted Bi-directional Feature Pyramid Network (BiFPN) was used to aggregate features from the previous backbone levels and finally a YOLOV4 head for the actual prediction. The network was only trained and analyzed on a large drone-dataset (VisDrone). Fig 1, shows the structure of the ViT-YOLO network.

In our work, we use a different framework for combining MSA's with YOLOV3 as outlined in [9], and we also evaluate the network on multiple datasets. Also the ViT-YOLO requires more computational power.

III. Technical Approach

A. Methodology

In order to address the main question of how MSA modules affect the performance of object detectors, we decided to take an incremental approach.

1) Experimental Approach: The first step of seeing the viability of adding MSAs to YOLOv3 was to partially replicate the improvements seen in [9] on the classification portion of the object detector. We took the feature extraction network and applied MSA modules to different parts of the network and compare both versions' Top-1 accuracy. If we were able to see notable improvements with the MSA, then applying it to YOLOv3 would be viable.

The second step was simply adding the same MSA module configuration (and derivative configurations) to YOLOv3. If we were able to see notable improvement in detection performance, then we can conclude that MSA modules are beneficial for object detection. Additionally, we can also conclude that the build-up rules defined in [9] applied to both classification **and** object detection. If we do not see any notable improvements, we continue applying different MSA configurations and identifying other causes to the lack of performance improvement.

2) Constraints: For fair comparisons, all hyperparameters directly not related to the primary modification would stay constant. This includes hyperparameters such as learning rate, learning scheduling, and image augmentations. We also kept the number of epochs and batch-sizes the same to rule out any influence they may have on convergence speed.

3) Metrics: For comparing effects on classification performance, we do a simple accuracy measurement where we calculate the fraction of correct prediction labels ("Top-1" implies we only considered the most confident prediction label) over the total number of predictions.

For comparing effects on object detection "performance", we track mean average precision (mAP) at an IoU threshold of 0.5 (denoted as mAP @ 0.5) of each YOLOv3 network. The mAP is a number from 0 to 1 that is computed from the area-under-the-curve (AUC) of a Precision and Recall graph. Precision, Recall, and mAP are all defined in equations 2, 3, and 1 respectively. The relative definitions of True Positive (TP), False Positive (FP), and False Negative (FN) are defined as the following:

- **TP** - A detection box that has a minimum shared area ratio - also known as intersection over union (IoU) - with a given ground truth as well as the same class.
- **FP** - When a given detection box has no matched ground truth box
- **FN** - All of the leftover, unmatched ground truth boxes

$$AP = \int_1^0 p(r)\, dr \tag{1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

B. Object Detector Structure

The object detector utilized in this work is an implementation of YOLOv3 by Ultralytics. There is no significant differentiation between the original C++ implementation of YOLOv3 and Ultralytics's implementation other than training procedures, which is later detailed in the "Data Set and Implementation" section. YOLOv3's structure is composed of two main parts - the feature extractor and the prediction head (also referred to as the "YOLO" head) [11].

The feature extractor is a residual-style CNN called Darknet53. This network is detailed in Figure 3 and contains five stages composed of a number of Residual blocks that are then downsized with a Conv block. A Residual block is composed of a 1x1 Conv block and a 3x3 Conv layer, whose input is summed with the original Residual block input [11]. Likewise, a Conv block is defined as a Convolutional layer with a Batch-normalization layer and ended with an activation layer (Leaky-ReLU).

C. MSA Structure

The MSA module we are utilizing is the same one from [9]. It is composed of a 1x1 Conv layer that reduces the dimensionality of the incoming feature maps to the dimensionality of the MSA. The attention module is very similar to the attention in [2], where "local" self-attention is applied to each patch of the input with the positional embedding simply being the index location of the patch. Since the MSAs are applied at the end of each stage, another 1x1 Conv layer is applied to the MSA's output to increase the channel depth back to the original input's channel value.

The MSA module also comes with a set of hyperparameters, which include patch size (window size), number of heads, input dimension size, and head dimension size. We kept the input size of YOLOv3 at 416x416, which means that the patch size had to be 13x13 because it needs to be divisible by the input shape. The number of heads were respectively 3,6,12, and 24 as we get further down the network (see Figure 5).

D. Applying MSA Modules

As per [9], we apply MSA modules at the end of each stage of Darknet53, except for the very first stage. The reason for this is because [9] found that MSA modules in the earlier stages of a network hurt classification performance. For our initial YOLOv3 MSA structure (detailed in 4), we reference AlterNet's final structure. Similar to the original paper, we also remove and relocate different MSA modules to see the effect on the performance of the network.

E. Preprocessing Methods

1) Classification: For our classification-based experiments, we utilize the following image augmentations:

- Image Normalization
- Random Crop
- Random Translation
- Random Horizontal Flip
- Random Augment (PyTorch function to automate random combination of hue, saturation, and brightness augmentations)
- Random Erase (Figure 6).

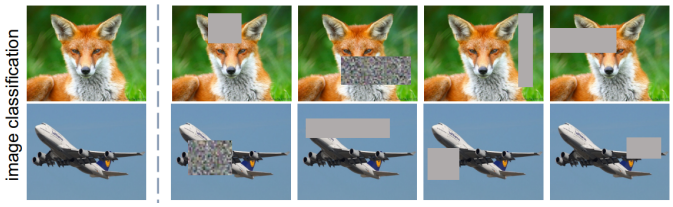

Fig. 6. **Illustration of RandomErase being applied to images for classification purposes.** RandomErase, as the name states, randomly removes portions of a given image in an attempt to increase the range of image features a network can use for its ultimate task. In [14], this was for image classification and person re-identification. Credit for illustration belongs to [14]

We utilize "Stochastic Depth" as the only training-time network function. Stochastic Depth randomly drops a given residual connection from a Residual block for a given run [4]. This training technique is also used by [9].

2) Object Detection: For our object detection experiments, we preserved the majority of the original Ultralytic YOLOv3 training paradigms. In terms of preprocessing, this included the following image augmentations:

- Image Normalization
- Random Translation
- Random Hue Change
- Random Saturation Change
- Random Brightness Change
- Random Horizontal Flip
- Letterbox (padding square input to preserve aspect ratio)
- Mosaic Image Function (Figure 7)

Fig. 7. **Example of Mosaic preprocessing.** As defined in [1], mosaic is supposed to reduce the likelihood of a CNN learning objects based on background or non-object based features.

Additionally, the training paradigm included Exponential Moving Average (EMA), which a given optimizer uses to reduce noisy gradients during training.

IV. Data Set and Implementation

A. Data Sets

For this work, we utilize two main object detection datasets and one classification dataset. For the object detection task, we trained YOLOv3 on Oxford IIT Pet dataset and Visual Objects Challenge (VOC) dataset.

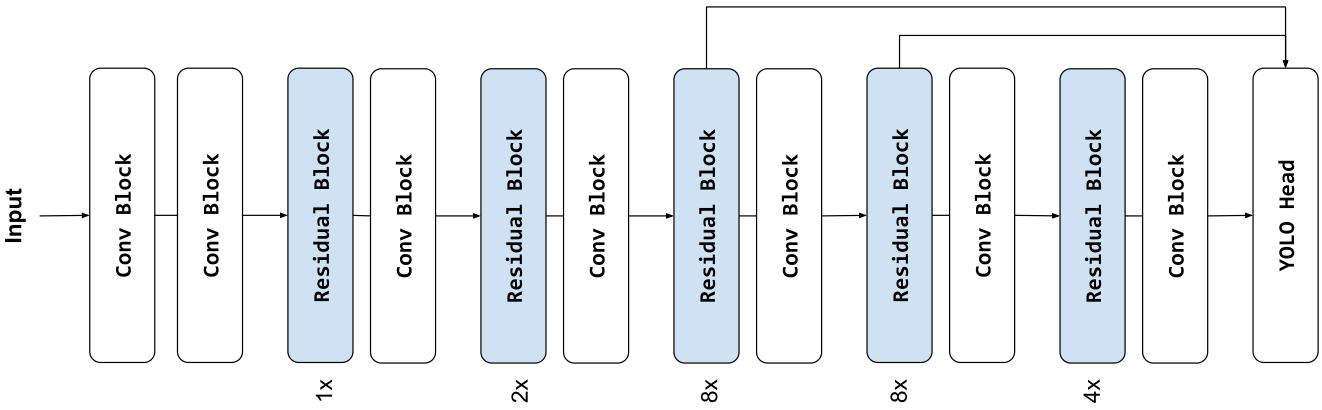

Fig. 2. **General structure of YOLOv3.** The network is composed of a feature extractor called Darknet53. The feature map output from the feature extractor is actually composed of three different feature maps. Each feature maps are from different parts of the feature extractor and are supposed to represent objects at different "scales". Each feature map scale is processed in parallel in the YOLO head. The output of the network is three different sets of predictions.

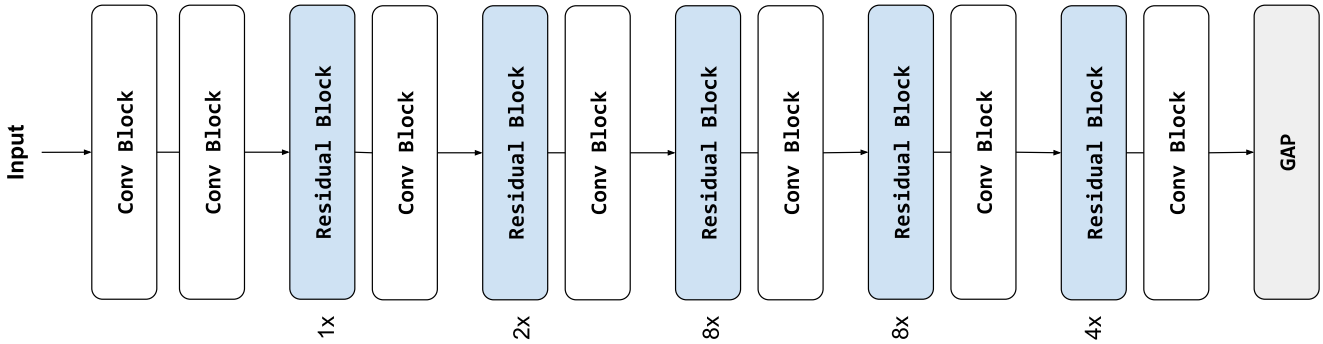

Fig. 3. **Structure of Darknet53 as a classifier.** The classifier reduces the feature map output from the Residual structure into a vector of class probabilities with Global Average Pooling (GAP). This allows the CNN to have flexible input sizes while maintaining the consistent class-based probability output.

1) Oxford IIT Pet Dataset: The Oxford IIT Pet dataset is composed of "37 [pet categories] with roughly 200 images for each class" [10]. The annotation style is visualized in Figure 8, and a given method should output either the localization of the head or segmentation of the whole pet and the classification of the specific pet breed.

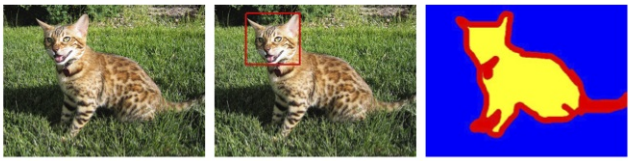

Fig. 8. **Illustration of annotations provided in the Oxford IIT Pet dataset.** Each image is of one pet, and for our purposes, we focused on localizing the head of the animal and classifying the animal by breed. This annotation diagram was taken directly from [10]

2) Visual Object Challenge Dataset: The VOC datasets are a set of multiple datasets based on various years of the challenges. In terms of literature importance, VOC2007 and VOC2012 were the most utilized datasets among the VOC family, which is why we are using them in our work. VOC2007 is composed of 20 classes with a total of 4,952 training images and 2,501 testing images.

Similarly, VOC2012 contains the same 20 classes but contains a total of 5,171 training images and 5,823 validation images. From Figure 9, the annotation style is similar to the Oxford IIT Pet dataset, where the objects are labeled with a bounding box and each box is assigned one of the 20 classes. In this work, we combine VOC2007 and VOC2012 training set and test on the VOC2007 test set.

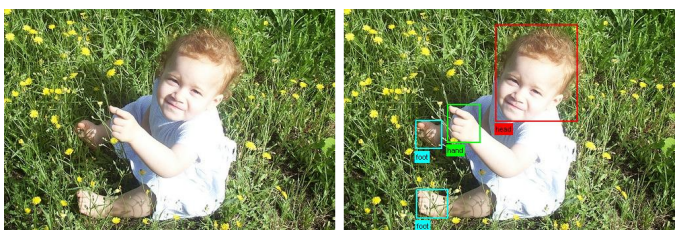

Fig. 9. **Illustration of annotations provided in VOC2007 dataset.** The dataset is composed of bounding boxes and segmentation masks. For our purposes, we are focusing on the bounding box. The image is taken from [3]

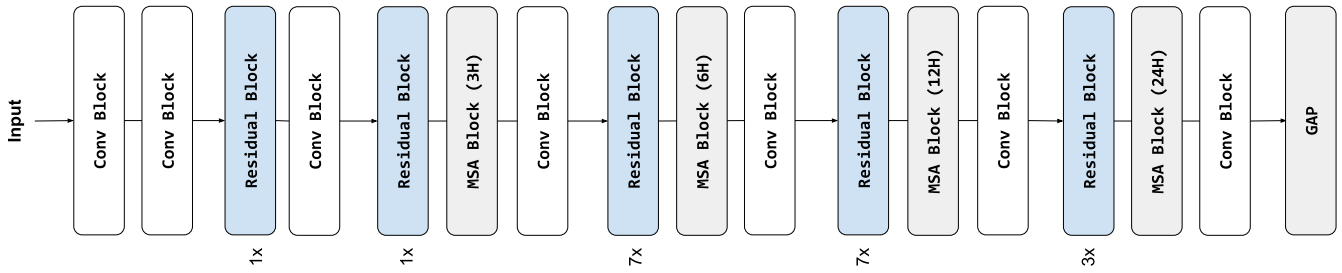

Fig. 4. **The proposed Darknet53 network with an addition of MSA modules.** Taking the same placements as AlterNet, we replace Darknet53's last Residual block from the 2nd stage to the last stage with an MSA module.

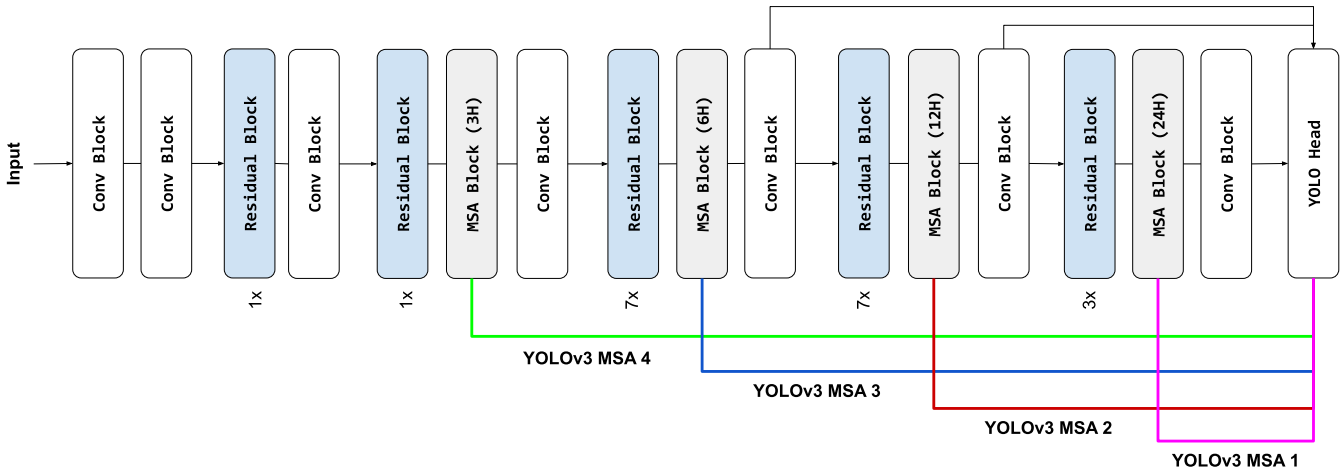

Fig. 5. **The proposed MSA configurations applied to YOLOv3.** Similar to the different versions of AlterNet, we vary the amount of MSA modules and their placement. The notation surrounded by parenthesis represents the number of heads in each MSA "block" (or module). In guidance with [9]'s findings, we increase our head number as we get further along in the network.

3) CIFAR-10: CIFAR-10 is a popular and very simple dataset that contains 10 classes. Each class is a generic object or subject, and the dataset is composed of 60,000 32x32 images [5].

B. Classification Training Procedures

For classification training, we utilized Adam as our primary optimization method due to an observed pattern quicker loss convergence. We trained for 350 epochs with a learning rate of 0.0005 and a Cosine Annealing learning rate scheduler (detailed in Figure 10). Both networks were trained with a batchsize of 1024, and the input size was 32 by 32.

Both networks used Categorical Cross Entropy (CCE) for their loss function. We take the model with the best validation performance and display the test results.

C. Object Detection Training Procedures

For object detection training, we kept in-line with the default training procedures. YOLOv3 loss function is a multi-task loss function composed of three different losses. Each of the losses and their descriptions are as follows:

- **Classification** - Applying CCE with the prediction and ground truth labels for corresponding boxes
- **Coordinate Localization** - The L2 distance between a matched prediction box coordinates and its corresponding ground truth box's coordinates
- **Objectness** - A logistic regression using BCE to determine if a given area of an image contains an object or not

For Oxford IIT Pet dataset, we trained each network for 350 epochs at a learning rate of 0.0001 with a batchsize of 16. For VOC, we trained each network for 56 epochs at a learning rate of 0.0001. Both training sets are using the same learning rate scheduler for all three loss functions. This scheduler is detailed in Figure 10. Both networks had an input size of 416 by 416.

Both Oxford and VOC training took around 7 hours to complete on a workstation with a AMD Ryzen 2700X and NVIDIA 2070 Super.

V. Experiments and Results Analysis

A. Classification Results

As mentioned in the Implementation section, we implement Darknet53 as a classifier and add MSA modifications seen in Figure 4. We trained on CIFAR-10 and see a notable

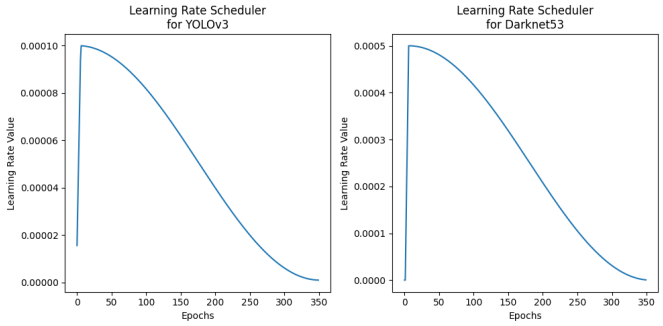

Fig. 10. **Standard learning rate for YOLOv3-ultralytics and Darknet53.** The scheduler starts with a near-zero value that increases for an x number of epochs until it reaches the initially set value. This is referred to as a "warm-up" phase, and after this phase is done, the scheduler reduces the learning rate each epoch with a "Cosine Annealing" equation.

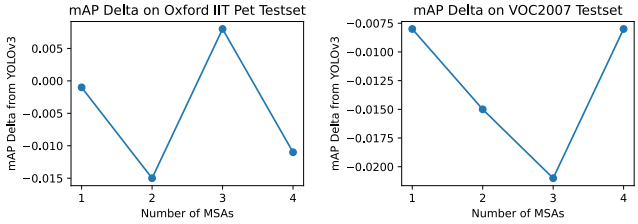

Fig. 11. **The mAP deltas for YOLOv3 MSAs on Oxford IIT Pet testset and VOC2007 testset. The number of MSA modules is not always correlated with the loss or gain of performance in this particular network.**

improvement of 1.23% on Darknet53 with MSA compared to the original Darknet53 network.

TABLE I
CLASSIFICATION ACCURACY OF DARKNET53 AND DARKNET53 WITH MSA AS DEFINED IN 4 ON THE CIFAR-10 TESTSET.

Model Type	Top-1 Accuracy
Darknet53	0.8803
Darknet53 with MSA	**0.8926 (+0.0123)**

B. Object Detection Results

TABLE II
MAP PERFORMANCE OF YOLOV3 VS YOLOV3 WITH MSA MODIFICATIONS ON THE OXFORD PET TESTSET.

Model Type	Oxford IIT Pet testset mAP @ 0.5	VOC2007 testset mAP @ 0.5
YOLOv3	0.92	**0.747**
YOLOv3 MSA 4	0.909 (-0.011)	0.739 (-0.008)
YOLOv3 MSA 3	**0.928 (+0.008)**	0.726 (-0.021)
YOLOv3 MSA 2	0.905 (-0.015)	0.732 (-0.015)
YOLOv3 MSA 1	0.919 (-0.001)	0.739 (-0.008)

VI. CONCLUSION

From our experiments, we conclude the following:

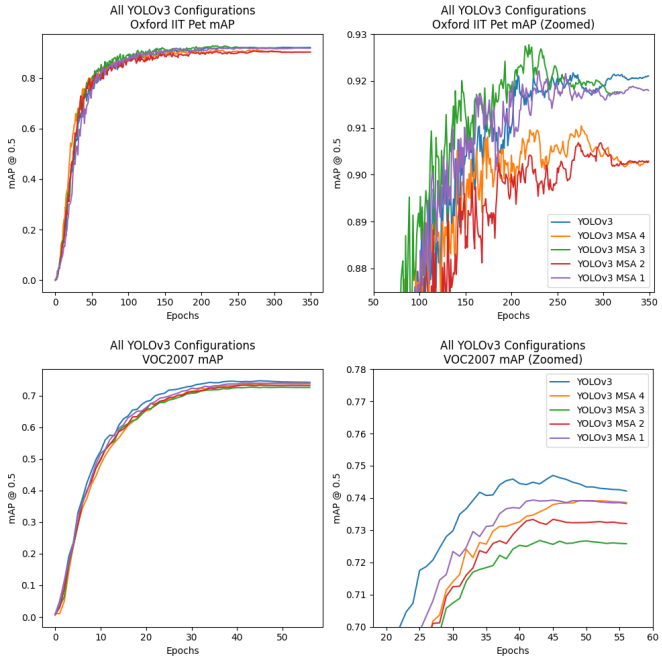

Fig. 12. **The mAP trend during training for Oxford IIT Pet testset and VOC2007 testset.**

TABLE III
TABLE COMPARING MAP PERFORMANCE OF YOLOV3 AND YOLOV3 MSA 4 WITH VARYING LEVELS OF AUGMENTATIONS

Model Type + Removed Augmentation(s)	VOC2007 testset mAP @ 0.5
YOLOv3 (No Augmentations)	0.4564
YOLOv3 MSA 4 (No Augmentations)	**0.4871 (+0.0307)**
YOLOv3 (No Mosaic)	**0.7105**
YOLOv3 MSA 4 (No Mosaic)	0.6859 (-0.0246)
YOLOv3 (No Mosaic + No EMA)	**0.706**
YOLOv3 MSA 4 (No Mosaic + No EMA)	0.6873 (-0.0187)

- When applied to classifiers, MSA modules make a notable improvement in classification accuracy (reaffirming work done by [9])
- When applied to YOLOv3, MSA modules marginally **decrease** mAP performance
- In some cases, MSA modules **can** improve mAP performance, but these improvements are not in alignment with the number of MSAs present in the network and are marginal
- In terms of lost mAP, the number of MSA modules did not always correlate to how much mAP was lost.
- Training trends show that YOLOv3 MSAs with any number of MSA heads increase localization and objectness loss
- The more MSA heads a network has, the greater the objectness loss

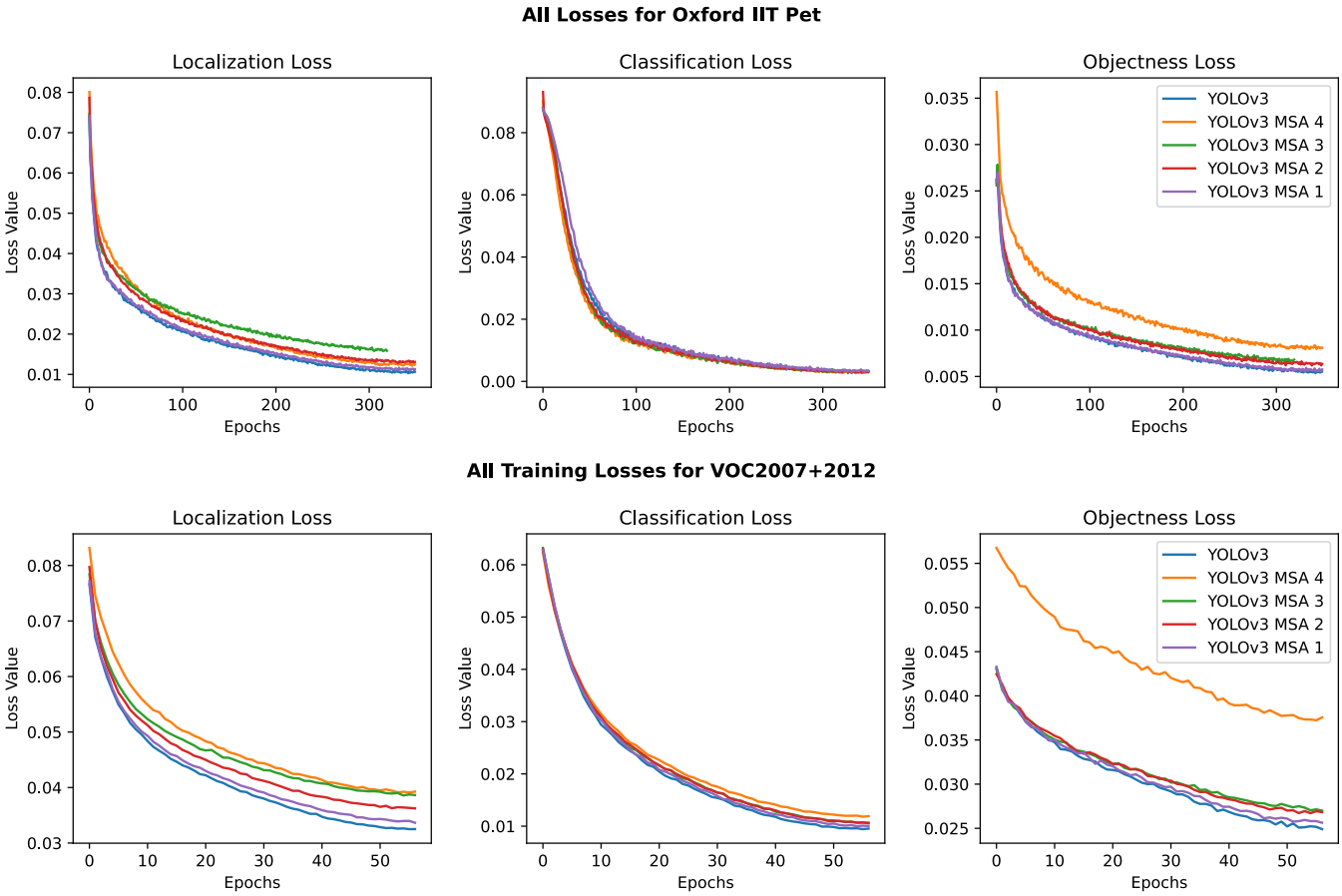

Fig. 13. **Loss graphs for all three components of the YOLOv3 loss function.** It is clear that YOLOv3 MSA configurations suffer significantly under the Objectness and Localization loss. We hypothesize that this is because both are not convex functions, and MSA are not well suited for optimizing in non-convex functions. Although, it is notable to consider that this increasing loss function is seen more starkly in VOC2007. This could be attributed to the more diverse set of objects learned, the greater number of images, and the more dense and diverse settings.

A. *Additional Conclusions*

1) Bad Localization: From the experimental results, we can see that MSA modules had an outsized **negative** affect on the localization and objectness portions of the loss function.

But why? From [9], MSAs improve network metric performance by ”smoothing” the loss surface of a convex function. If the function is non-convex, MSA does not perform as well as a Conv layer. Unfortunately, L2 distance and logistic regression (even with a convex loss function) are not **guaranteed** to be convex [8]. From Figure 13, it seems that both loss surfaces for the tested datasets (particularly for VOC) were not convex.

2) Training Procedures Matter: As stated in the list above, we did not see any notable improvements with MSA, and so we decided to investigate further into **why**. It is well-established that augmentations can drastically help with mAP performance, so we decided to investigate if a set of augmentations were holding back the MSA network rather than helping. Firstly, we removed **ALL** augmentations and trained YOLOv3 and YOLOv3 MSA 4. In this case, YOLOv3 MSA 4 outperformed YOLOv3 by 3% mAP.

This is the biggest change in mAP recorded in this work thus far. Although the overall mAP for both networks is drastically lower than with any augmentations, it gave credence to our hypothesis that MSA-based networks were being negatively affected by a non-optimal set of augmentations. We then tried training the same networks without the Mosaic augmentation, which has helped CNN performance but had no conclusive effect on Transformer-based networks. Based on [7], EMA can negatively affect Transformer learning, so we also ran a training without Mosaic and EMA.

All the results are located in Table III, but removal of these two augmentations still showed the MSA network lagging behind the CNN network. In this case, the mAP delta is greater than any of the deltas in Table II. Although, we affirm that EMA did have a negative effect on the MSA.

3) MSA Head Dimensions: From our Technical Approach section, we stated that each MSA module has a dimension length for a given head. We had set all the heads to a dimension of 64 to fit the models on the present hardware. From our literature review, networks such as ViT YOLO have their head dimensionality at 1024. A possible way of increasing performance would be to increase the head dimensions.

It is clear that conventional CNN networks can benefit from the addition of MSAs. But from our experiments and conclusions, it comes with a bold asterisk. Regardless, we see the future of object detection incorporating a multitude of different layers to increase true scene interpretation.

VII. Code Design

Our code is available on GitHub: https://github.com/vjsrinivas/cs525-final and contains all the code needed to re-run the experiments described in this paper.

It is structured in the following:

- models (contains classifier and detectors)
- classifier (the Darknet53 models with and without MSAs)
- detectors (contains the yolov3 folder)
- yolov3 (all the Ultralytics YOLOv3 GitHub repo stripped down)

We use the YOLOv3 code from Ultralytics as stated previously. You can find it here: https://github.com/ultralytics/yolov3. We also borrow code for the Darknet53 network from https://github.com/developer0hye/PyTorch-Darknet53.

We heavily modified to include parsing for MSAs and programming the networks to utilize the MSA modules. We also implemented the options to turn on/off different augmentations as well as graphing functions that are displayed in the figures above. Finally, we wrote the entire training loop for the Darknet53 and Darknet53 MSA classification.

VIII. Workload Distribution

A. Samuel Okhuegbe

- Worked on literature review
- Setup on Google Cloud machines to run results related to effects of augmentations.

B. Vijaysrinivas Rajagopal

- Applied MSAs to YOLOv3
- Trained initial set (Table II) on hardware

References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[4] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

[5] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[6] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*, 2021.

[7] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[8] Gabriele De Luca. Why does the cost function of logistic regression have a logarithmic expression?, Jul 2020.

[9] Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

[10] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[12] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[13] Zixiao Zhang, Xiaoqiang Lu, Guojin Cao, Yuting Yang, Licheng Jiao, and Fang Liu. Vit-yolo: Transformer-based yolo for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2799–2808, 2021.

[14] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation, 2017.