# *Team True Detectors*
# Investigating Multi-Object Detection

**Payal Agarwal** A53268036 | **Rohit Kumar** A53238243 | **Shreya Nayak** A52371298 | **Siddharth Singh** A53250934

## Abstract

Multi-object detection is one of the most important and challenging branches of computer vision. In this project, we perform an extensive analysis of YOLOv3 and implement some of most recent techniques for training a deep neural network for YOLOv3 like deformable convolutions, group normalization, and anti-aliased networks. Also, we provide experimental results on CenterNet which unlike YOLOv3 is an anchor-free object detection algorithm. For YOLOv3 we get improvement in classification loss (2.5%) by making it anti-aliased, and in mAP (0.8%) by using deformable convolutions. In CenterNet (ResNET-101 backbone) we get an improvement in mAP (1.5%) using deformable convolutions.

## 1 Introduction

Multi-object detection is an essential ingredient in multitude of computer vision tasks like instance segmentation, pose estimation, tracking and action recognition. It has commercial value in surveillance, autonomous driving, visual question answering and UAV applications. Pre-existing domain-specific image object detectors usually can be divided into two categories i.e. two-stage detector like Faster R-CNN [7] and one-stage detector like YOLOv3 [6]. Recently, anchor-free based detection methods [12] have gained prominence in academia as well as industry by the virtue of overcoming many shortcomings of anchor-based object detectors which we will delve upon in detail in section 3. In our paper we studied two recent architectures for object detection- YOLOv3 and CenterNet.

For all our experiments, we have used PASCAL VOC dataset [2] for our experiments. This dataset consists of 20 classes and we have used VOC 2012/2007 trainval as training images (total 16551) and VOC 2007 test as validation images (total 4952). code is available here: https://github.com/siddharths16/ECE-285-Project-YOLOv3.

Our baseline YOLOv3 model achieves mAP of **0.627** and **0.667** while using a confidence threshold of 0.1 and 0.01 respectively. Our baseline CenterNet (with ResNet backbone) achieves **0.753**.

## 2 YOLOv3 : Anchor based object detection

YOLOv3 is a fully-convolutional anchor based single-stage object detector. It divides the image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for the detecting that object. Each grid cell predicts $B$ bounding boxes. For each bounding box it predicts four coordinates, objectness score and confidence score for each class. In YOLOv3, the detection is done by applying $1 \times 1$ detection kernels on feature maps of three different sizes at three different places in the network, which are referred to as YOLO layers.

One of the main drawbacks of YOLOv2 was its inability to detect objects of smaller scales. To counteract this, YOLOv3 makes prediction at three scales, which are precisely given by downsampling the dimensions of the input image by 32, 16 and 8 respectively. Due to which YOLOv3 shows improved

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

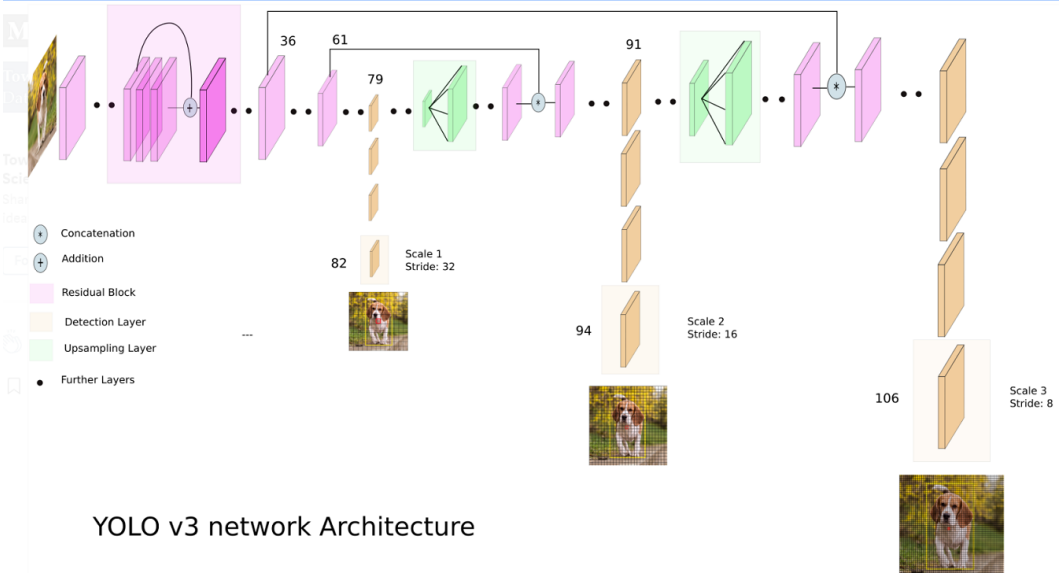



Figure 1: The detection is done on feature maps of three different sizes at three different places in the network. [3]

performance at reduced scale. YOLOv3 has also been made deeper by using 53 convolutional layers compared to 19 layers in YOLOv2.

In conclusion, YOLOv3 performs on par with most other single-stage detectors and most two-stage detectors while being three times faster.

### 2.1 Evaluation Metric: Generalized Intersection over Union (GIoU)[8]

Object detection consists of two tasks: localization, which is determining the location of the object in an image, and classification, which is assigning a class to the object. One of the most popular evaluation metric in object detection is Intersection over Union (IoU). It has the property of being scale invariant.

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{|I|}{|U|} \tag{1}$$

Where $A$ and $B$ are the prediction and ground truth bounding boxes respectively.

Hamid Rezatofighi et al. [8] shows that there is not a strong correlation between minimizing the cost function and increasing the IoU value. When $|A \cap B| = 0, IoU = 0$, IoU does not reflect if the two shapes are in vicinity of each other or are far away.

GIoU extends on IoU by using the smallest possible convex shape $C$ enclosing both $A$ and $B$ and taking the ratio of area occupied by $C$ excluding $A$ and $B$ with the total area of $C$. This ratio is then subtracted from the value of IoU to give us GIoU.

$$GIoU = IoU - \frac{Area\ in\ C\ excluding\ A \cup B}{Total\ area\ in\ C} \tag{2}$$

Which results in IoU becoming the upper bound for GIoU, i.e. $IoU(A, B) \geq GIoU(A, B)$. This bound becomes tighter as $A$ and $B$ have stronger shape similarity and proximity.

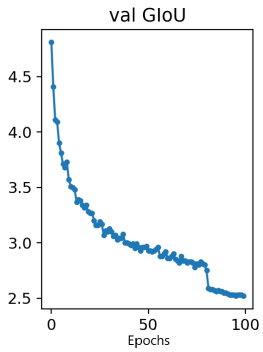



Figure 2: GIoU loss across epochs for baseline YOLOv3: GIoU loss reaches 2.52.

### 2.2 Deformable Convolution

A key challenge in visual recognition is how to accommodate geometric variations in object scale, pose, viewpoint and part deformation. In current CNN architectures, internal mechanisms to han-

dle geometric transformations are lacking which causes problems. Ex: the receptive field sizes of all the activation units in the same CNN layer are the same which is undesirable. Because different locations may correspond to objects with different scales or deformation, adaptive determination of scales or receptive field sizes is desirable for visual recognition with fine localization.

Deformable Convolutions [1] adds 2D offsets to the regular grid sampling locations in the standard convolution. These offsets are learnt via additional convolutional layers. During training, the convolutional kernels for generating output features and the offsets are learned simultaneously.

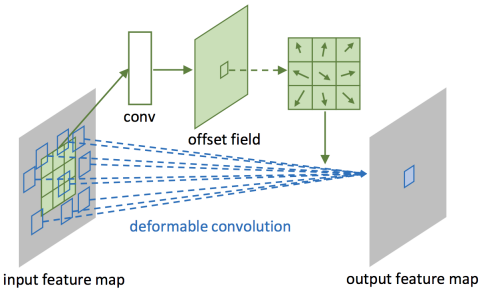


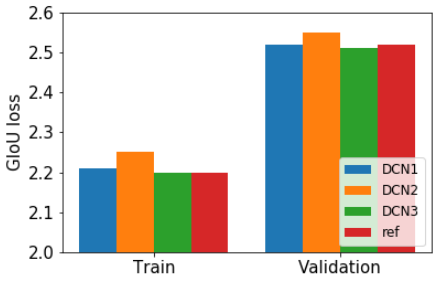

Figure 3: A 3 x 3 deformable convolution

For all the experiments, we applied `defConv` on the last layers, as suggested in the paper. The intention here is to correct the alignment problem i.e., for a cell on the feature maps close to the output, its projected spatial location on the image is not aligned with the location of its receptive field center. We performed the following experiments using `defConv`:

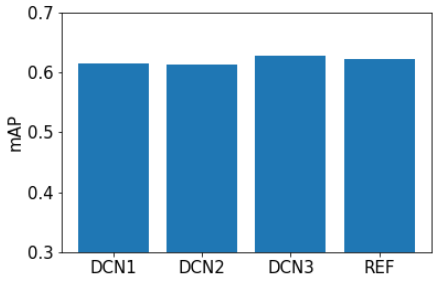
- DCN1: We change last 3 convolutional layers before the last YOLO layer to `defConv`.
- DCN2 : We change the last 3 convolutional layers with kernel size =3, before all the 3 YOLO layers to `defConv` layers.
- DCN3 : We change the last 3 convolutional layers, before all 3 YOLO layers to `defConv` layers.

In DCN2, we did not change the convolutional layers with kernel size 1 to `defConv` layers since it is meant to learn cross channel variance, and not impact receptive field. However, it turns out that DCN3 gives highest mAP (67.5%) and has lowest validation GIoU loss. While DCN2 has highest GIoU loss. So, mixing `defConv` and convolutional layers degrades results.

DCN3 improves overall mAP and GIoU in both train and validation datasets. Thus, using `defConv` layers before all 3 yolo layers improve results over the baseline. We ran our experiments for 100 epochs and get improvement in mAP of $0.8\%$.




(a) Train and validation GiOU loss




(b) mAP

Figure 4: Adding Deformable-Convolution layer to YOLOv3. DCN3 has a 2.2% improvement over

### 2.3 Anti-Aliased Networks

Modern convolutional neural networks (CNNs) are not shift invariant (they are shift equivariant) [11]. Even small input shifts or translations can change the result drastically. The reason CNNs lose their shift invariance is due to layers used for downsampling like max-pooling, strided convolutions (convolutions with stride greater than 1) and average pooling. These operations are naive methods for subsampling , which without a preceding low pass filter ( Fig 5) can lead to the high frequency components and get aliased into low frequency components.

Richard Zhang [11] proposed an anti-aliasing filter (`BlurPool`) for classification networks which complements the above mentioned downsampling operations to make them shift invariant. `BlurPool` is a combination of blurring and subsampling operation. Although [11] is for classification networks, we were interested in its performance with object detection networks.


Figure 5: Image processing perspective to downsampling: image must be first anti-aliased so that high frequency components don't appear as low frequency components. This operation is then followed by a downsampling operation.[5]

### 2.3.1 Method

To perform this experiment, we modify our model to contain `BlurPools` as well as add a routine while testing to check mAP with diagonal translation of the test images.

YOLOv3's model for PASCAL-VOC dataset contains 75 convolutional layers. Out of these, 5 layers are strided-convolutional layers (`conv2d(stride>1)`). To anti-alias our model, all strided-convolutions must have stride equal to 1, followed by a `BlurPool` layer which has stride equal to that of the original strided-convolutional layer (Fig 6).

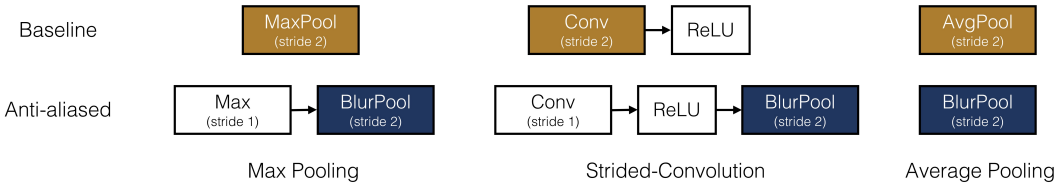


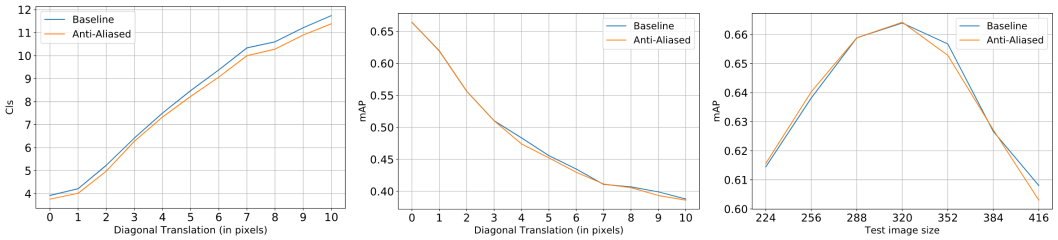

Figure 6: Anti-aliasing of `Conv2d(stride>1)` and other downsampling operations using `BlurPool`

While translating the test images, we make sure that the bounding boxes for the objects in the corresponding image also get translated. We translated those images and bounding boxes in which the bounding boxes don't get clipped upon translation. After passing this basic test, images were diagonally rolled using `torch.roll()` such that image wraps from the top right to bottom left.

### 2.3.2 Results and Inference

As per Fig 7b and 7c, there is no significant change in the mAP of the network by making it anti-aliased. This prompted us to look at the classification loss of the two networks. Fig 7a shows that there is some improvement in classification loss for the anti-aliased model. This could imply that [11] is more suited to classification networks than object detection networks.

Our YOLOv3 network was trained with an image size of 320. Fig 7c shows that the mAP is maximum at a similar test image size and progressively reduces on increasing or decreasing the test image size. mAP is still maintained above 0.6 for $\pm 30\%$ change in scale.




(a) Classification loss, lower is better.

(b) mAP with diagonal translation of test images.

(c) mAP with changing test image sizes

Figure 7: Performance comparison of baseline and anti-aliased YOLOv3.

### 2.4 Group Normalization

During training, the distribution of the learnable parameters changes from one epoch to another. The varied ranges of the parameters over different layers leads to slow convergence and poorer solutions. To achieve better and faster solutions, normalization is done at each hidden layer which involves the following computation:

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i) \tag{3}$$

Here, $x$ is the feature computed by a layer and $i$ is an index. For 2D images $i = (i_N, i_C, i_H, i_W)$ where $N$ is the batch axis, $C$ is the channel axis, $H$ and $W$ are the height and width of the image. $\mu$ and $\sigma$ are the computed mean and standard deviation over the set $S_i$.

In Batch Norm, $S_i$ consists of pixels sharing the same channel index. Thus, it computes $\mu$ and $\sigma$ along the $(N, H, W)$ axis. It is to be noted that in batch normalization, the normalization is done within a minibatch. However, Batch Norm cannot ensure the model accuracy rate when the batch size becomes smaller. As a result, researchers today are normalizing with large batches, which is very memory intensive, and are avoiding using limited memory to explore higher-capacity models. Hence, batch normalization poses a limitation on how small the batch size can be.

To do away with this limitation, Yuxin Wu et al. [9] proposed another normalization technique called Group Normalization. Hence, normalization is along $(H, W)$ axis along a group of $\frac{C}{G}$ channels. Based on experiments done on ImageNet classification data using ResNet models, the claim in [[9]] is that Group Norm performs better than Batch Norm on small batch sizes. Hence, by using Group Norm, ML practitioners can do away with the limitation on the batch size.

We conducted two experiments to check if these claims hold true for YOLOv3.

#### 2.4.1 Experiment 1: Independent of Batch Size

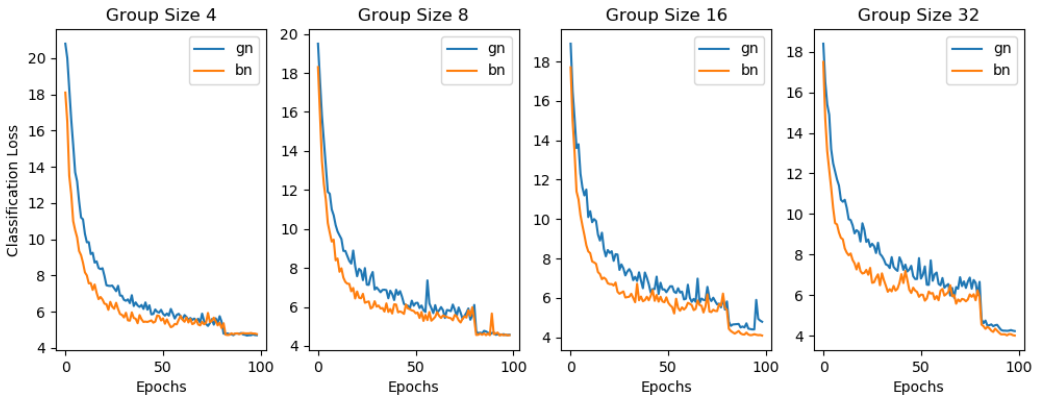



Figure 8: Comparison of classification Loss for Group Norm(gn) and Batch Norm(bn) for various batch size (Image Size 224, VOC 2007-2012)

For Group Norm, the channels are divided into groups and normalization is done within these groups. Consequently, Group Norm shouldn't show much variation on changing batch size. This experiment was done to study the variations in the performance of the network when batch size is varied. Hence, for various batch sizes, both batch norm and group norm were used and the performance after training on images of size 224 for 100 epochs were observed. From the Fig 8 it can be seen that group norm gives worse results for larger batch sizes but for smaller batch sizes, the results are comparable.

#### 2.4.2 Experiment 2: Variation with Group Size

Yuxin Wu et al.[9] claims that there is not much of a difference in mAP with changes in group size. We trained the nets for a constant batch size of 32 and varied the number of groups to 4,8,16 and 32. The plot comparing the results are in Fig 9. mAP for groups 4, 8, 16 and 32 is 0.584, 0.586, 0.585 and 0.588 respectively. Hence, the maximum deviation of mAP with respect to the best (number of groups = 32) is just 0.004. It can be seen that even though the performance with different number of

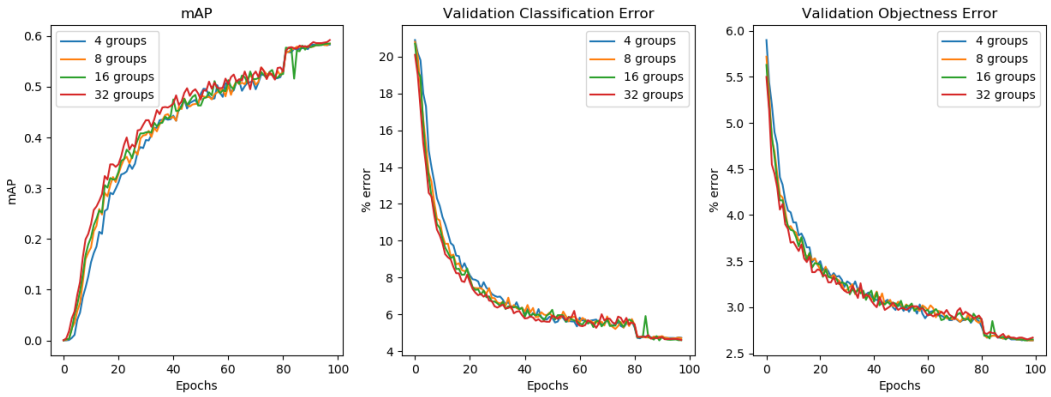



Figure 9: mAP, Classification Error and Validation Error comparison for various number of groups (Image Size 224, VOC 2007-2012)

groups is almost similar, group normalization with 32 groups performs slightly better than the nets with lower number of groups.

### 2.5 Anchor Estimation

Parameter tuning for anchor boxes is an important task for YOLOv3s. If anchor boxes are not tuned correctly, the neural network will face difficulty in detecting objects of differents scales and shapes. The reference YOLOv3 implementation [6] is using same anchor boxes for COCO and VOC datasets (Anchor_Default). In our opinion, this is not a good choice. The reason being, COCO dataset has 80 different objects, while VOC has only 20. The anchor box shapes depend on the type of object to be identified. For example, a car, would usually need a wider anchor box compared to a person. So, we performed experiments to come up with better anchor boxes. We see a 0.8% improvement in mAP and over 20% improvement in gIoU, over baseline using anchor boxes created by Anchor_dIoU experiment discussed later (Fig 11).

#### 2.5.1 Method

We tried two different techniques to generate anchor boxes. In both the techniques, we extract all the bounding boxes for every image in the training set. We then apply K-Means based clustering on these boxes to obtain the clusters. The centroid of each cluster represents an anchor box. For 3 layered YOLOv3 architecture, we compute 9 clusters. The largest 3 boxes are used as anchors for the first YOLO layer and so on.

We use two different metrics when clustering using K-Means. In technique 1 (Anchor_Euclidean), we used standard K-Means with Euclidean distance as metric for clustering. In the other technique (Anchor_dIoU), we use a different distance metric to create clusters which is described in the paper YOLOv2. The distance between a box and a cluster centroid is given by the following equation.

$$d(box, centroid) = 1 - IoU(box, centroid) \tag{4}$$

The box is assigned to the cluster based on $min\ d(box, centroid)$.

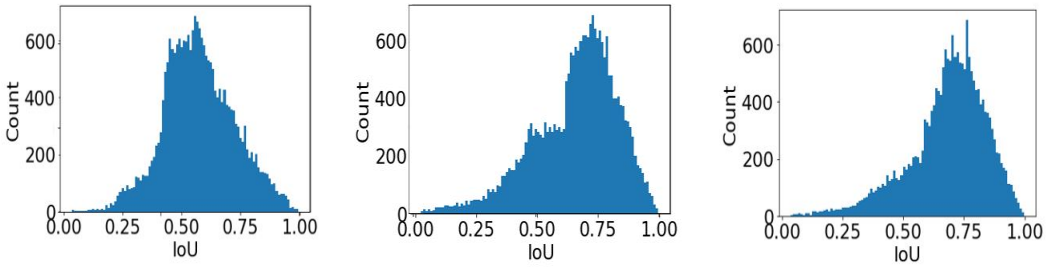
We obtain 9 anchor boxes from each technique. To compare the two methods, we compute the IoU of all the bounding boxes in the training set to the cluster centroid it belongs to. This is a good indicator of the quality of bounding boxes, as the cluster's centroid is representing all the boxes that belong to that cluster.

#### 2.5.2 Result and Inference

Mean IoU is highest when anchor boxes are selected using the second technique. Even the variance is lower. Mean IoU has improved significantly over using the default Anchor boxes available for COCO dataset. Table 1 summarizes the results.

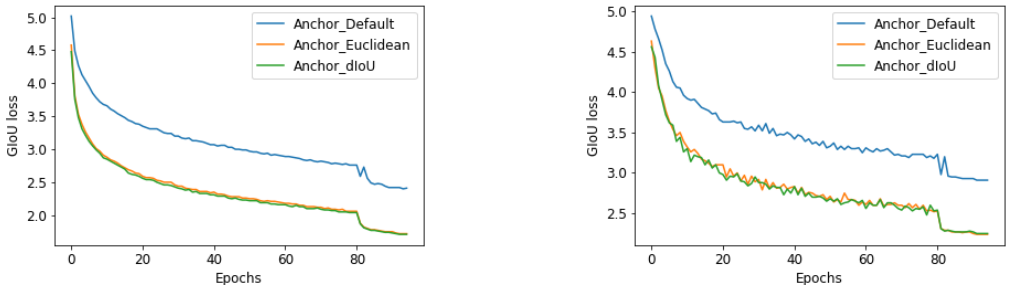
| IoU distribution | Anchor_Default | Anchor_Euclidean | Anchor_dIoU |
|---|---|---|---|
| min | 0.1766 | 0.0313 | 0.0375 |
| max | 0.9976 | 0.9968 | 0.9975 |
| mean | 0.6387 | 0.6624 | 0.6839 |
| variance | 0.0162 | 0.0285 | 0.0246 |

Table 1: IoU distribution of bounding boxes with anchor boxes created using different techniques. Anchor boxes created using IoU in distance metric gives best mean and minimum variance.




(a) Default Anchor Box (b) Euclidean based Anchor Box (c) IoU based Anchor Box

Figure 10: Distribution of IoU of bounding box with corresponding cluster centroid




(a) Training Set (b) Validation Set

Figure 11: GIoU loss comparison using different Anchor boxes. At 100th epoch, both Anchor_Euclidean and Anchor_dIoU gives same training GIoU loss of 1.4, but for validation set, Anchor_dIoU loss is lower by 1%.

### 2.6 Miscellaneous Experiments

#### 2.6.1 Activation Functions

Designing activation functions that enable fast training of accurate deep neural networks is an active area of research. The rectified linear activation function, has eased the training of deep neural networks by alleviating the problems related to weight initialization and vanishing gradient. We are using leaky-ReLU in our YOLOv3 model. This addresses the *dead neuron* issues in the ReLU network.

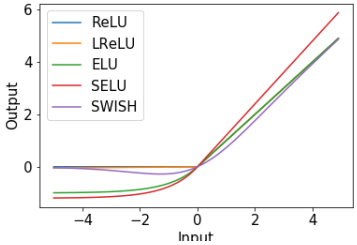
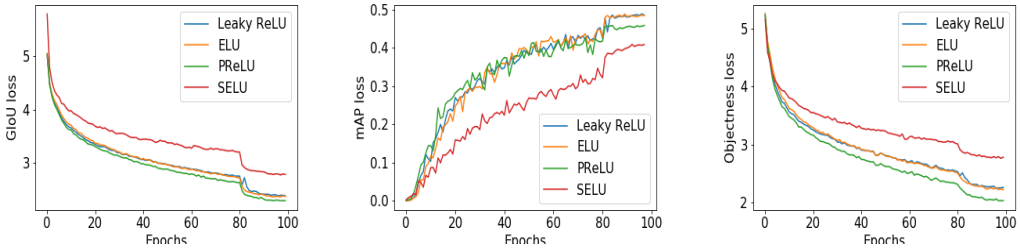



Figure 12: Study of different Activation functions

We used different activation functions like *parametric* ReLU which treats the leakage parameter of LReLU as a per-filter learnable weight. We also used the *exponential* linear unit function (ELU) and the *scaled* exponential linear unit function (SELU) activation functions . Like ReLU, LReLU, and PReLU, ELU reduces the vanishing gradient problem. SELU extends this property ensuring that activations close to zero mean and unit variance that are

propagated through many network layers will converge towards zero mean and unit variance, even under the presence of noise and perturbations.

We tried using Swish Activation function, but it occupies excessive memory. All the Linear Unit functions perform inplace operations in pyTorch, while Swish activation function creates a copy of data. This increases the memory requirement.




(a) GIoU comparison on train set (b) mAP comparison on train set (c) Objectness loss in train set

Figure 13: Study of using different Activation functions

We see *parametric* ELU has lowest gIoU loss and objectness loss compared to all the other activation functions. We see same trend for Validation dataset. But, the mAP, with PReLU drops with increasing epochs. It can be seen that in initial iterations, mAP improves faster with PReLU. This is expected as Leaky RELUs help in speeding up learning. So, using a parametric variant, helps to speed up the process further. But, LReLU gives best results at the end of 100 epochs. SELU on the other hand, performed very poorly. Learning a scaling factor for the positive values of input, can take longer to stabilize and improve accuracy. Since, we ran same number of epochs for all the activation functions, it might not be a fair comparison. We clearly see that SELU has a slower rate of improvement and would need more Epochs to match up the performance.

### 2.6.2 Model Initialization

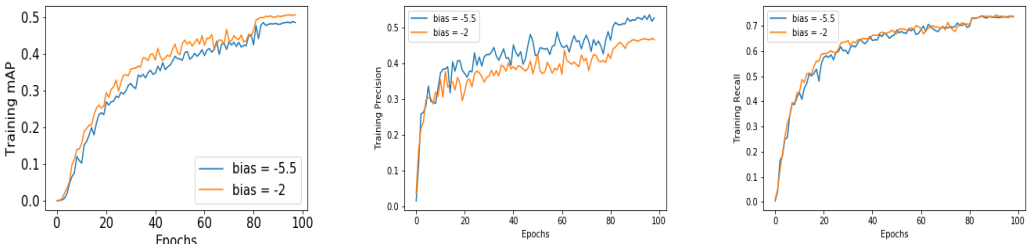
Binary classification models are by default initialized to have equal probability for outputs. In the presence of class imbalance, the loss due to the frequent class can dominate total loss and cause instability in early training. In VOC dataset, nearly 30% of the dataset is dominated by the class person. To model the concept of prior as explained in [4], we initialized the bias terms of the Conv2d() modules directly preceding the YOLO layers for neurons involved in detection and classification (but not regression). A bias of -5 correspond roughly to a 0.01 probability (i.e. torch.sigmoid(-5) = 0.007, so in this manner all detection neurons will output null detections in the first few batches. If it was initialized to 0, half of the neurons will report detections in the first batch, causing huge instabilities. We tried, two different values of initialization, to study how the training mAP is impacted and varies with epochs. In the base case, we initialized the bias to -5.5 and in the second case, we initialized it to -2. The reason being, we wanted to increase the initial probability from 0.007 to 0.12.

We see an improvement of 2.1% in mAP, with VOC 2012 dataset. (Fig 14) Since the findings were quite interesting, we tried running same experiments on a new VOC dataset, which is a combination of 2007 and 2012 datasets. We see 0.04% improvement in mAP in this case.

## 3 CenterNet: Anchor-free Object Detection

Most successful single stage detection algorithms like YOLOv3, SSD etc. relies on some form of anchor initialization to refine to the final detection location. Though much faster than their counterparts, their speed and performance is still limited by the choice of the anchor boxes. Further, there are few drawbacks that comes with this strategy of object detection:

- Anchor boxes introduce additional hyperparameters which increases the model complexity. Further, often detection is sensitive to these hyperparameters, thus a huge effort is required for hyperparamter tuning.
- The scales and aspect ratios of pre-defined anchor boxes are kept fixed during training, thus the next step cannot adjust the boxes which makes detection difficult for objects of all sizes.




(a) mAP comparison (b) Precision comparison (c) Recall Comparison

Figure 14: Comparison of mAP, precision and recall with different model initialization VOC(2012)

- Most of the pre-defined anchors are negative samples, which causes great imbalance between positive and negative sample during training.
- Finally, one design choice based on a particular dataset is not always applicable to other applications, thereby these methods lose generality [10].

These shortcomings served as the motivation to develop the anchor-free object detection methods to avoid the complicated computation related to anchor boxes.

### 3.1 Method

This method takes a novel approach by modelling an object as a single point- the center point of its bounding box. It relies on keypoint estimation to find center points and regresses to all other object properties such as size, 3D location, orientation and even pose. Further, CenterNet is end-to-end differentiable, simpler, real-time and more accurate than corresponding bounding box based detectors which is demonstrated by our experiments.

This approach is closely related to anchor-based one-stage approaches like YOLOv3 we discussed. A center point can be seen as a single shape-agnostic anchor. However, there are some major differences. First, CenterNet assigns the anchor based solely on location rather than based on box overlap. Second, since it has only one positive "anchor" per object, it does not require Non-Maximum Suppression (NMS) which is used in YOLOv3. Third, CenterNet uses a larger output resolution (output stride of 4) compared to traditional object detectors which eliminate the need for multiple anchors.

### 3.2 Experiment Results

We experimented with 3 different backbone architectures namely: ResNet-101, DLA-34 (with deformable convolution) and Hourglass-104 with batch-size 32 for 100 epochs. Apparently, hourglass needs more epochs to converrge and has slow inference as well compared to other backbone architectures. For ResNet-101 we tried with and without deformable convolution layers with input image size of 384 with data augmentation. We report the inference with respect to scaling if images. As we can see in Fig. 17, the mAP first increases and then decreases which leads to conclusion that if we perform inference on image-size much larger or smaller than the training input image-size then it results in poor inference results. Further from table 2, we see that mAP of CenterNet is higher than YOLOv3, where as FPS of YOLOv3 beats CenterNet. From this we can conclude that CenterNet provides a good trade-off between accuracy and inference time.

## 4 Conclusion

We trained our YOLOv3 model on VOC dataset and supplemented it with deformable convolutions, anti-aliasing and show slight improvement in mAP and classification loss respectively. Our experiments on group norm, anchor-box estimation, and changes to activation functions give us an insight into the workings of YOLOv3 and modern recipes of training neural network.

We could not try multiscale training and swish activation function because of memory issues. Also, finding a method to translate the bounding box when translating the images while performing anti-aliasing experiments was time-consuming.

| Model | mAP | FPS |
|---|---|---|
| DLA-34 | 0.795 | 45 |
| Hourglass | 0.686 | 15 |
| Resnet-101 | 0.753 | 51 |
| Resnet-101-DCN | **0.768** | 52 |
| YOLOv3 | 0.635 | 43 |

Table 2: mAP of CenterNet (with different backbone) and YOLOv3

| Optimizer | mAP |
|---|---|
| Adam | 0.795 |
| SGD | 0.759 |

Table 3: mAP using different optimizers for CenterNet(DLA backbone).

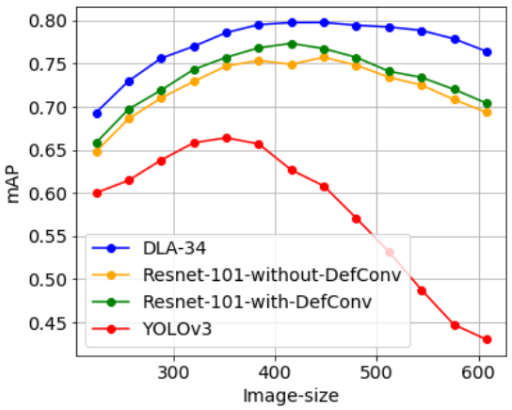



Figure 15: mAP vs image-size

We also trained our CenterNet model on PASCAL VOC and applied deformable convolutions on its ResNet backbone and show an improvement in mAP. Finally, we demonstrate how inference changes depending on the test image size for YOLOv3 and CenterNet.

## References


[1] Jifeng Dai et al. "Deformable Convolutional Networks". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (Oct. 2017). DOI: 10.1109/iccv.2017.89. URL: http://dx.doi.org/10.1109/ICCV.2017.89.

[2] M. Everingham et al. "The Pascal Visual Object Classes Challenge: A Retrospective". In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[3] Ayoosh Kathuria. *What's new in YOLO v3?* Medium. Apr. 29, 2018. URL: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b (visited on 12/06/2019).

[4] T. Lin et al. "Focal Loss for Dense Object Detection". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017, pp. 2999–3007. DOI: 10.1109/ICCV.2017.324.

[5] Tejus Adiga M. *Downsampling*. Technology. URL: https://www.slideshare.net/tejusadiga/resampling-54490779 (visited on 12/06/2019).

[6] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/. 2013–2016.

[7] Shaoqing Ren et al. "Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 91–99. URL: http://dl.acm.org/citation.cfm?id=2969239.2969250.

[8] Hamid Rezatofighi et al. "Generalized Intersection over Union". In: (June 2019).

[9] Yuxin Wu and Kaiming He. "Group Normalization". In: *Lecture Notes in Computer Science* (2018), pp. 3–19. ISSN: 1611-3349. DOI: 10.1007/978-3-030-01261-8_1. URL: http://dx.doi.org/10.1007/978-3-030-01261-8_1.

[10] Tong Yang et al. "MetaAnchor: Learning to Detect Objects with Customized Anchors". In: *CoRR* abs/1807.00980 (2018). arXiv: 1807.00980. URL: http://arxiv.org/abs/1807.00980.

[11] Richard Zhang. "Making Convolutional Networks Shift-Invariant Again". In: *ICML*. 2019.

[12] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. "Objects as Points". In: *CoRR* abs/1904.07850 (2019). arXiv: 1904.07850. URL: http://arxiv.org/abs/1904.07850.