

Bristol DAQ ML Studies

Bristol DUNE Meeting 10.3.21

Joel Greer + Raul Stein

Possible Overfitting

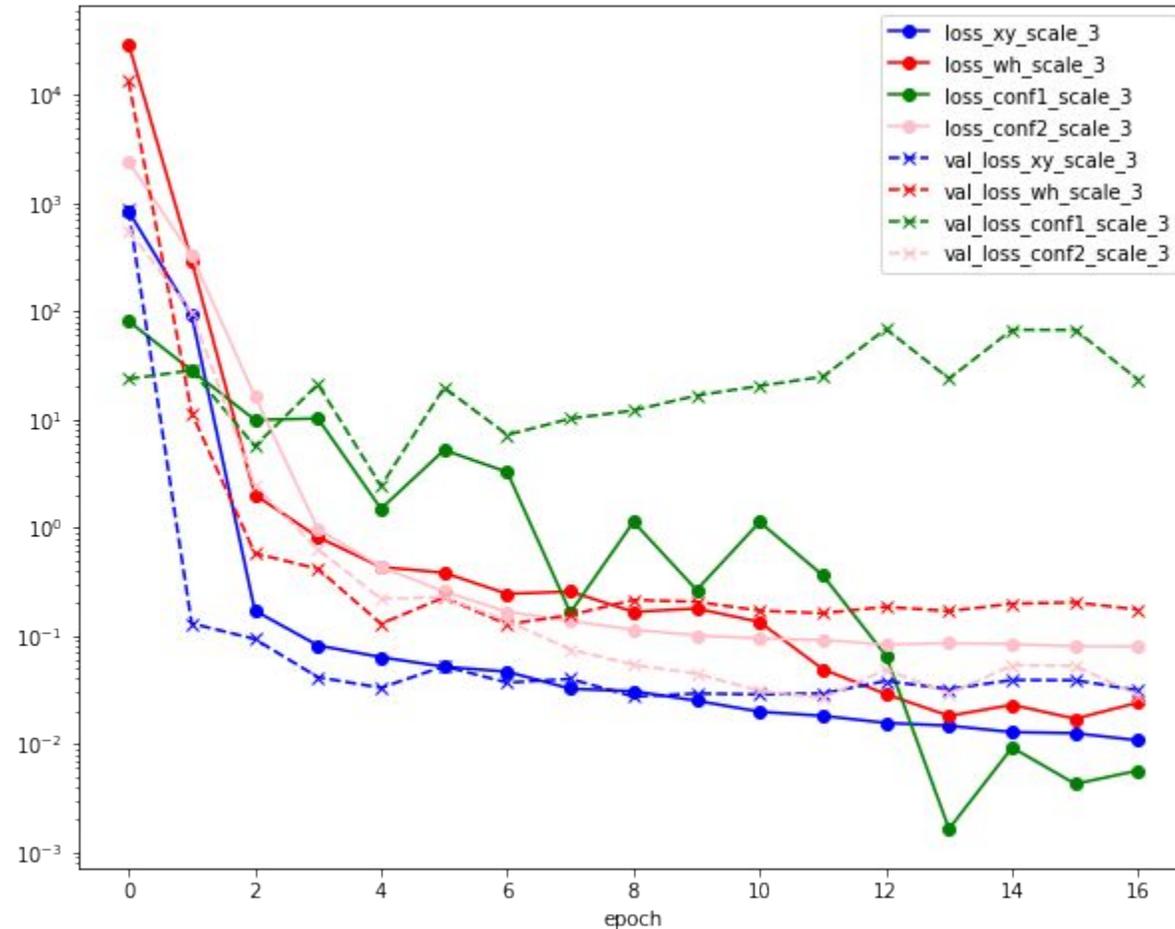
We have been testing things on datasets which are apparently quite small to train YOLOs detectors on.

- 200 images for raccoons
 - 176 train
 - 16 val
 - 8 eval
- 768 images for bottles
 - 688 train
 - 72 val
 - 8 eval
- 536 images for SNNu
 - 480 train
 - 48 val
 - 8 eval

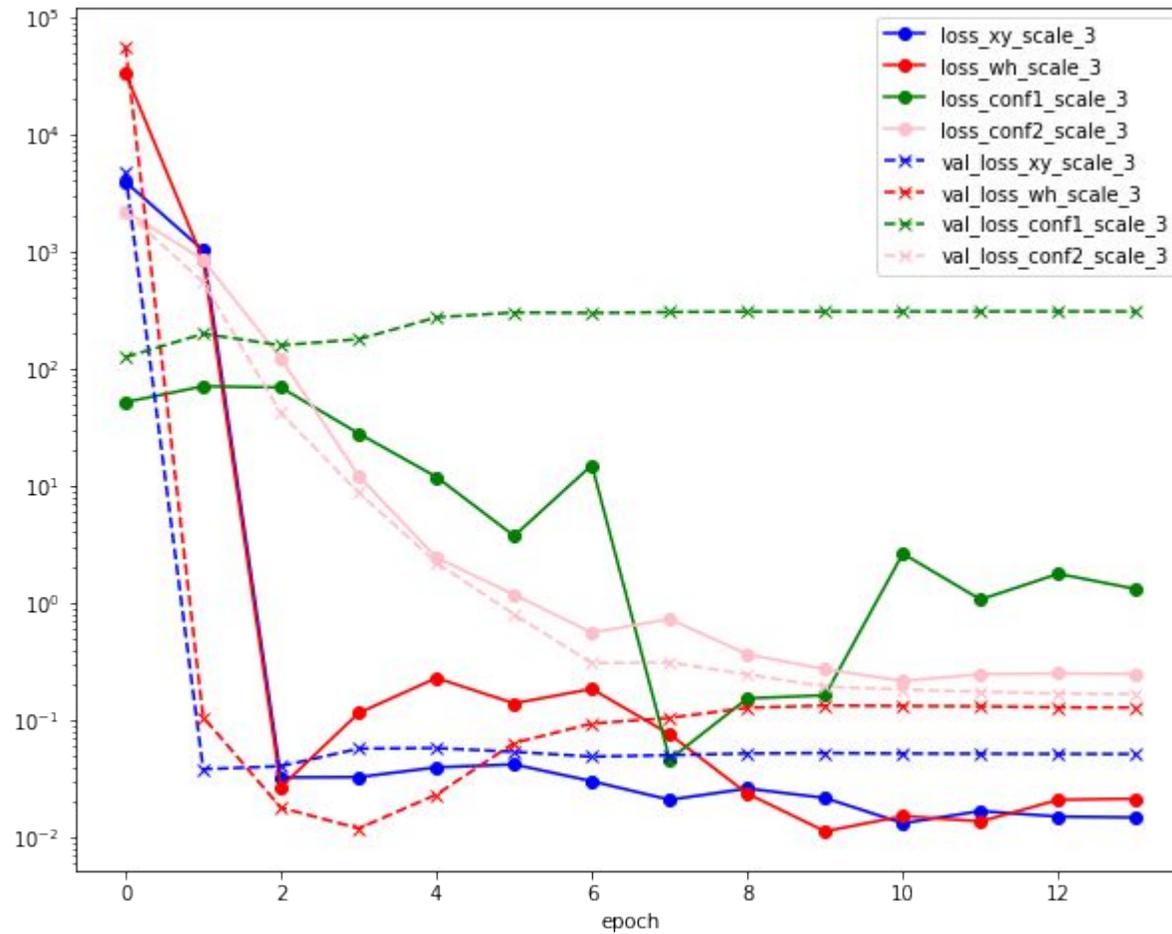
We also have disabled image augmentations. There are some such as flipping, shifting which we think we could implement even in our SNNu dataset.

Possible Overfitting - SNNu

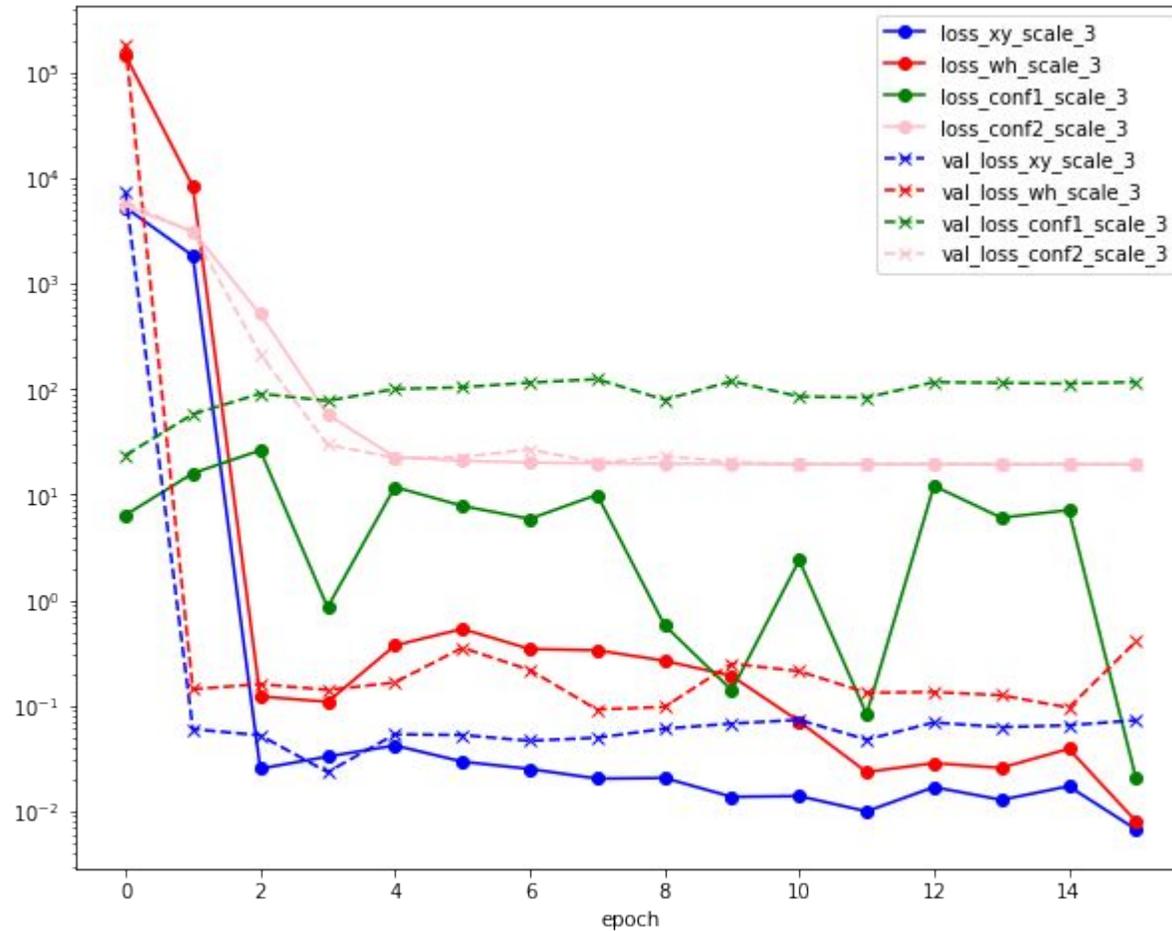
Looking at the loss terms from the training set vs those from the validation set might help identify overfitting.



Possible Overfitting - Raccoons



Possible Overfitting - Bottles



Dataset Size

- We actually have ~30 images in our SNNu dataset which we can make use of, so have set a run which trains over the entire set to try and avoid overfitting. We split this by 80%/10%/10% into train, validation and evaluation datasets.
- We could also try finding a larger non-sparse dataset to use as a similar check.
- We will also look at the effect of image augmentations, which can make a dataset effectively much larger, as long as it is already quite generalised

Backup Slides

YOLOv3 Loss

x,y term

```
xy_delta = xywh_mask * (pred_box_xy-true_box_xy) * wh_scale * self.xywh_scale
```

- xywh_mask - all 1's if in warmup batch, otherwise only 1 for predictions with associated truth box
- pred_box_xy = `(self.cell_grid[:, :grid_h, :grid_w, :, :] + tf.sigmoid(y_pred[... , :2]))`
which centres a box on the grid cell which it was predicted by and adds the sigmoided predicted output x,y values
- true_box_xy - if in a warmup batch, we care about all predictions and add `self.cell_grid[:, :grid_h, :grid_w, :, :] * (1-object_mask)` to the true_box_xy, so that we effectively create a truth box in the centre of the grid cell for all predicted boxes which actually have no associated truth box.
- wh_scale - scale factor of $1(480 \times 480) - 2(0 \times 0)$ applied to each element depending on the relative area of the truth box and the downsampled image
- self.xywh_scale - scaling factor = 1.

YOLOv3 Loss

w,h term

```
wh_delta = xywh_mask * (pred_box_wh-true_box_wh) * wh_scale * self.xywh_scale
```

- `xywh_mask` - all 1's if in warmup batch, otherwise only 1 for predictions with associated truth box
- `pred_box_wh` - `y_pred[..., 2:4]` which is the natural logarithm of the scaling factor for the width and height of the predicted boxes. Which can be multiplied by `anchor_w,h/downsampled_image_w,h` to get output boxes
- `true_box_wh` - if in a warmup batch, we care about all predictions and add `tf.zeros_like(true_box_wh) * (1-object_mask)` to the `true_box_xy`, so that we effectively create a truth box in the centre of the grid cell for all predicted boxes which actually have no associated truth box which has a w,h of 0x0.
- `wh_scale` - scale factor of $1(480 \times 480) - 2(0 \times 0)$ applied to each element depending on the relative area of the truth box and the downsampled image
- `self.xywh_scale` - scaling factor = 1.

YOLOv3 Loss

conf term

```
conf_delta = object_mask * (pred_box_conf-true_box_conf) * self.obj_scale +  
(1-object_mask) * conf_delta * self.noobj_scale
```

- `object_mask` - 1 for predictions with associated truth boxes, 0 otherwise
- `pred_box_conf` - `tf.expand_dims(tf.sigmoid(y_pred[..., 4]), 4)`, the objectness scores for all predictions
- `true_box_conf` - 1 for grid cell anchors with associated truth boxes, 0? otherwise - should check again
- `self.obj_scale` - scaling factor = 50
- `(1-object_mask)` - 0 for predictions with associated truth boxes, 1 otherwise
- `conf_delta` - predicted box objectness score for all predicted boxes which overlap with a truth box by 0.33 IOU or less
- `self.noobj_scale` - scaling factor = 1

Last week I found that the contribution to this term from predictions with associated truth boxes was negative! Due to `pred_box_conf-true_box_conf`. Will minimise cost more when the predictions are worse!

YOLOv3 Loss

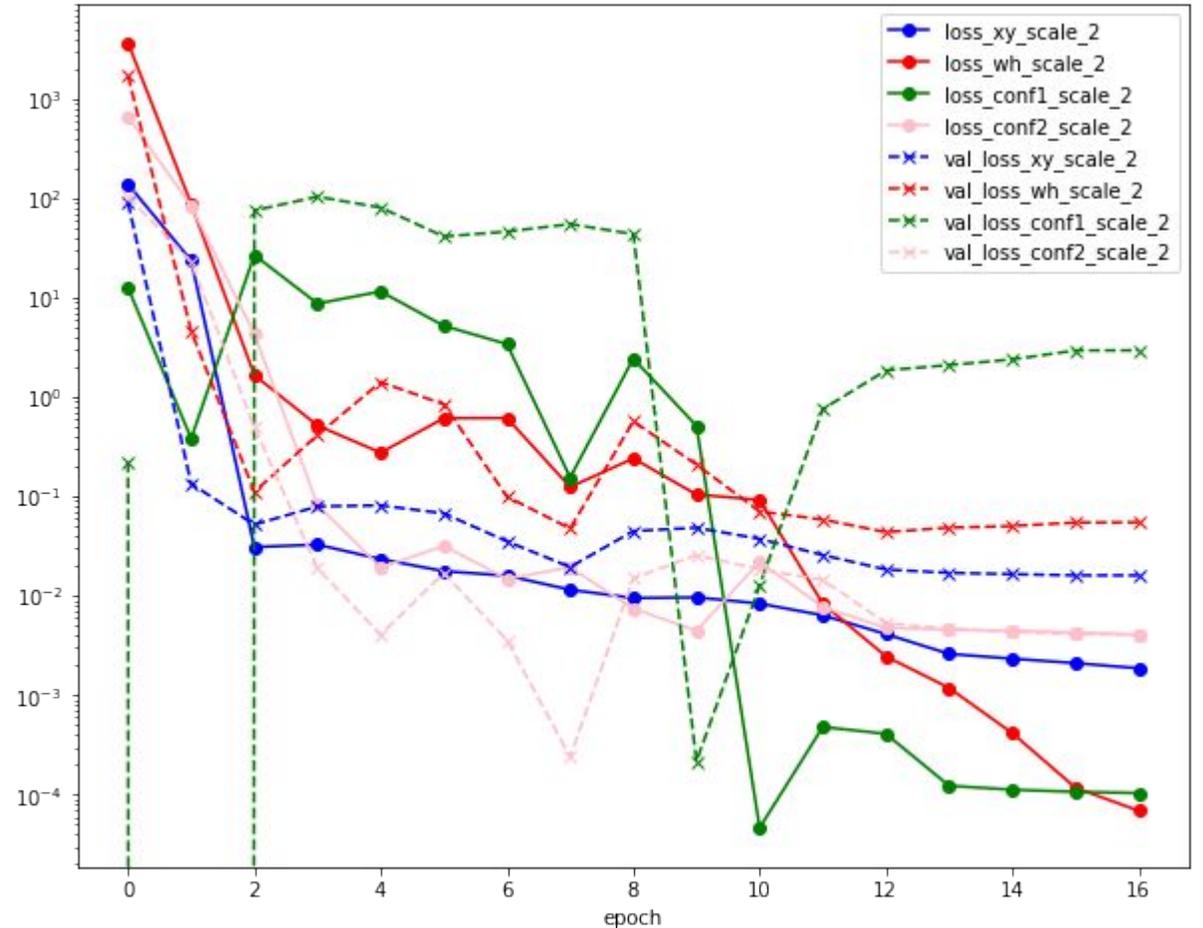
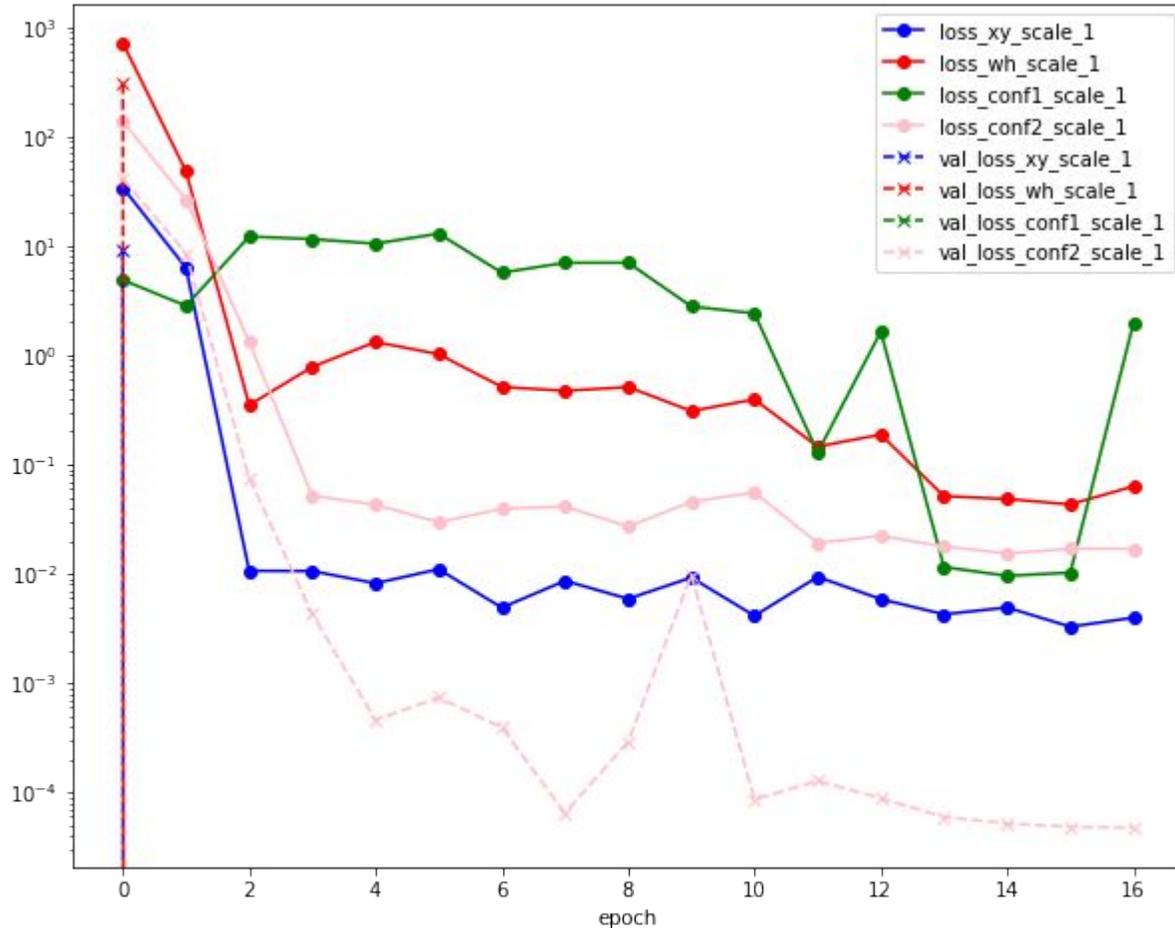
class term

```
class delta = object mask * \  
tf.expand_dims(tf.nn.sparse_softmax_cross_entropy_with_logits(labels=true_box_class,  
logits=pred_box_class), 4) * \  
self.class_scale
```

- object_mask - 1 for predictions with associated truth boxes, 0 otherwise
- true_box_class - one-hot class of truth box
- pred_box_class - vector of predicted classes. List should be 1 class long. - ought to explicitly check dimensions are okay in this calculation
- self.class_scale - scaling factor = 1

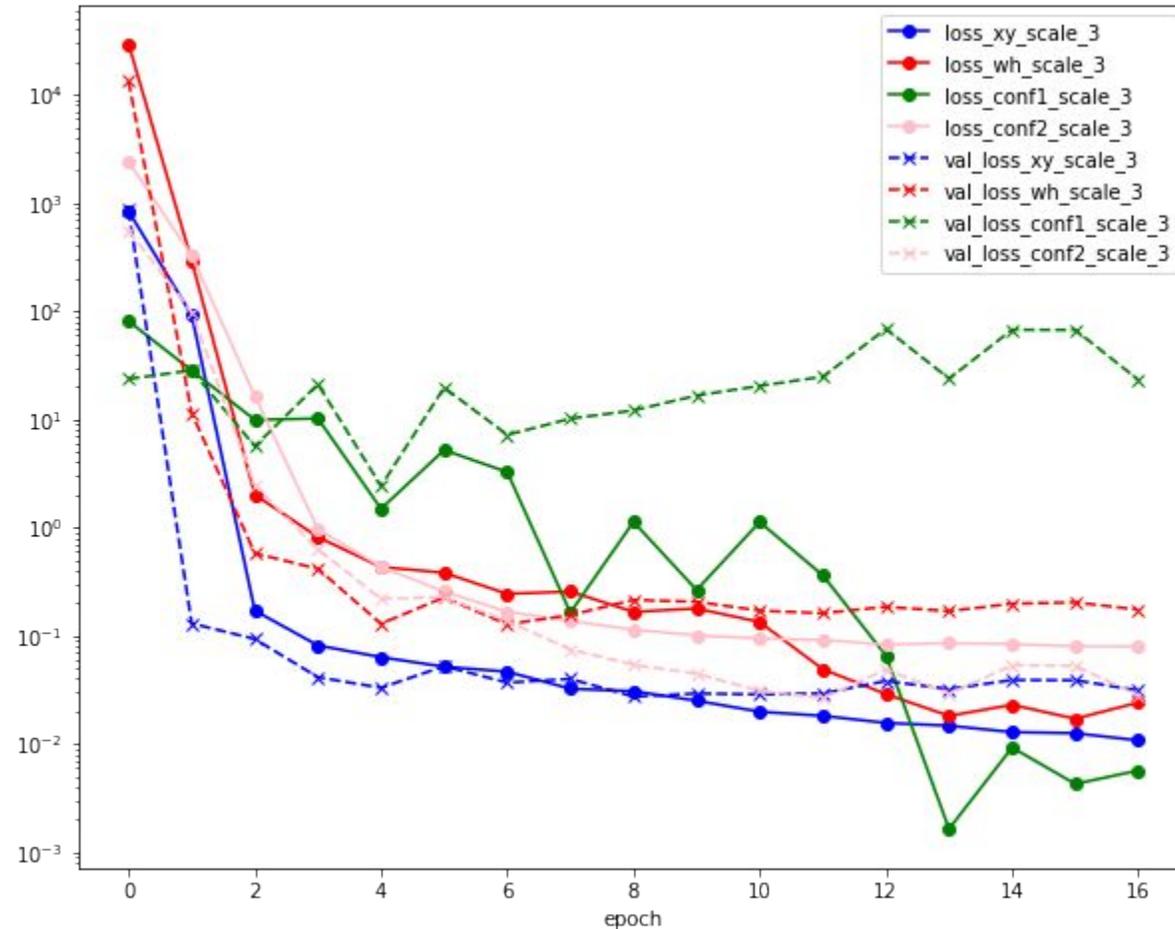
Possible Overfitting - SNNu

Looking at the loss terms from the training set vs those from the validation set might help identify overfitting.

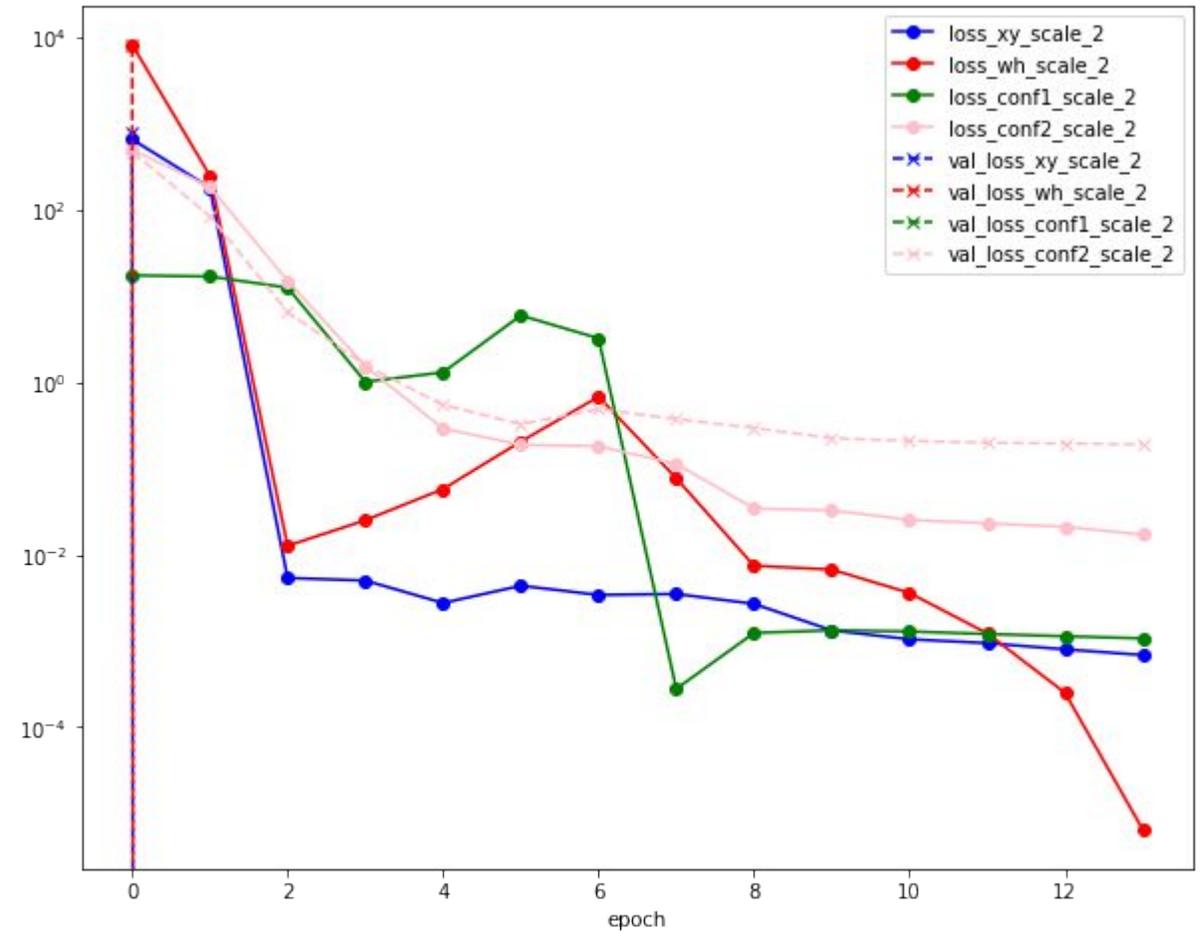
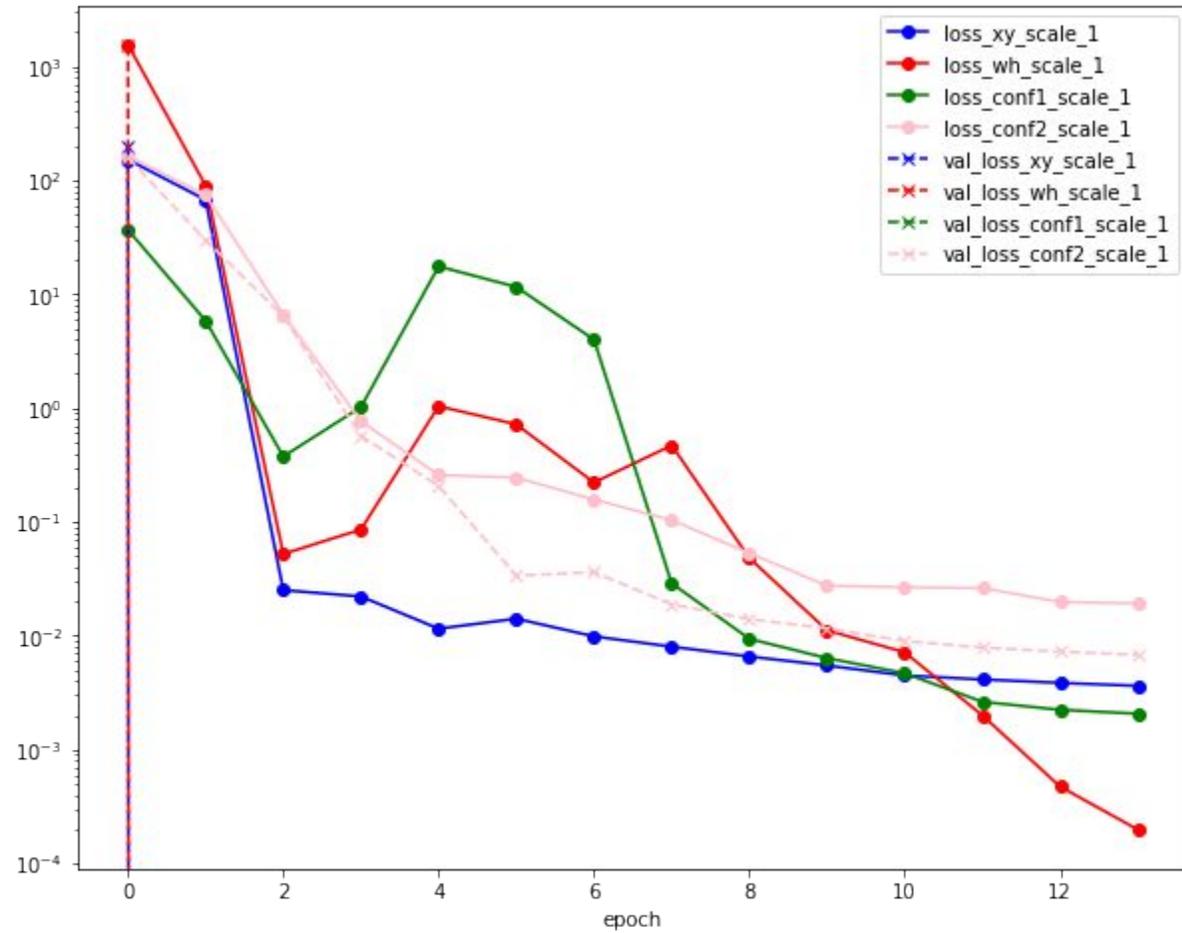


Possible Overfitting - SNNu

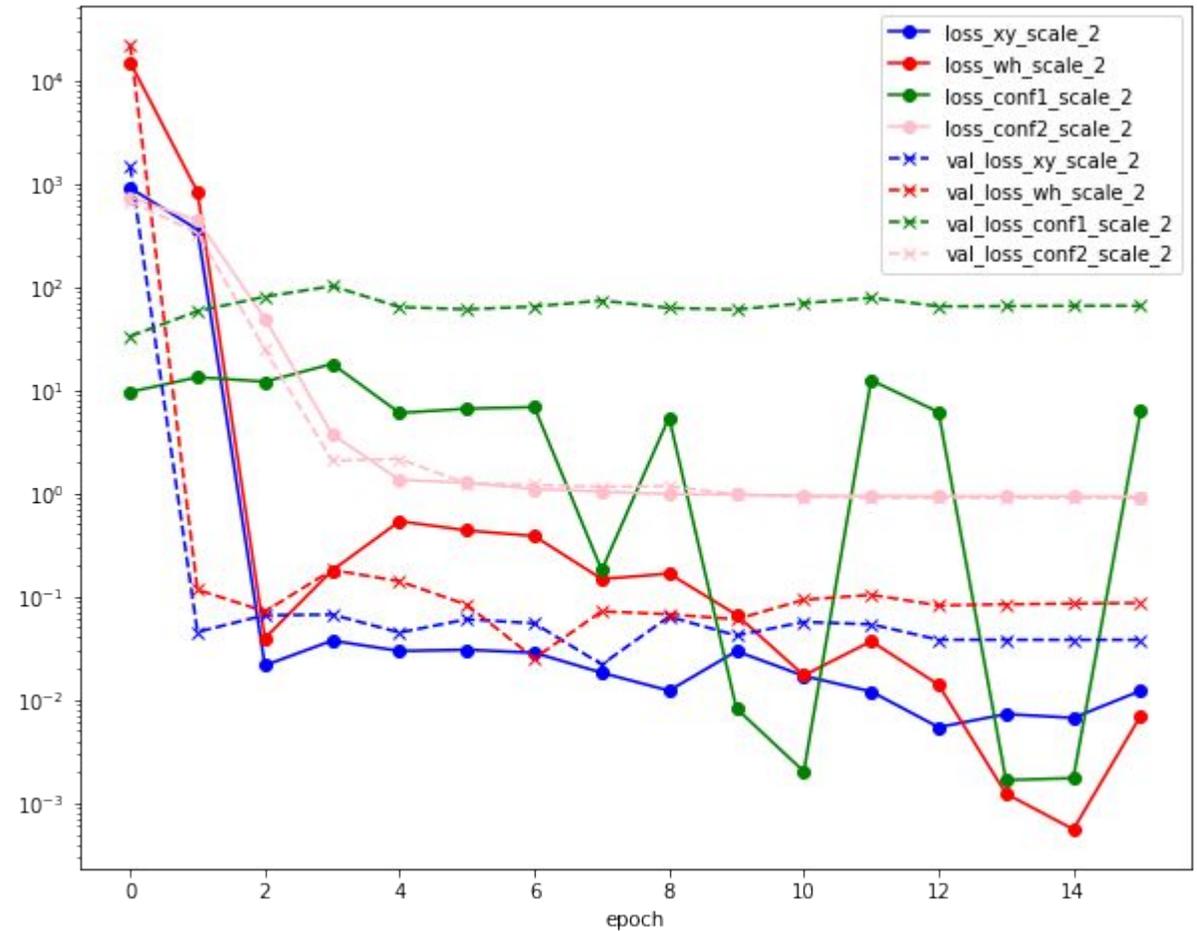
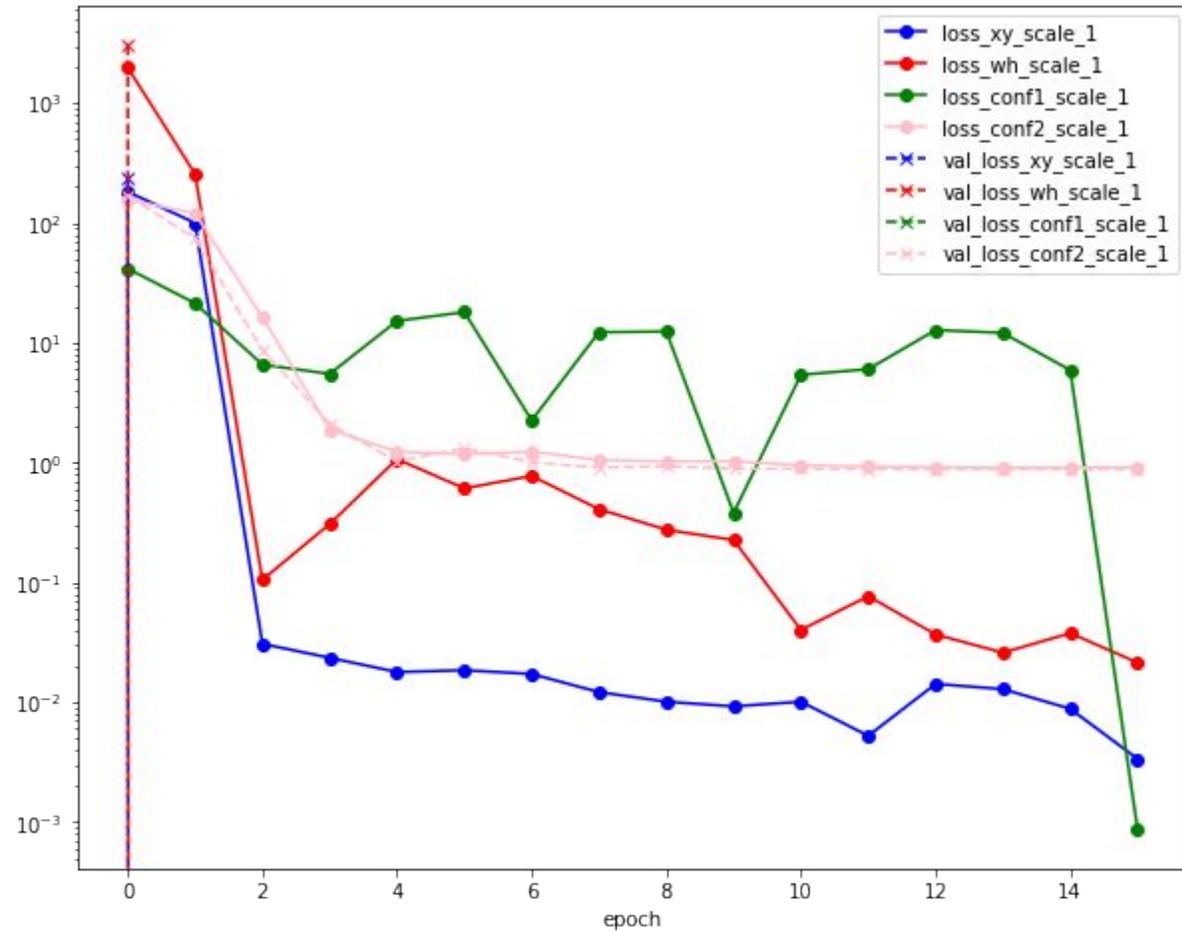
Looking at the loss terms from the training set vs those from the validation set might help identify overfitting.



Possible Overfitting - Raccoons



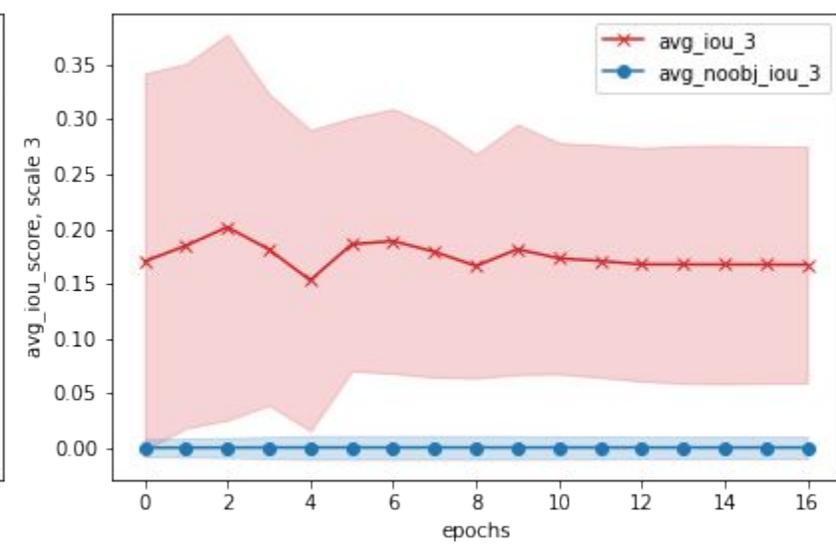
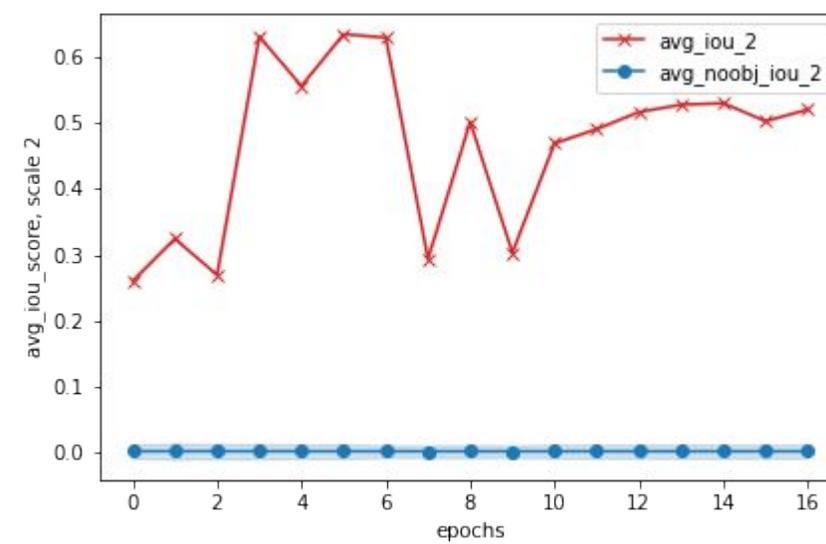
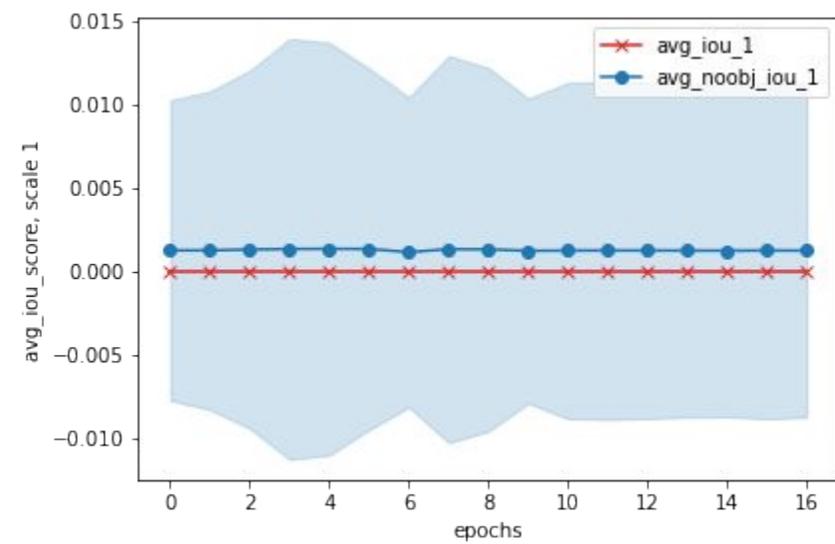
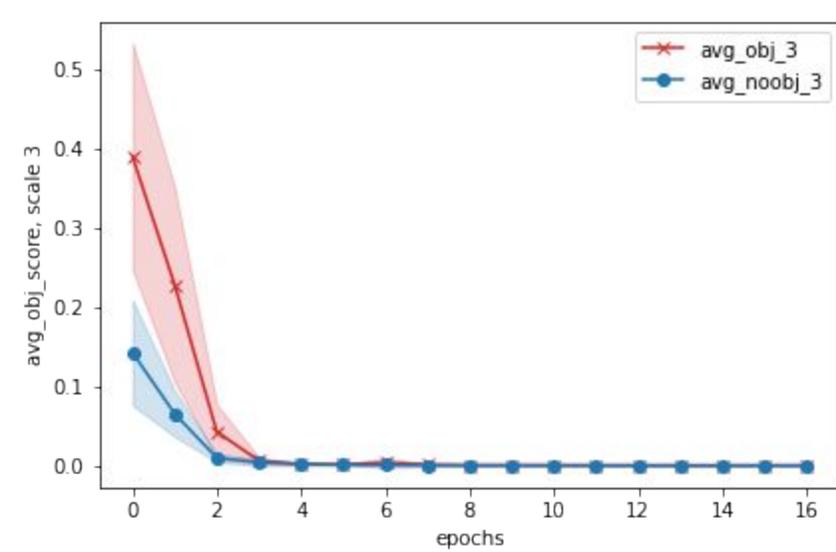
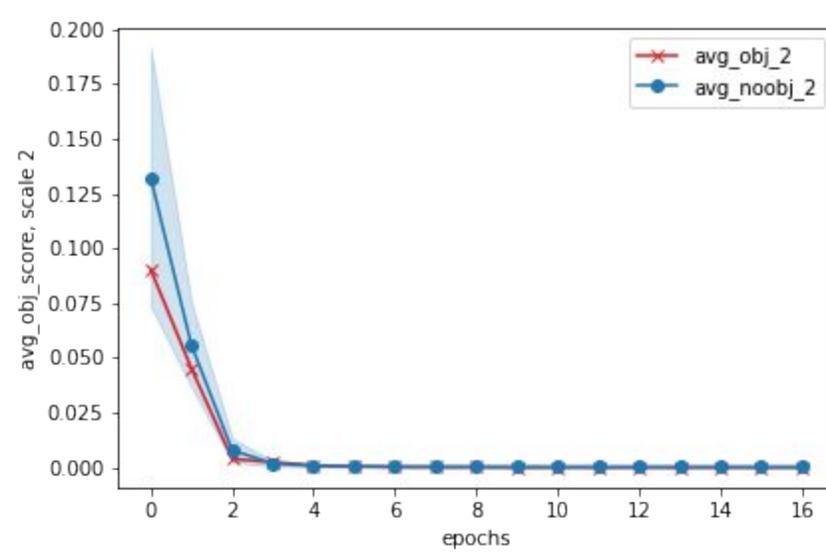
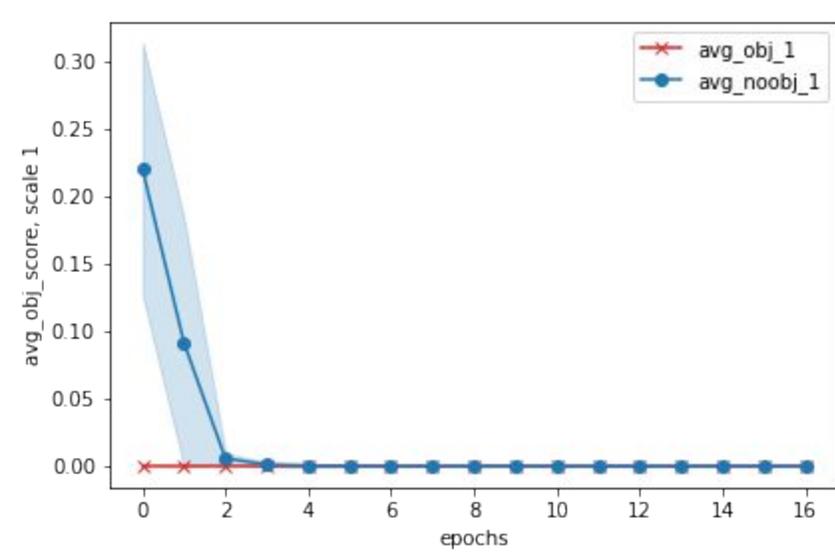
Possible Overfitting - Bottles



YOLOv3 Loss

We seem to have a problem where the network fails to learn to predict good objectness scores for predictions made by grid cells in the network which are local to a truth box and have which use anchor (seed) boxes of a similar scale.

For example, with SNNus we saw this drop-off



YOLOv3 Loss

Looking into the loss function to understand why.

From each of the 3 scales of the network we get a contribution to the loss. They are currently all equally weighted.

At each scale, there are 4 contributions to the loss:

- x,y
- w,h
- objectness score
- class scores - expect this to be 0 if there is only 1 class

YOLOv3 Loss

```
loss_xy      = tf.reduce_sum(tf.square(xy_delta),      list(range(1,5)))  
loss_wh      = tf.reduce_sum(tf.square(wh_delta),      list(range(1,5)))  
loss_conf    = tf.reduce_sum(tf.square(conf_delta),    list(range(1,5)))  
loss_class   = tf.reduce_sum(class_delta,              list(range(1,5)))  
  
loss = loss_xy + loss_wh + loss_conf + loss_class
```

I find `loss_class` is always 0, as expected

Warmup Training

Some of the contributions to the loss depend on whether we are doing 'warmup training' or not. This forces predicted boxes to be the same size as the anchor box

```
batch_seen = tf.assign_add(batch_seen, 1.)
```

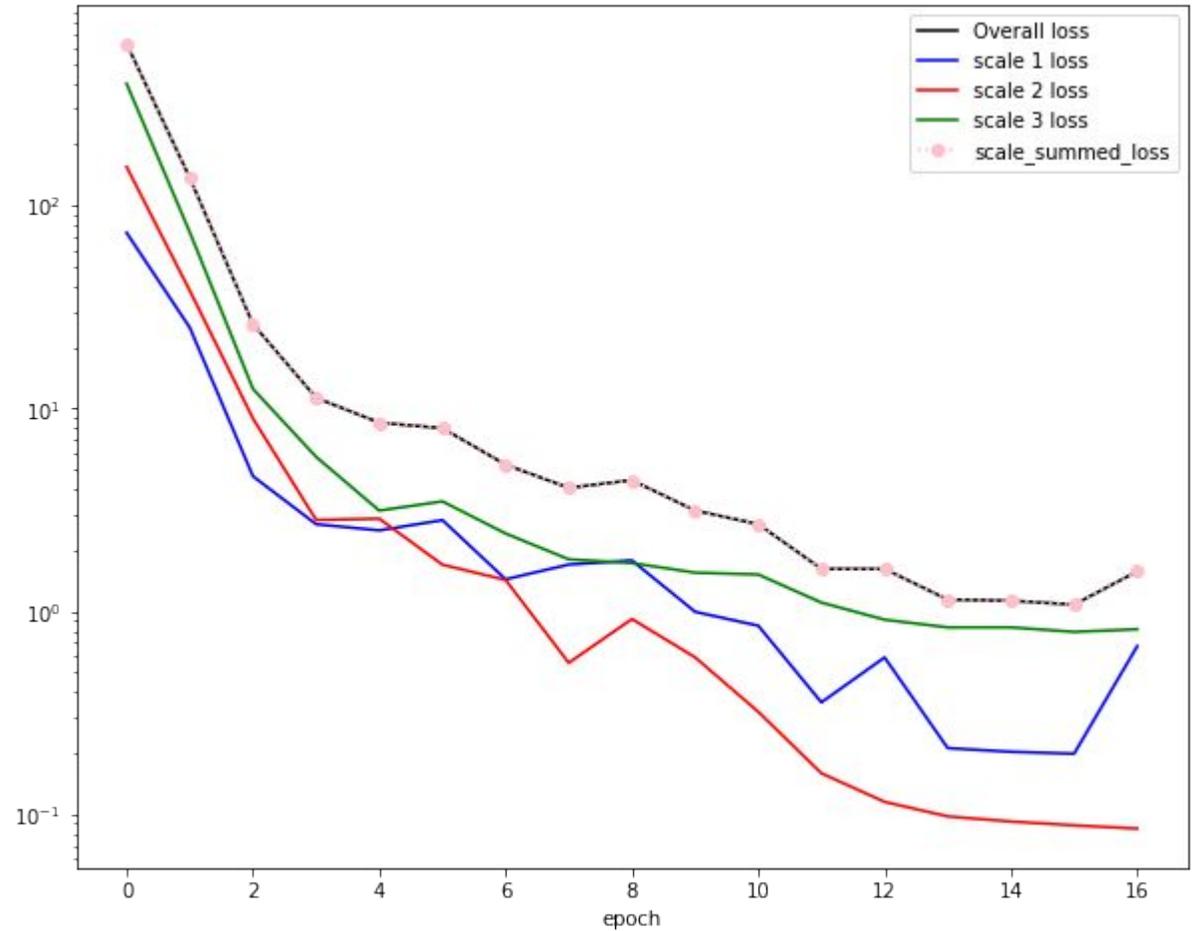
```
true_box_xy, true_box_wh, xywh_mask = tf.cond(tf.less(batch_seen, self.warmup_batches+1),  
                                              lambda: [true_box_xy + (0.5 + self.cell_grid[:, :grid_h, :grid_w, :, :]) * (1-object_mask),  
                                              true_box_wh + tf.zeros_like(true_box_wh) * (1-object_mask),  
                                              tf.ones_like(object_mask)],  
                                              lambda: [true_box_xy,  
                                              true_box_wh,  
                                              object_mask])
```

- If in warmup batch, effectively add a truth box for all predictions with no associated truth box which is in the centre of the grid cell of that prediction
- If in warmup batch add 0.0 to all entries in width and height tensor with no associated truth box - which makes the predicted boxes the same shape as the anchor boxes
- If in warmup batch include contributions from all predictions (via xywh_mask), otherwise mask out any contributions to the box localization parts of the loss which have no associated truth box

Overall Loss

To determine the total loss in the network, the loss contributions from each scale are summed.

It is less clear how the contributions to the loss for a given scale actually sum together to give the loss value for that scale.



Overall Loss

It is less clear how the contributions to the loss for a given scale actually sum together to give the loss value for that scale.

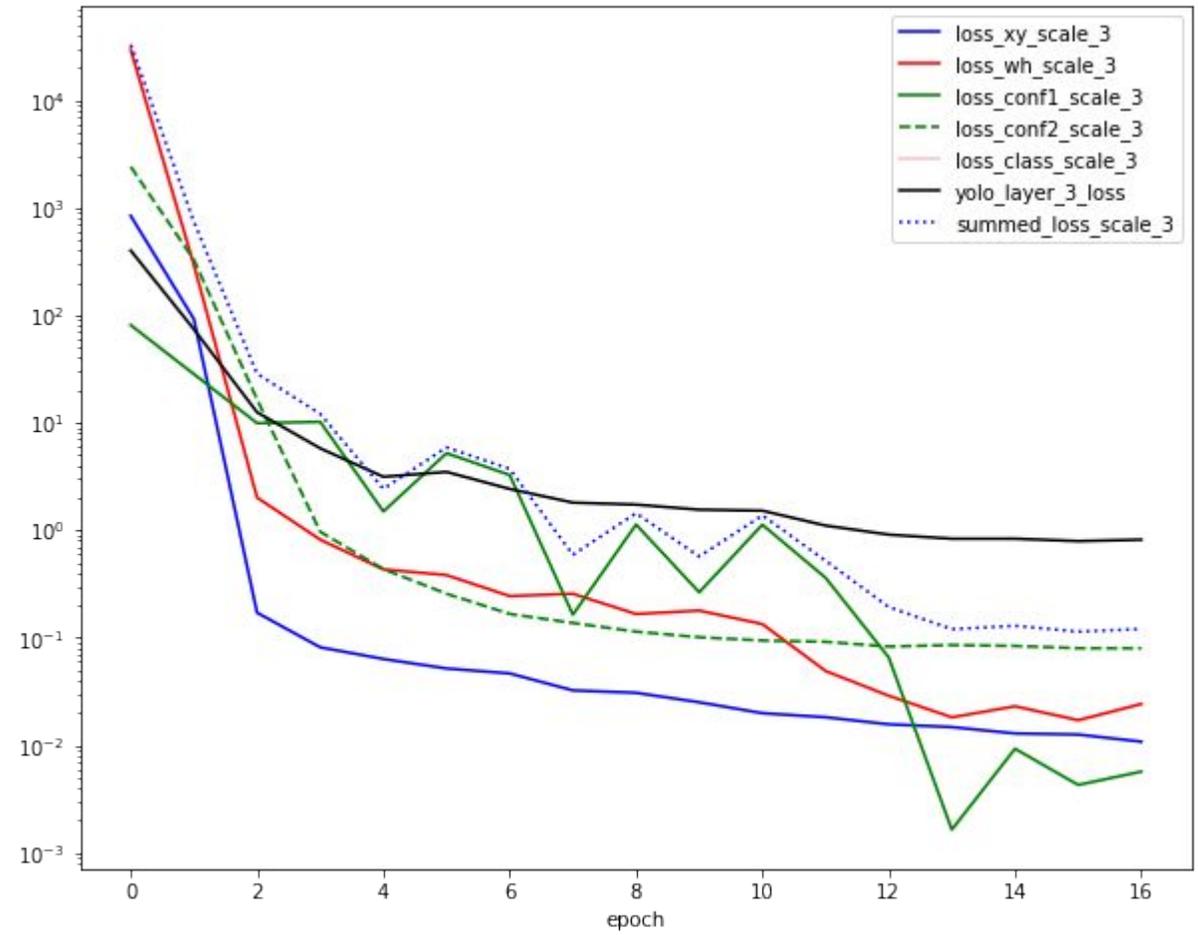
The network uses an Adam optimizer and calculates the final loss via:

```
Loss = tf.sqrt(tf.reduce_sum(y_pred))
```

Where:

```
Y_pred = [loss_yolo_1, loss_yolo_2,  
loss_yolo_3]
```

The losses in this list are not scalars though...



Overall Loss

From a single scale, the loss seems to have the dimensions of 8 (ie:the batch size) and be left to aggregate over multiple batches. At the moment I understand this aggregation to be a mean over all batches. Not clear to me why the sum of the loss terms which I have logged doesn't equal the loss for this scale...

Maybe this is to do with the optimizer - but not really sure

```
loss_xy      = tf.reduce_sum(tf.square(xy_delta),      list(range(1,5)))  
loss_wh      = tf.reduce_sum(tf.square(wh_delta),      list(range(1,5)))  
loss_conf    = tf.reduce_sum(tf.square(conf_delta),    list(range(1,5)))  
loss_class   = tf.reduce_sum(class_delta,              list(range(1,5)))  
  
loss = loss_xy + loss_wh + loss_conf + loss_class
```

Investigate Loss - Part 1

conf term

```
conf_delta = object_mask * (pred_box_conf-true_box_conf) * self.obj_scale +  
(1-object_mask) * conf_delta * self.noobj_scale
```

Firstly, ensure the contribution from the predicted object score from a predicted box which has an associated truth box is not negative:

```
conf_delta = object_mask * (true_box_conf-pred_box_conf) * self.obj_scale +  
(1-object_mask) * conf_delta * self.noobj_scale
```

Investigate Loss - Part 1

conf term

Also, separate out the contributions from predicted boxes with associated truth from those without before applying MSE cost function:

```
conf_delta1 = object_mask * (true_box_conf - pred_box_conf) * self.obj_scale
conf_delta2 = (1 - object_mask) * conf_delta * self.noobj_scale
```

So that:

```
loss_conf1 = tf.reduce_sum(tf.square(conf_delta1), list(range(1, 5)))
loss_conf2 = tf.reduce_sum(tf.square(conf_delta2), list(range(1, 5)))
loss = loss_xy + loss_wh + loss_conf1 + loss_conf2 + loss_class
```

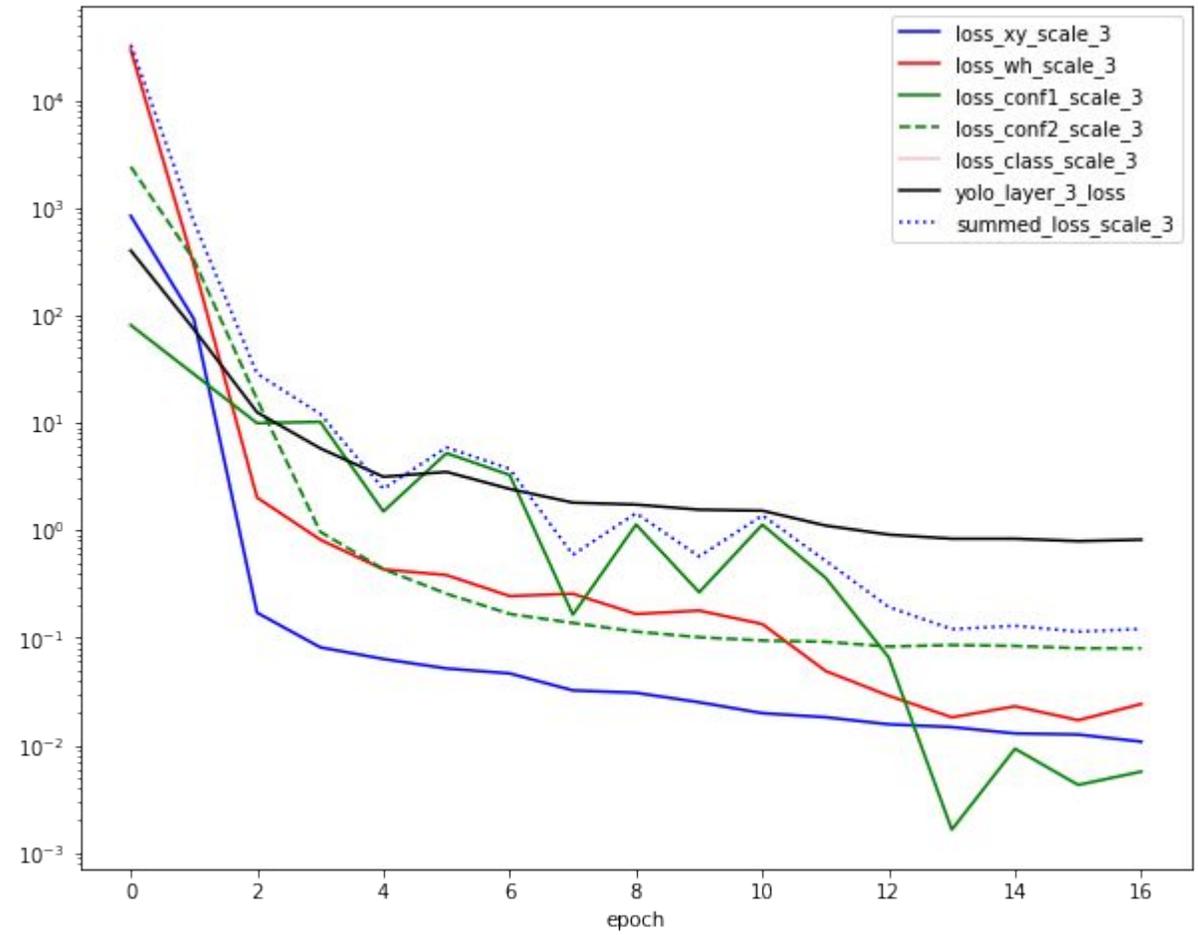
and compare these contributions

Investigate Loss - Part 1

Looking at the contributions to the loss from object confidence terms, first for the most detailed scale (has the most grid cells).

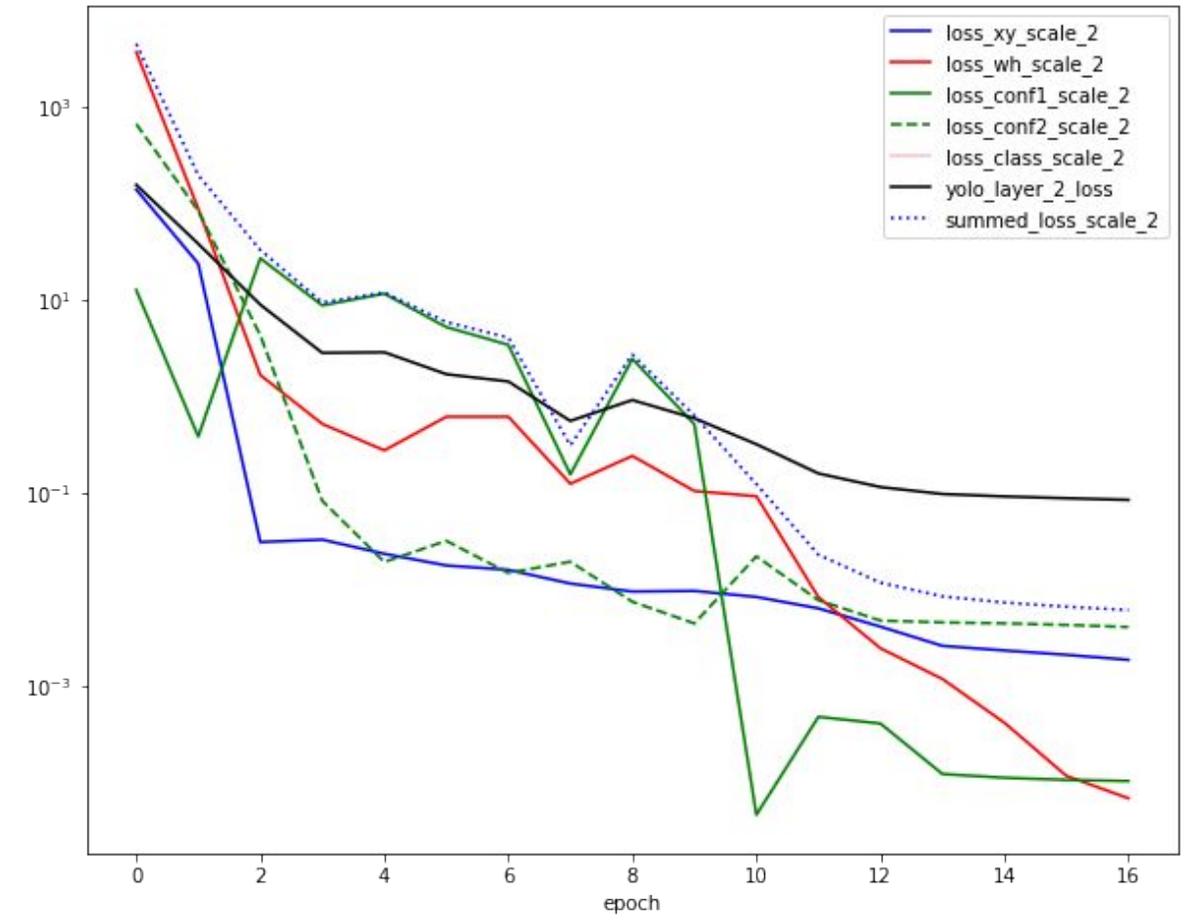
After we finish the first 2 epochs, the contribution to the loss from the object confidence term with associated truth boxes is the largest contributor to the loss until ~epoch 12.

However, the cost function doesn't decrease very monotonically! Are we jumping between local minima?



Investigate Loss - Part 1

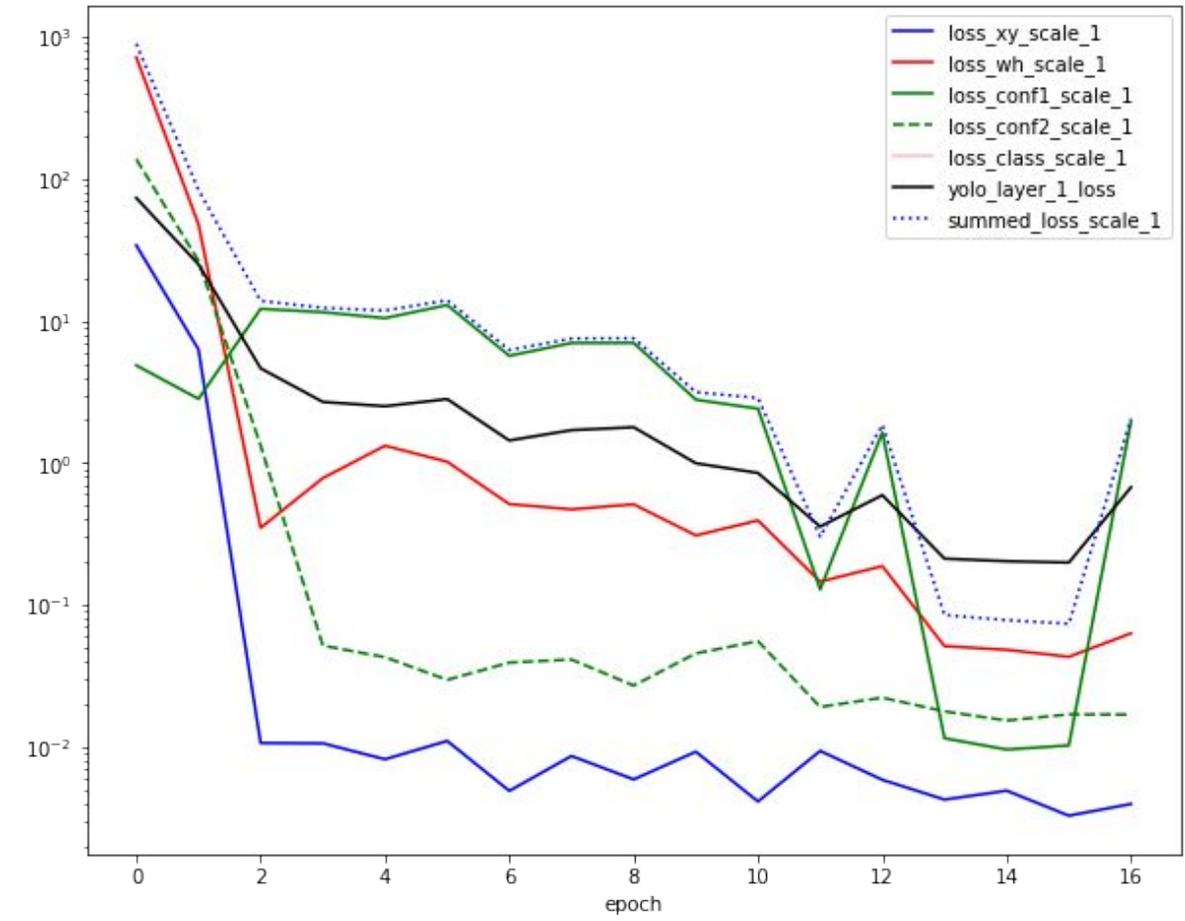
The other 2 scales have smaller contributions to the loss but show reasonably similar behaviour where the contribution to the loss from the object confidence term with associated truth boxes is large compared to the other terms for most of training.



Investigate Loss - Part 1

The same is true for the coarsest scale.

But in all cases this term behaviours in a more volatile way than the others.



Investigate Loss - Part 1

	loss_conf1_scale_1	loss_conf2_scale_1	loss_conf1_scale_2	loss_conf2_scale_2	loss_conf1_scale_3	loss_conf2_scale_3	epochs
0	4.863313	137.684311	12.627116	663.375732	80.834183	2399.086670	0
1	2.825784	26.481405	0.384205	83.355148	28.263361	326.590149	1
2	12.162152	1.320167	26.921486	4.299109	9.889252	16.447977	2
3	11.512643	0.051764	8.747487	0.082776	10.187667	0.967854	3
4	10.458376	0.042846	11.578640	0.019323	1.493090	0.434784	4
5	12.918453	0.029678	5.225511	0.031954	5.203952	0.256548	5
6	5.680512	0.039270	3.418803	0.014872	3.257629	0.166545	6
7	6.992757	0.041288	0.155682	0.019435	0.163925	0.137319	7
8	7.005771	0.027073	2.446544	0.007478	1.129768	0.113972	8
9	2.789043	0.045585	0.514700	0.004462	0.264861	0.101106	9
10	2.410783	0.055573	0.000047	0.022099	1.125782	0.094264	10
11	0.128484	0.019131	0.000481	0.007748	0.361972	0.091761	11
12	1.637591	0.022234	0.000410	0.004772	0.065747	0.082847	12
13	0.011516	0.017806	0.000123	0.004572	0.001646	0.085598	13
14	0.009587	0.015356	0.000112	0.004474	0.009297	0.083927	14
15	0.010250	0.016966	0.000107	0.004304	0.004281	0.079959	15
16	1.961592	0.016902	0.000104	0.004096	0.005701	0.079892	16

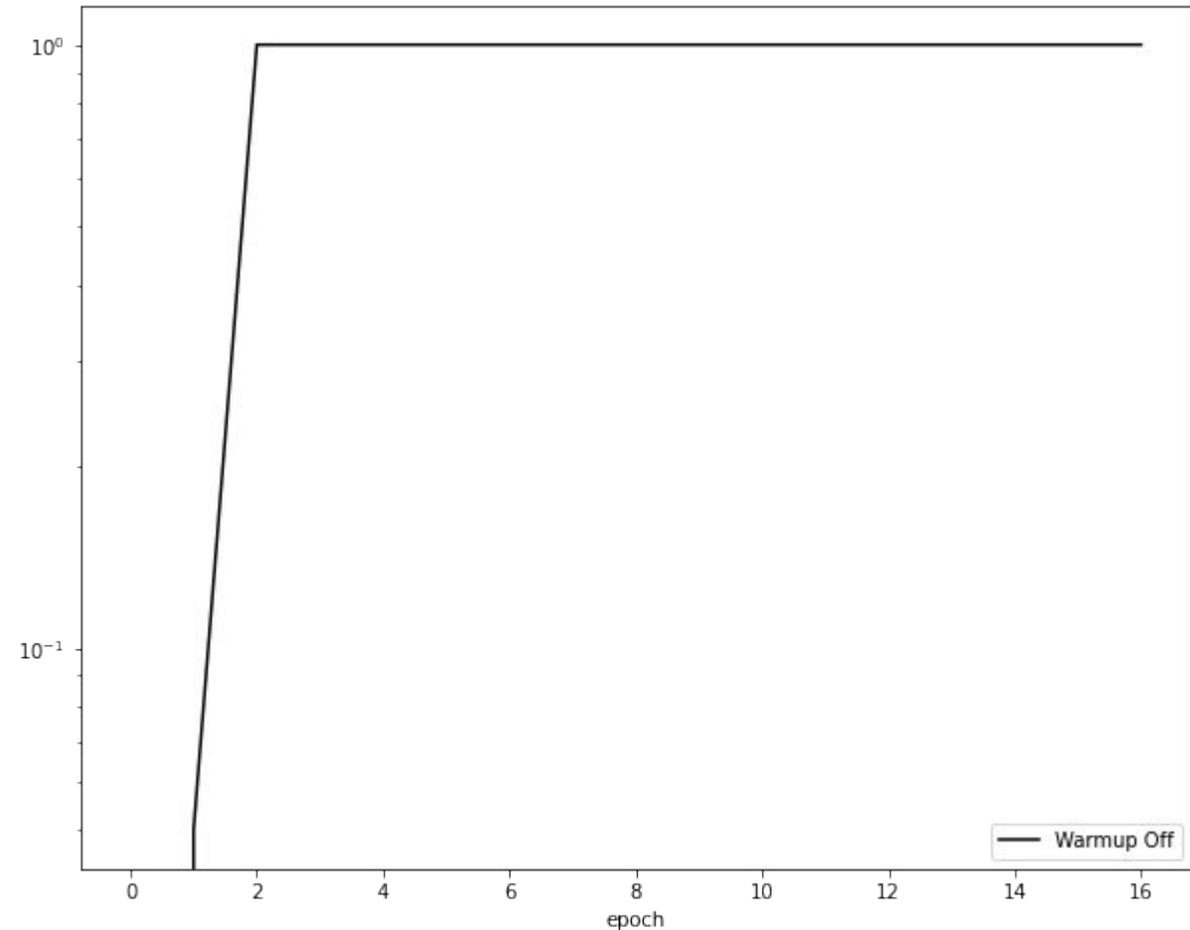
Investigate Loss - Part 2

How often are we in a warmup batch?

All of the first epoch and 1/20th of the second epoch (if aggregation is indeed done by mean over batches).

According to some of my reading, warm-up batches are designed to let the network stabilize for the object classification problem and after that let it train on localization.

Will try more warmup batches as well as what happens if I explicitly only train on obj score during warmup.



Investigate Loss - Part 3

Some parts of our loss function may be different from those used in the YoloV3 paper.

Firstly: *YOLOv3 predicts an objectness score for each boundingbox using logistic regression.*

We do apply a sigmoid function to the output object score, but use a MSE cost function for the loss from this term. It isn't clear which cost function is used in the paper. Online code seems to use MSE, which can be sub-optimal for classification problems.

If the object score isn't improved by changes to application of warmup batches, it may be useful to investigate the use of the cross-entropy (log-loss) for the objectness classification problem when using logistic regression. This should allow a smoother cost function for the objectness term.

Investigate Loss - Part 4

Some parts of our loss function may be different from those used in the YoloV3 paper.

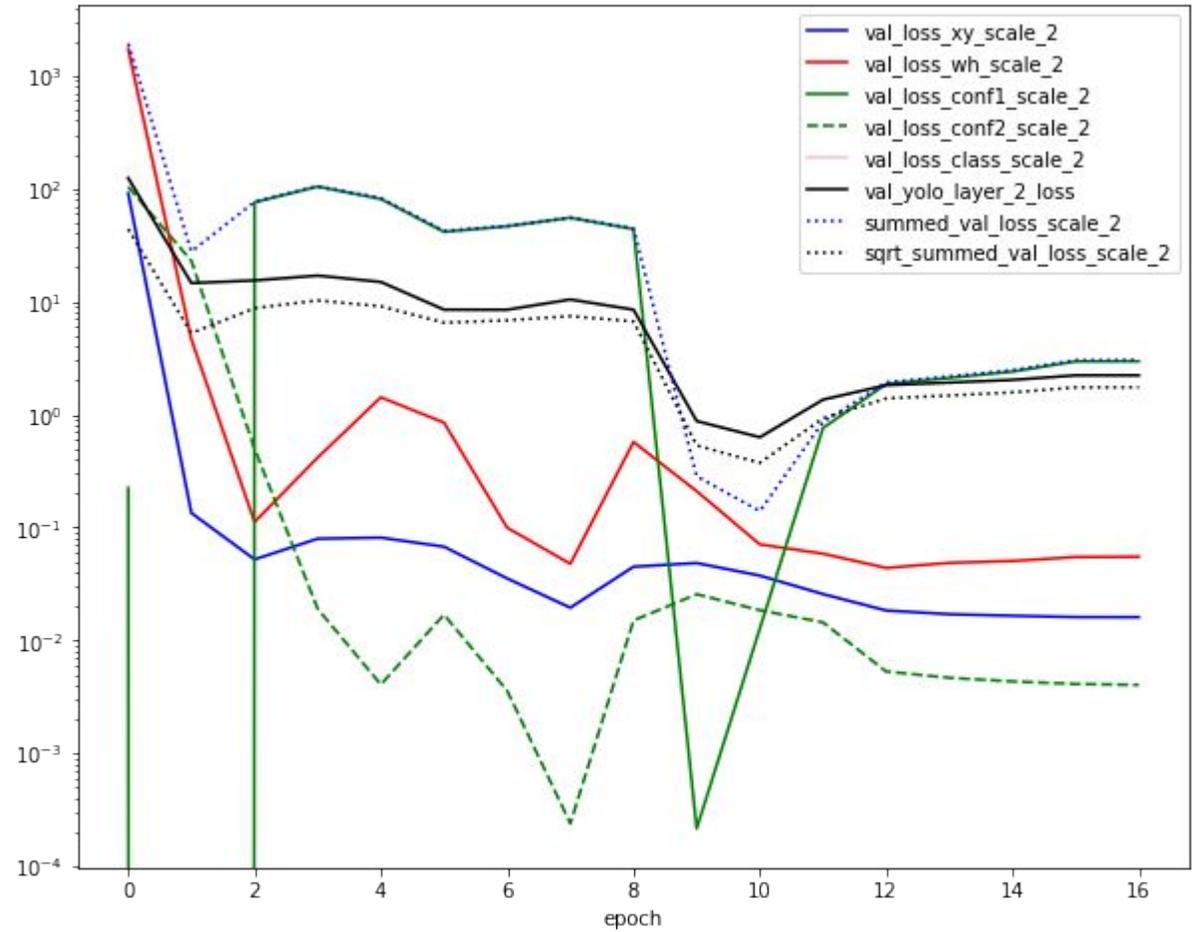
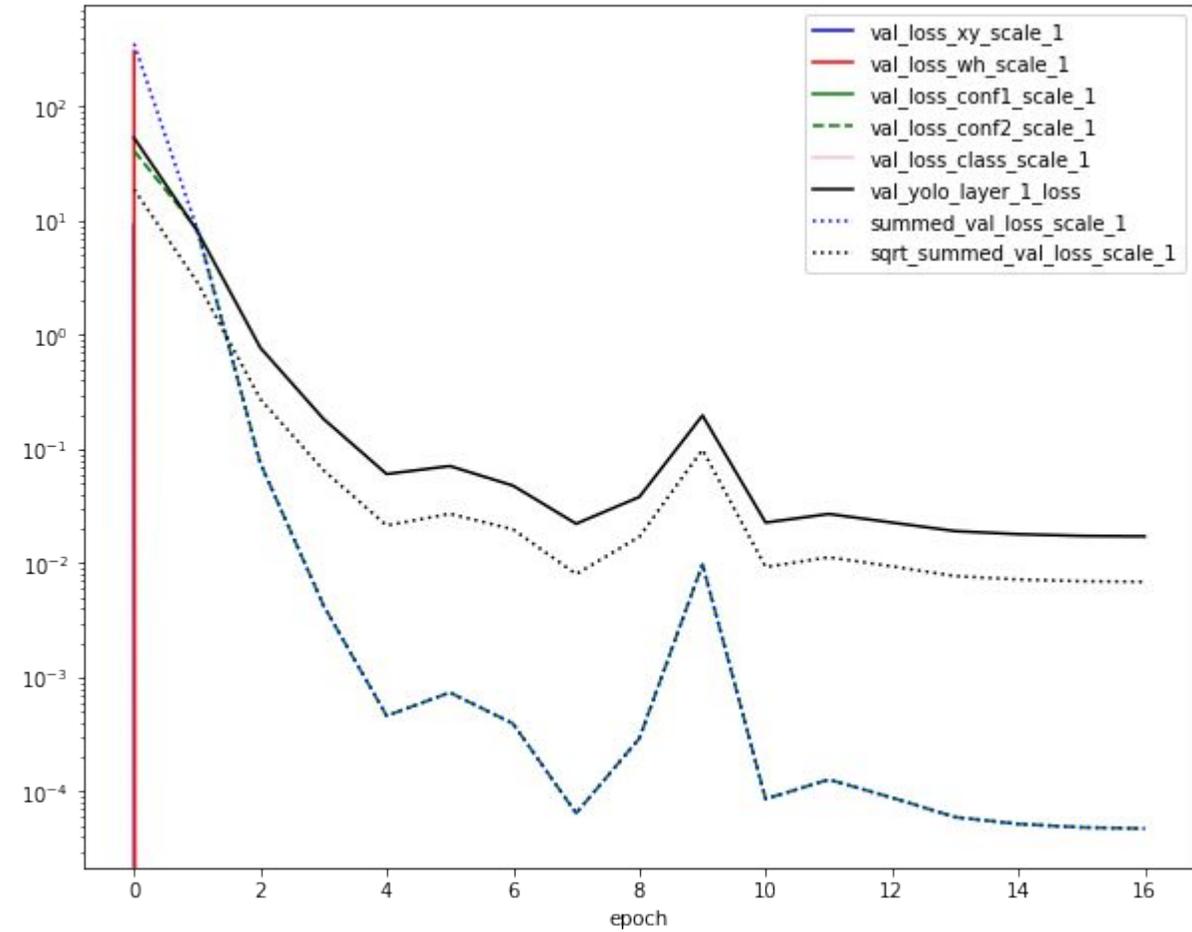
Secondly, YoloV3 paper doesn't specify a definition for the truth object score or the predicted objectness score. At the moment it is set to 1 for the anchor in a given cell grid which has the maximal overlap with a truth box. However, for some online YoloV2 implementations, this is multiplied by the IOU before the predicted object confidence is subtracted in the loss function.

Not clear in my mind if this or multiplying the sigmoid'ed output of the part of our network which is responsible for predicting 'objectness' by the IOU would be a better idea. This is how I think it is written in the YoloV2 paper though.

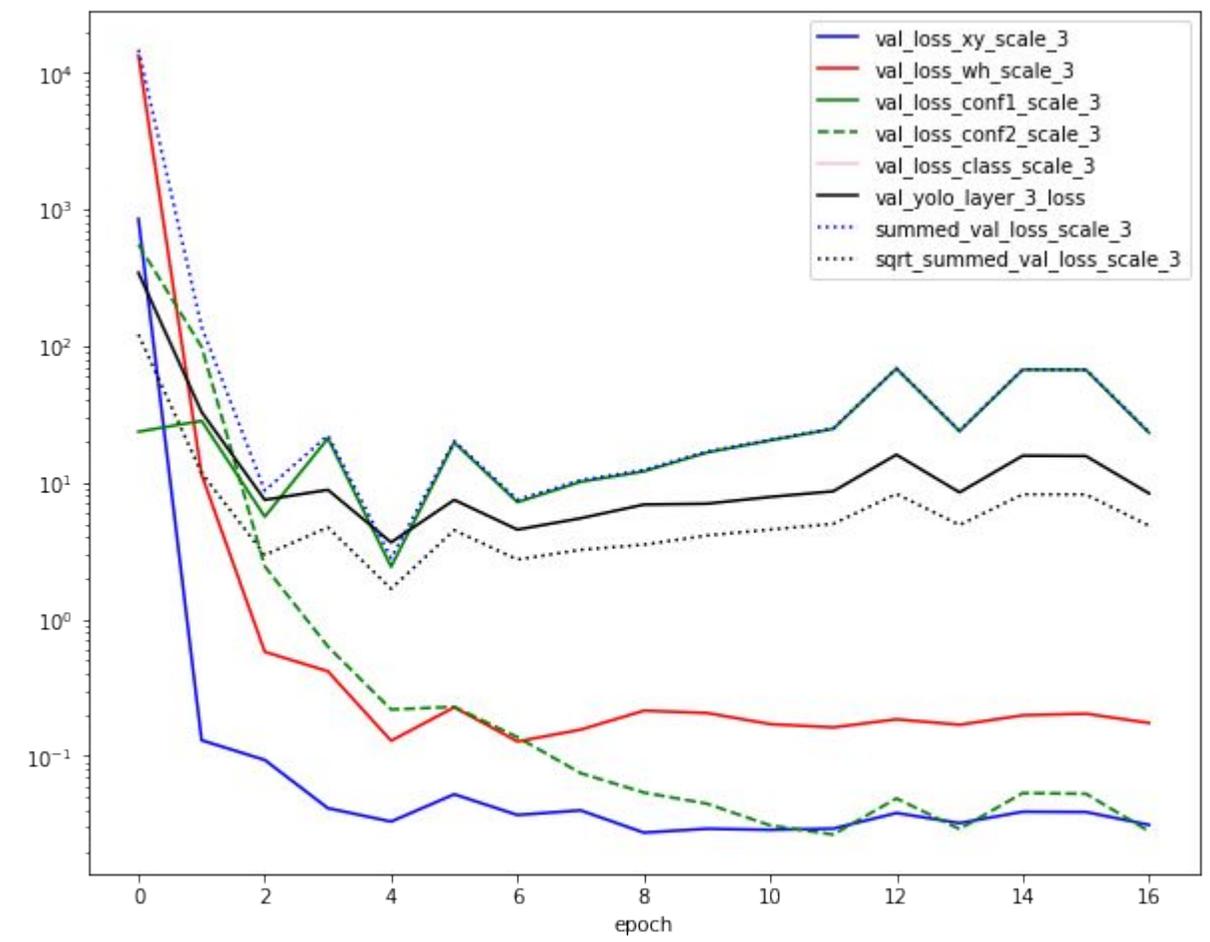
Summary

- Aggregation of the loss terms over each batch and over multiple batches is still unclear
- Warmup batches seem to occur during the first 1.05 epochs of training. I expected 2 from the values set in the code, so understanding of this has room for improvement.
- If warmup batches are for improving objectness scores before training on localization, then why do they have large contributions from the localization terms?
- Can altering the number of and contributions from warmup batches improve objectness score predictions?
- Cross-entropy might be useful to smooth out training of objectness scores
- Use of IOU with object scores might improve performance
- Also probably worth upweighting the contribution to the loss from objectness and seeing if I can get the network to learn based on that alone.

Val Loss For Each Scale -SNNu



Val Loss For Each Scale -SNNu



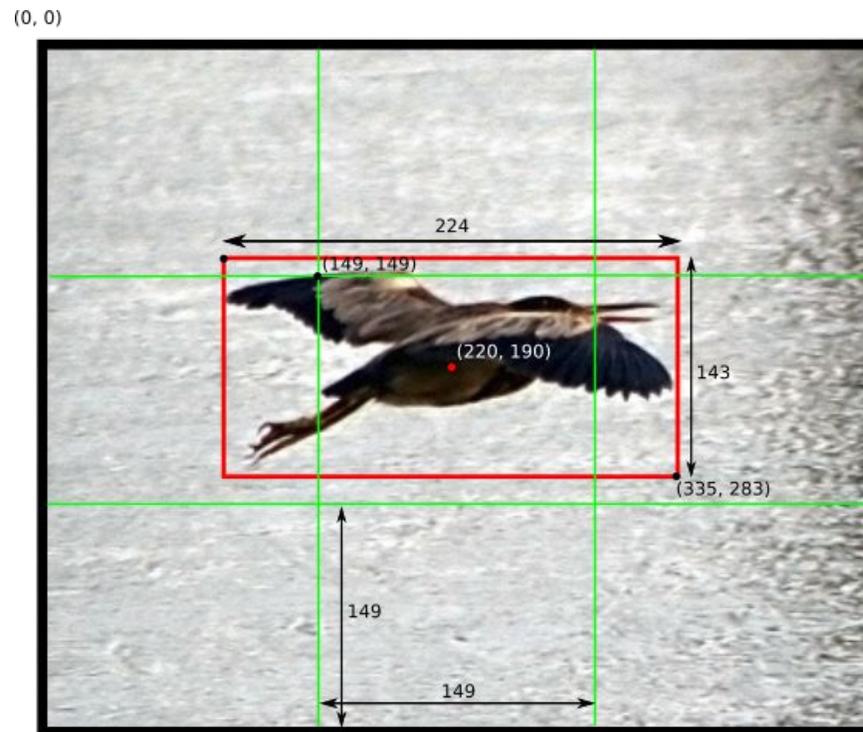
Contents

1. YOLO network
 - a. Overview
 - b. Thresholds
 - c. Loss
 - d. Truth
 - e. Metrics
2. Investigation of what the network is learning as it trains
 - a. Raccoons
 - b. SNNu

YOLOv3 Network - overview

You Only Look Once (YOLO) network predicts both the class and localization of features it identifies in an image.

- YOLOv3 looks at the image at 3 different scales
- For each scale the network divides an image into $S \times S$ grid cells (15x15, 30x30, 60x60 in our cases)
- Each grid cell makes 3 prediction boxes, each guided by a reference box for that scale
- Predictions have $(x, y, w, h, \text{obj_score}, \text{class_scores})$
- Predictions are thresholded based on the class score (we have only 1 class, so we use the object score instead)
- Then the remaining predictions from all 3 scales are recombined to get rid of overlapping predictions
- The remaining predictions are the network outputs



(447, 447)

$$x = (220 - 149) / 149 = 0.48$$

$$y = (190 - 149) / 149 = 0.28$$

$$w = 224 / 448 = 0.50$$

$$h = 143 / 448 = 0.32$$

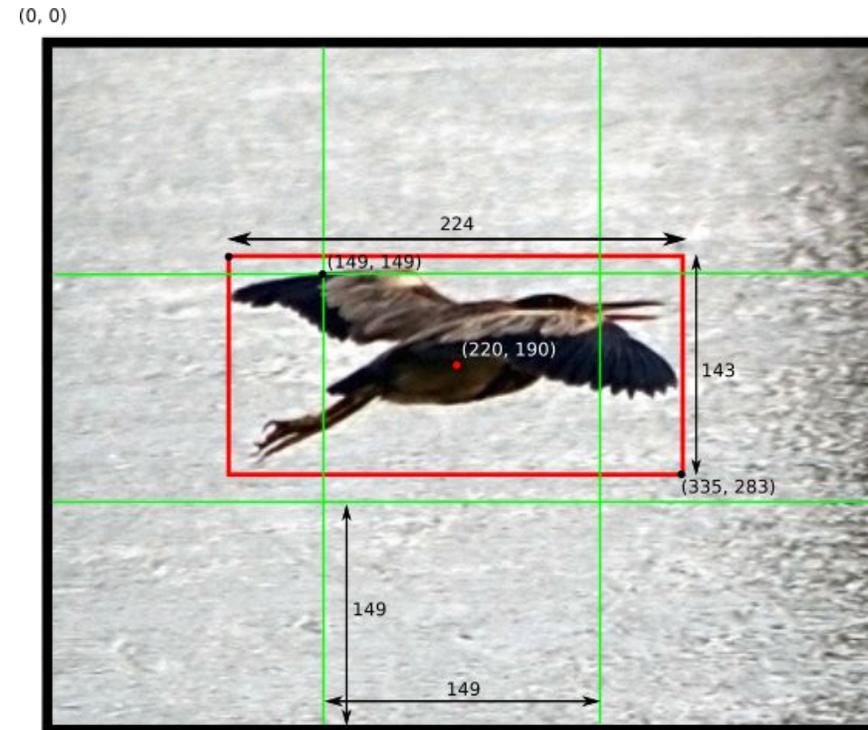
YOLOv3 Network - thresholds

Given a trained model we therefore have 2 discriminators which we can toggle to affect the performance of our network:

1. The object score threshold
2. The algorithm used to discard remaining predictions which overlap

The algorithm for discarding remaining predictions:

- Get prediction with highest object score
- Compute its intersection over union (IOU) with the other predicted boxes
- Discard boxes which have an IOU larger than a chosen threshold
- Repeat for the remaining box with the next largest object score



(447, 447)

$$x = (220 - 149) / 149 = 0.48$$
$$y = (190 - 149) / 149 = 0.28$$
$$w = 224 / 448 = 0.50$$
$$h = 143 / 448 = 0.32$$

YOLOv3 Network - loss

During training we minimise the loss, which has contributions from 4 different aspects the network is trying to optimise

1. Bbox x,y positional term, only uses the bbox prediction with maximal IOU, and is only non-zero if an object is in the cell
2. BBox w,h term, stronger contribution from larger bboxes for equivalent w,h difference to truth
3. Class prediction term, the square of Confidence - IOU
4. Classification loss term, only contributes when an object is present in the cell

S: number of cells along one axis

B: number of bboxes per cell

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

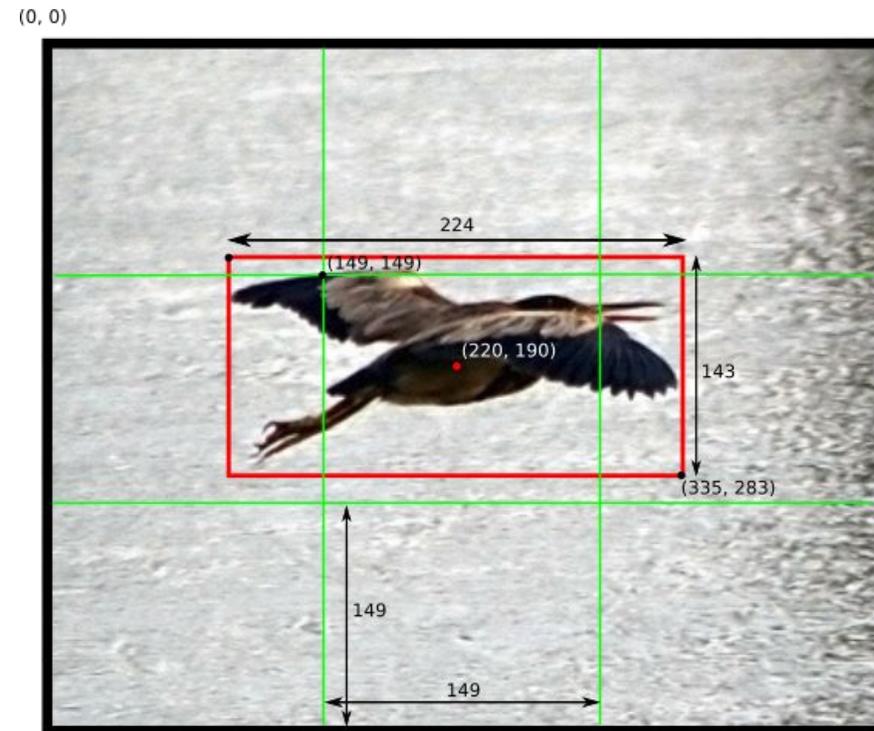
$$\sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2$$

YOLOv3 Network - truth

A truth box is a box containing information about the location of an object in an image which the network should learn to identify.

- A truth box containing a truth object is assigned to 1 scale based on which one of the reference boxes (anchors) is closest in size to it.
- It is also assigned to the grid cell at that scale which contains the centre of the truth box
- The object score and IOU with the truth box for the prediction box produced by this grid cell anchor is what the loss is designed to improve during training.
- Prediction boxes made by grid cell anchors with no associated truth box should contribute in a positive way to the loss function if they have low objectness scores and a low IOU with truth boxes



(447, 447)

$$x = (220 - 149) / 149 = 0.48$$

$$y = (190 - 149) / 149 = 0.28$$

$$w = 224 / 448 = 0.50$$

$$h = 143 / 448 = 0.32$$

YOLOv3 Network - metrics

We want to understand:

1. Whether the performance of the network is improving during training and, if not, why
2. At which epoch was the network performing best
3. What thresholds on object score and IOU of predicted boxes produce the best performance

To understand whether performance of the network is improving during training, we can look at the average object scores for:

- predictions which are made using an anchor in a grid cell which has an associated truth box
- predictions made using an anchor in a grid cell which has no associated truth box

We can also look at the distribution of object scores for these two cases

YOLOv3 Network - metrics

We can also look at the IOU with truth boxes of these 2 cases. Note that this is a different concept than the IOU of predicted boxes with one another!

We can use the IOU of predicted boxes with truth boxes to evaluate whether the network learns to predict boxes with the right size and locality. We can look at the distribution and might want to set a minimum threshold on this when evaluating false positive rate, recall and precision.

In future we could potentially determine the % of the SADC of the signal from a SNNu within the prediction box.

In order to determine good object score and IOU score thresholds, we need to look at the recall, precision and false positive rate of the network

- Recall: proportion of truth boxes which are identified
- Precision: proportion of positive predictions which are true positives

Network Training Investigation

We are currently trying to understand what is happening during training. We are still looking only at metrics computed from a single batch (8 images) in this presentation.

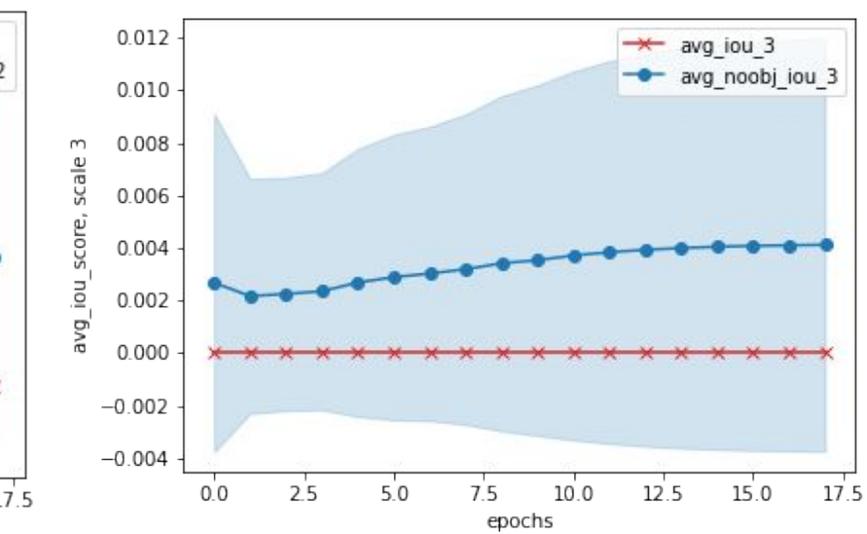
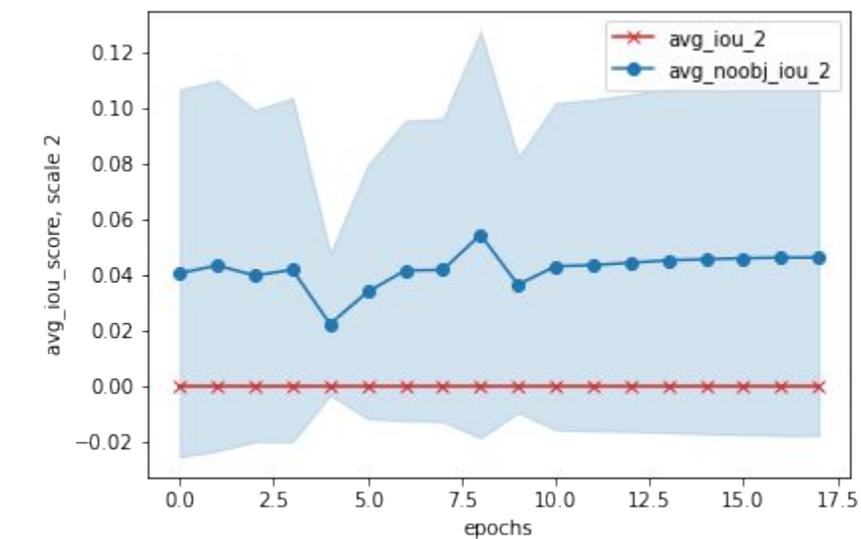
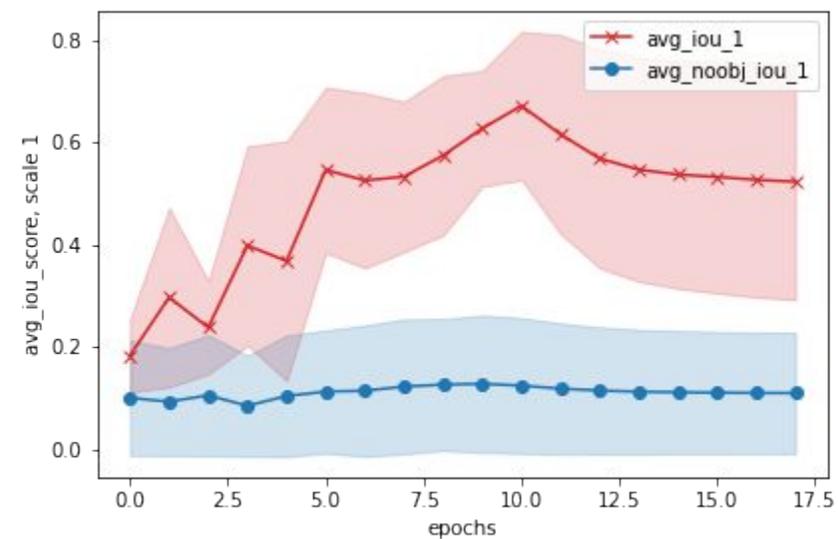
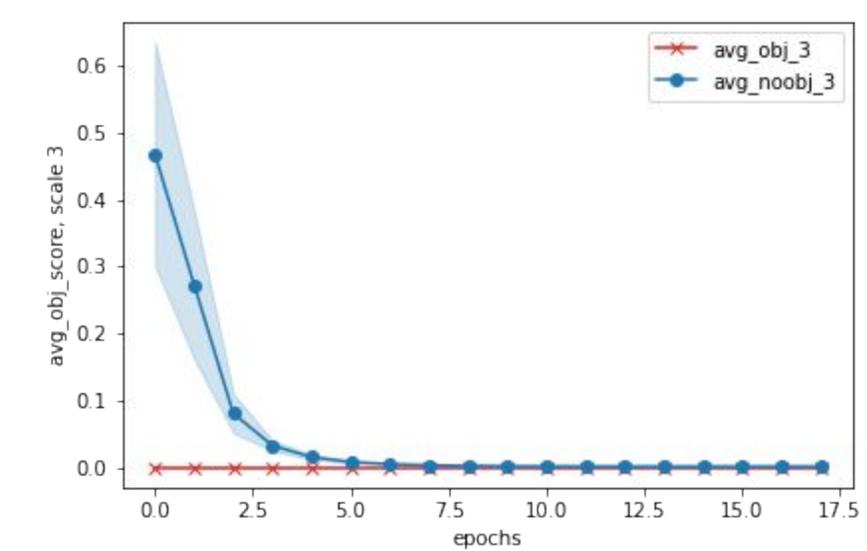
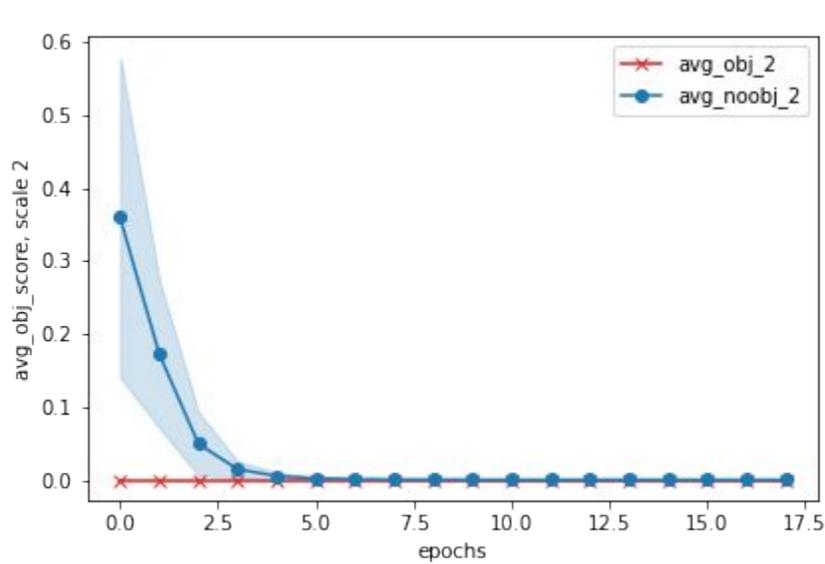
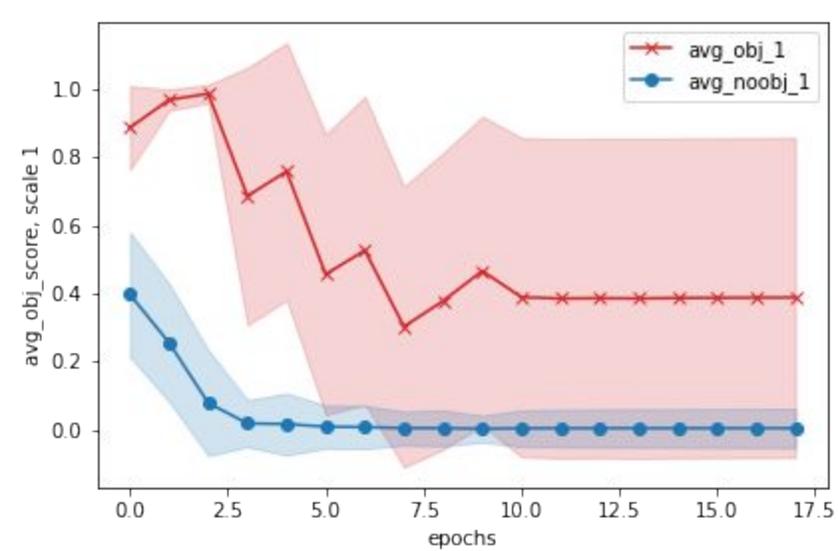
We expect the object scores from predictions with associated truth boxes to be higher than for predictions without associated truth boxes, and would naively expect the separation between the two distributions to increase as the network trains and its loss is minimized.

The same is expected for the IOU of predicted boxes with truth boxes.

Network Training On Raccoons

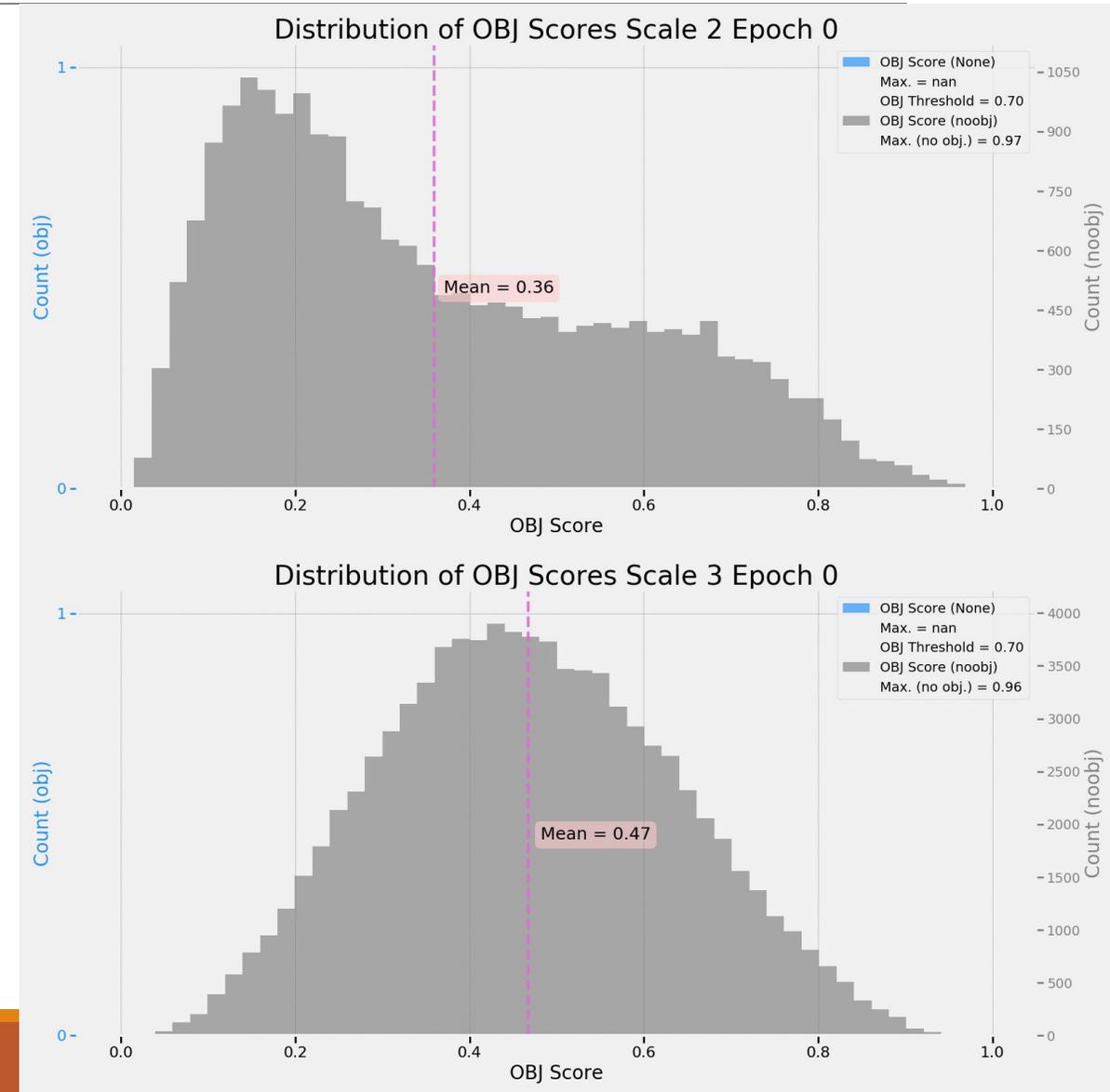
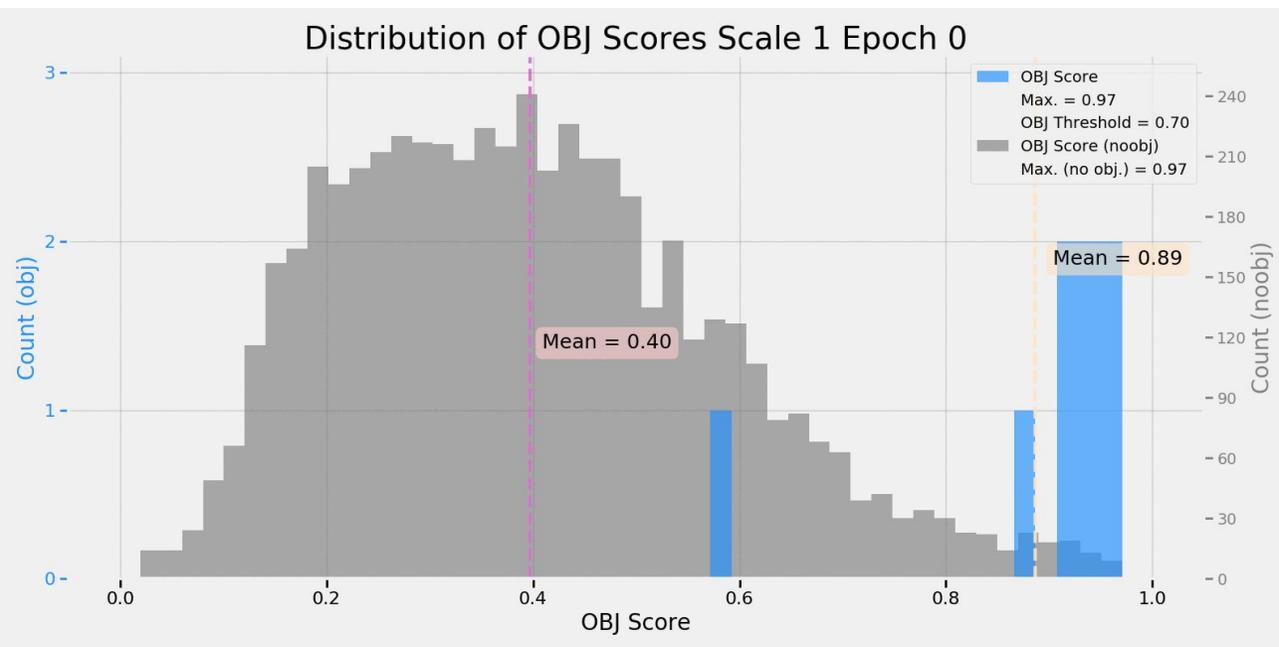
- We have a total of 8 truth boxes in this dataset (1 per image in the batch).
- All the truth boxes are closest in size to a large anchor box and therefore are associated with scale 1 (the coarsest scale - with the fewest grid cells).

On the following slide we plot the average object score and iou score for predictions from all 3 scales, along with error bars of 1 standard deviation.



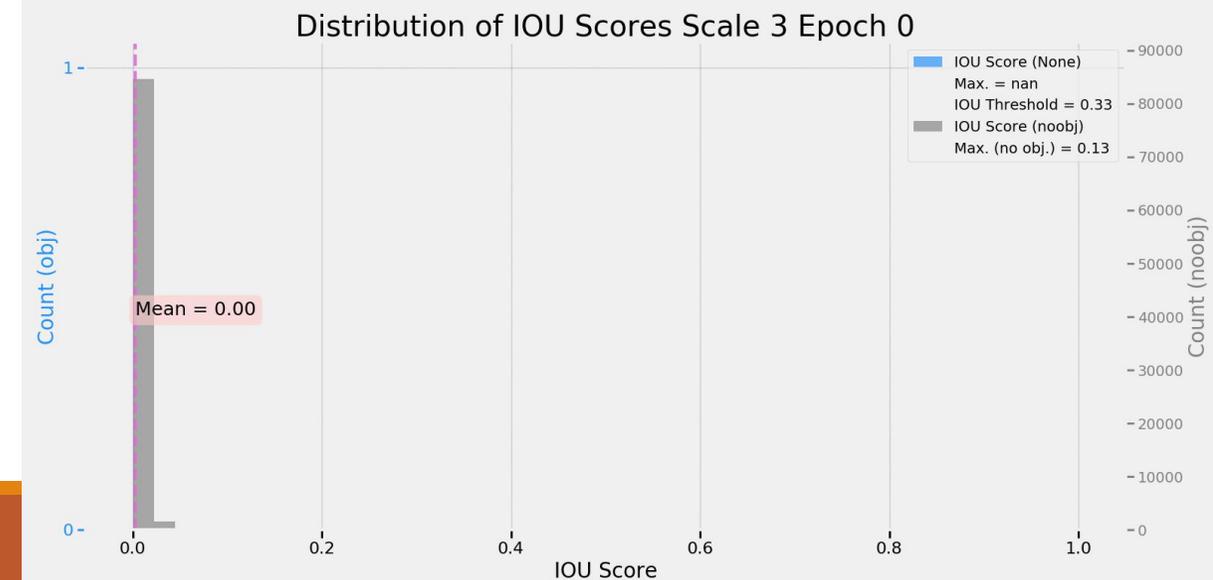
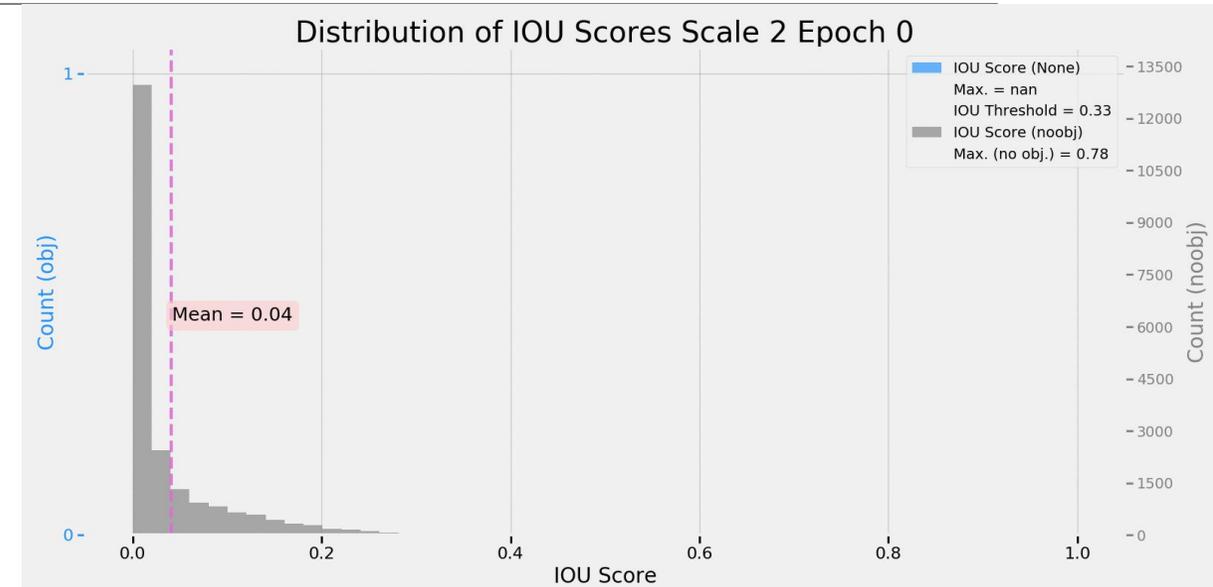
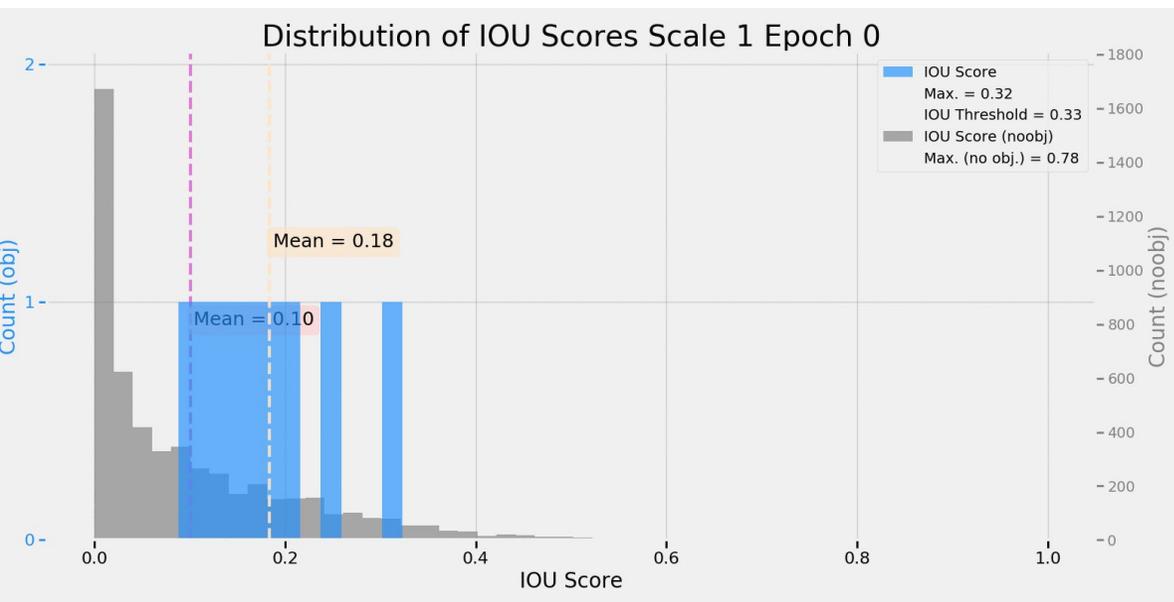
Network Training On Raccoons

What do the distributions of object scores and iou scores look like as we train for predictions from each scale?



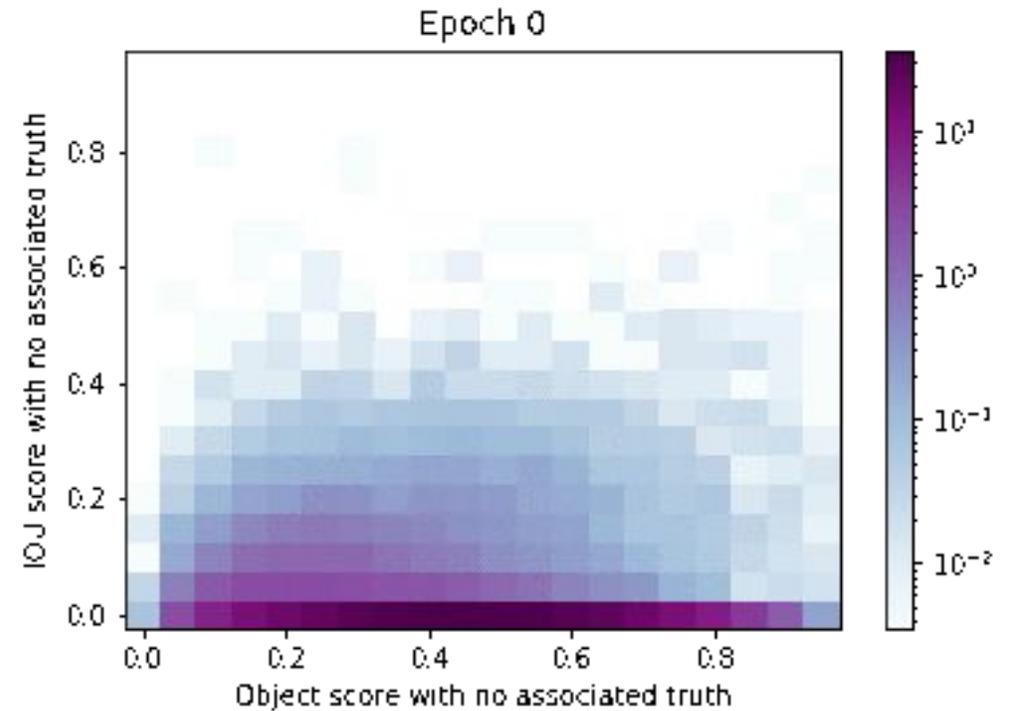
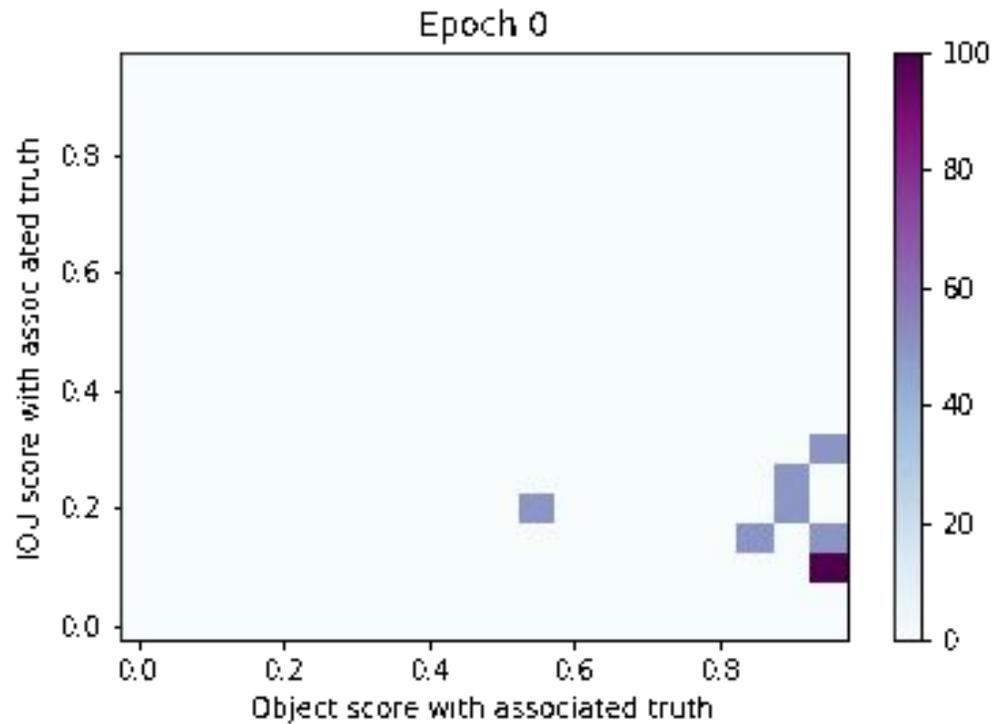
Network Training On Raccoons

What do the distributions of object scores and iou scores look like as we train for predictions from each scale?



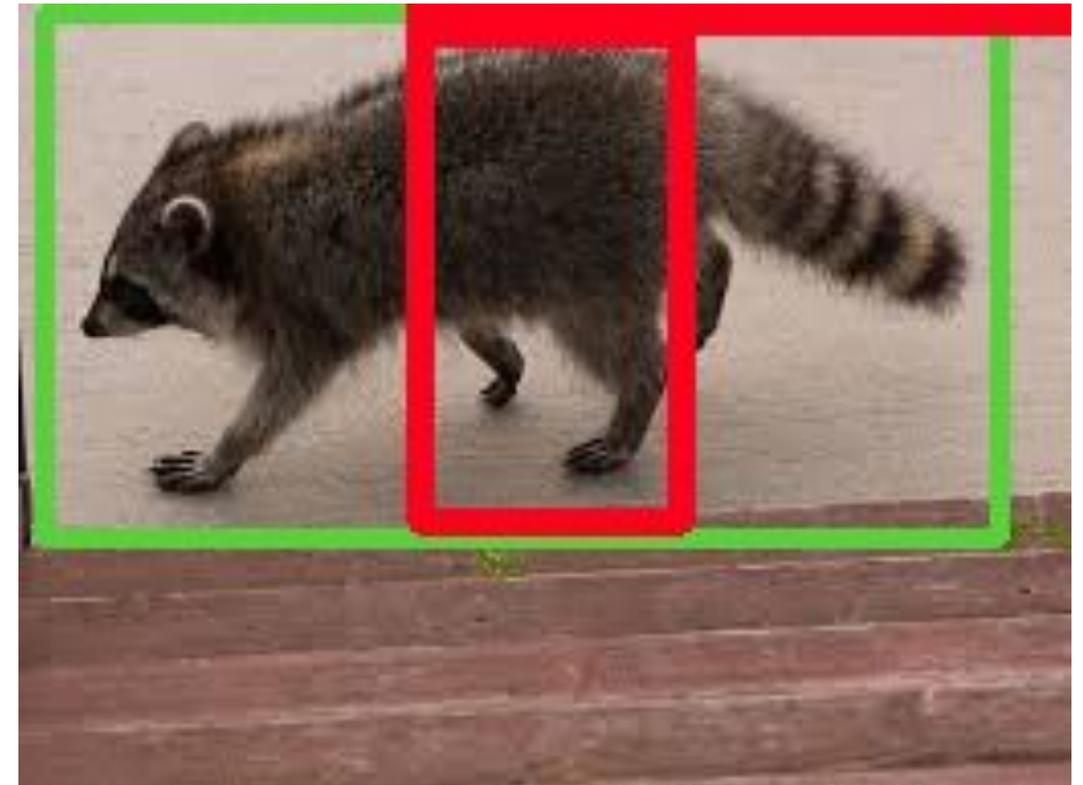
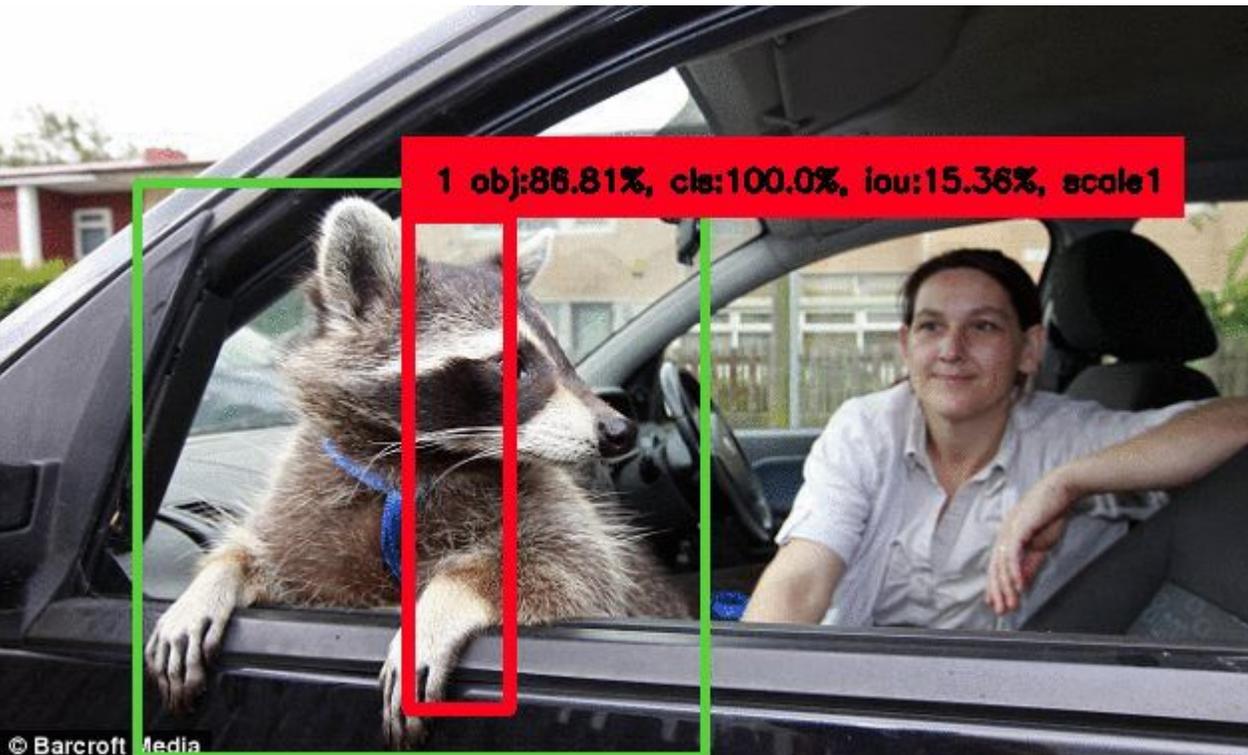
Network Training On Raccoons

What is the spread of object scores and iou scores for all predictions from all scales?



Network Training On Raccoons

Let's look at a couple of examples for the prediction boxes with associated truth boxes throughout training



Network Training On Raccoons

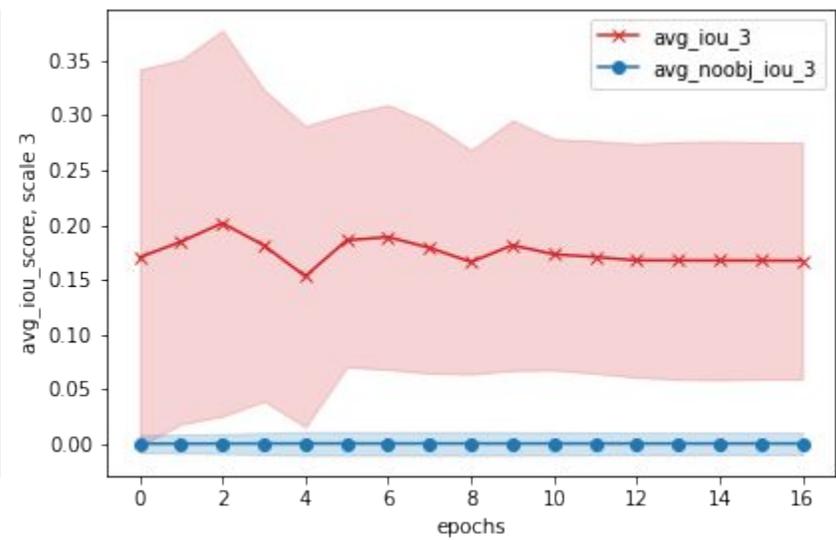
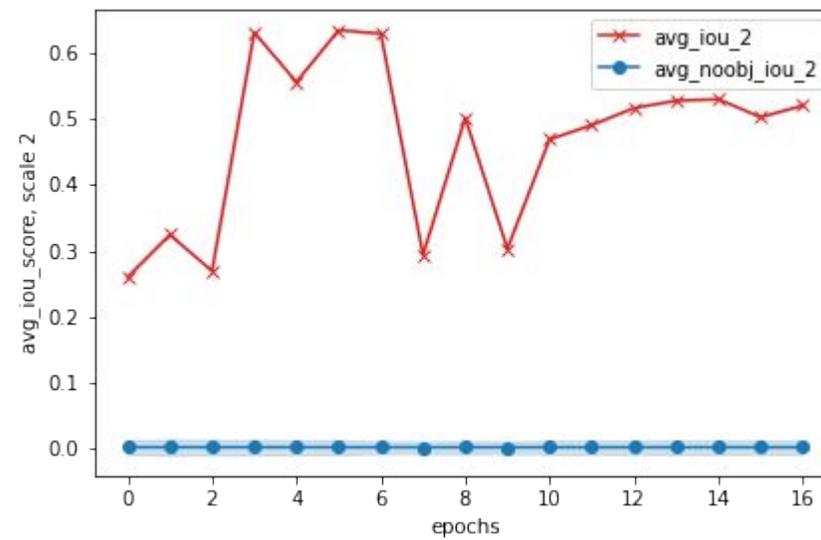
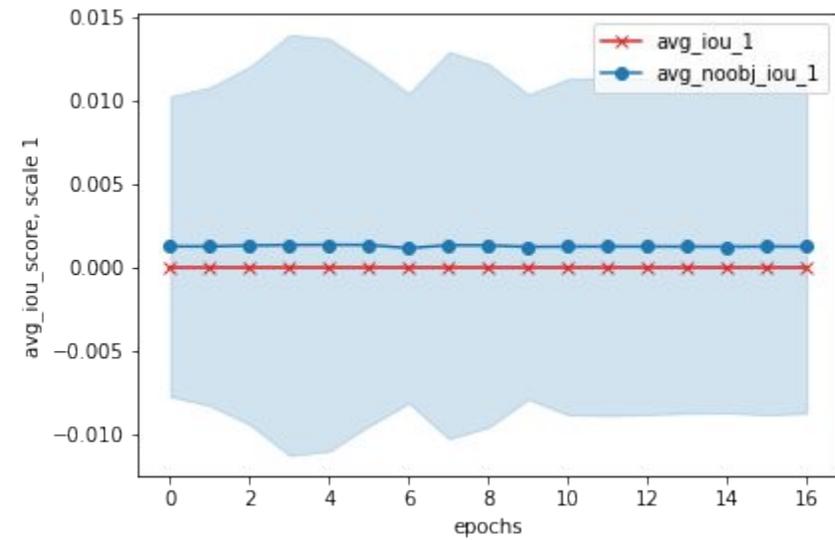
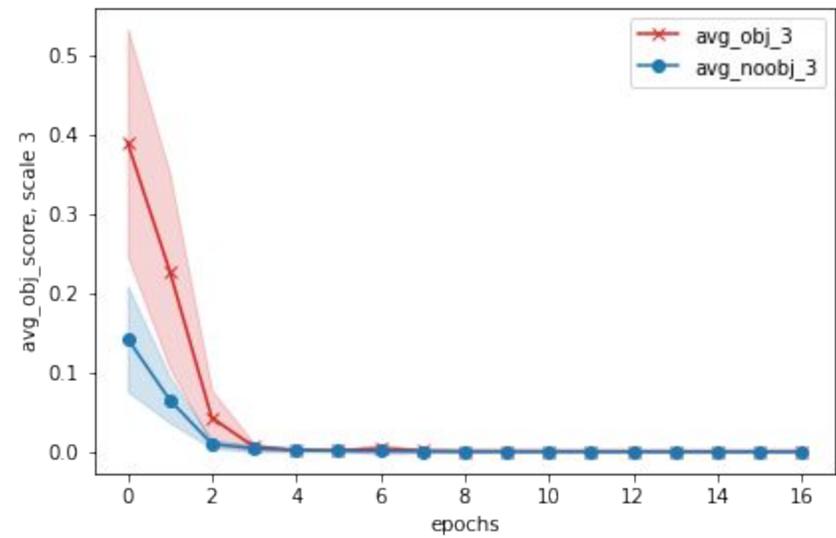
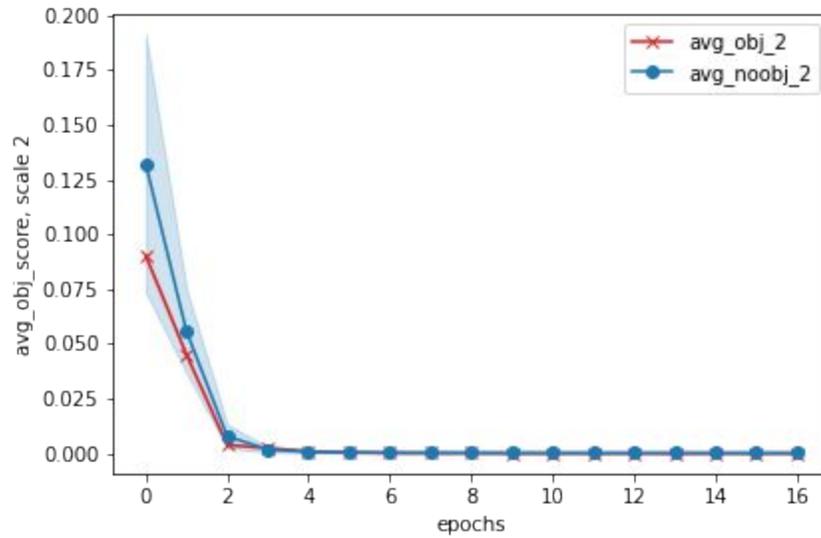
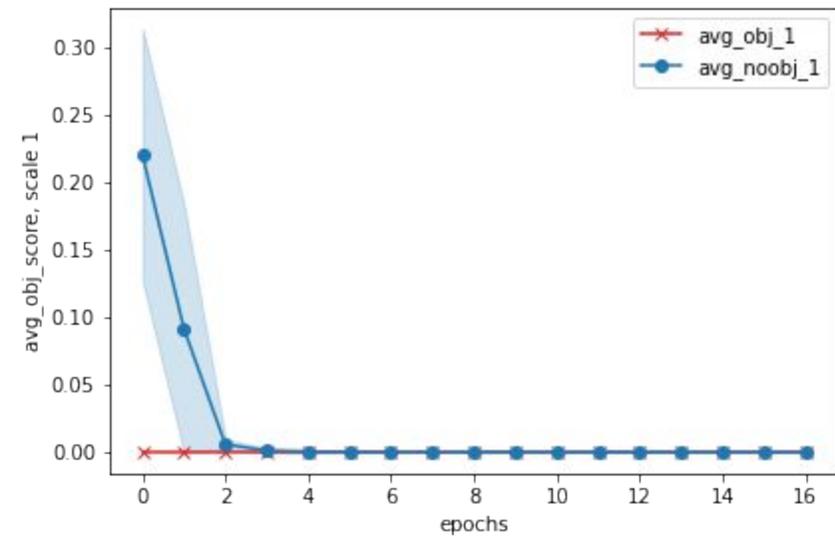
- The average OBJ scores for boxes with and without associated truth boxes are within 1 std deviation in later epochs because the average OBJ score for predictions with associated truth decreases after the 3rd epoch. Std deviation might decrease with the use of more batches to compute these metrics
- The OBJ scores for predicted boxes with associated truth boxes appear to separate well from those without associated truth boxes when looking at the histogram for scale 1. For scales 2 and 3 the object scores for predictions quickly reduce to near 0.
- However, when we plot the OBJ score vs the IOU score, we see a number of examples where the OBJ score is large when we use a log scale. These examples tend to have IOU scores which are similar to those of the predictions which do have an associated truth box. Perhaps they are made by other anchor boxes from the same scale in the same/adjacent grid box. Some of these predictions might be removed when applying an IOU threshold for the overlap of predicted boxes
- The IOU score distribution for predictions with no associated truth box appears to have a longer tail for predictions made by scale 1 of the network. The tail is also longer for scale 2 than scale 3. This makes sense as scale 2 and 3 make predictions whose size guided by smaller bounding boxes.

Network Training On SNNu

Do we see something similar when we train on the SNNu dataset? The main differences between this and the raccoons dataset are the increased sparsity and the different downsampling method.

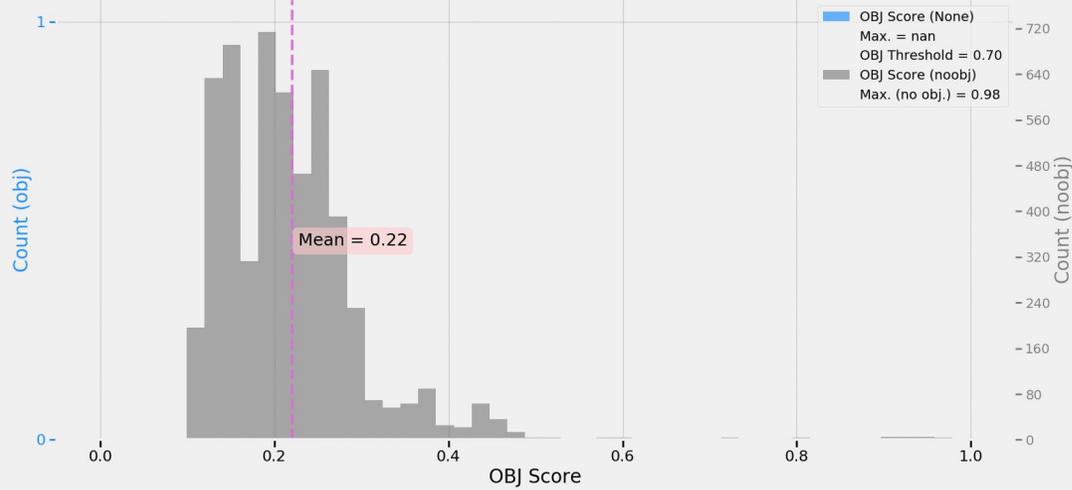
- We have a total of 12 truth boxes in this dataset
 - 1 for scale 2, the medium granularity
 - 11 for scale 3, the finest scale

Once again, on the following slide we plot the average object score and iou score for predictions from all 3 scales, along with error bars of 1 standard deviation.

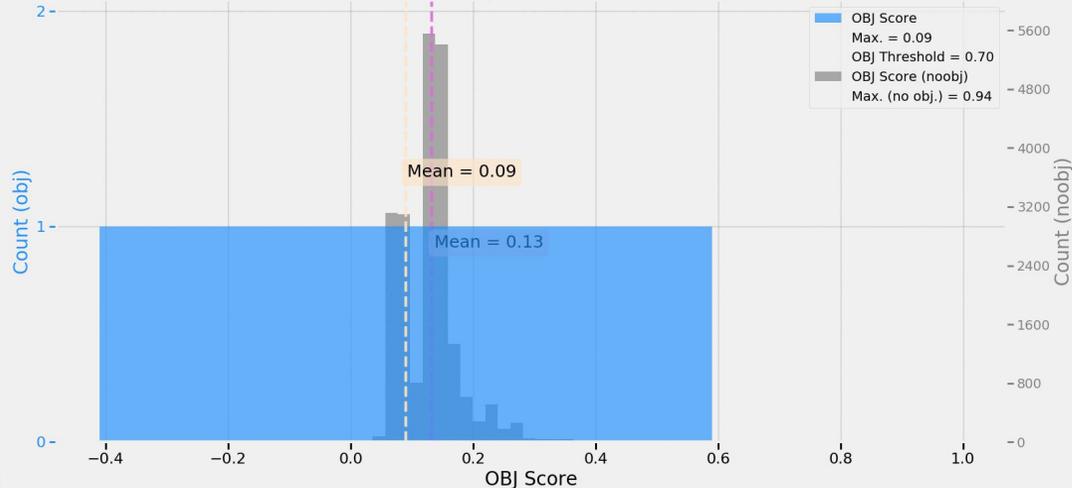


Network Training On SNNu

Distribution of OBJ Scores Scale 1 Epoch 0



Distribution of OBJ Scores Scale 2 Epoch 0

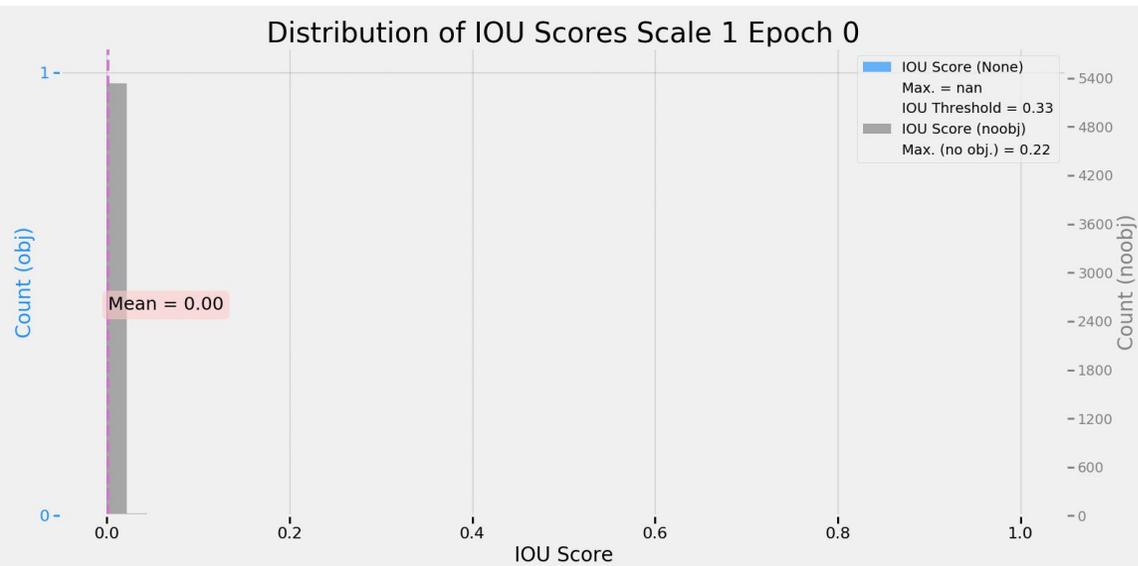


What do the distributions of object scores and iou scores look like as we train for predictions from each scale?

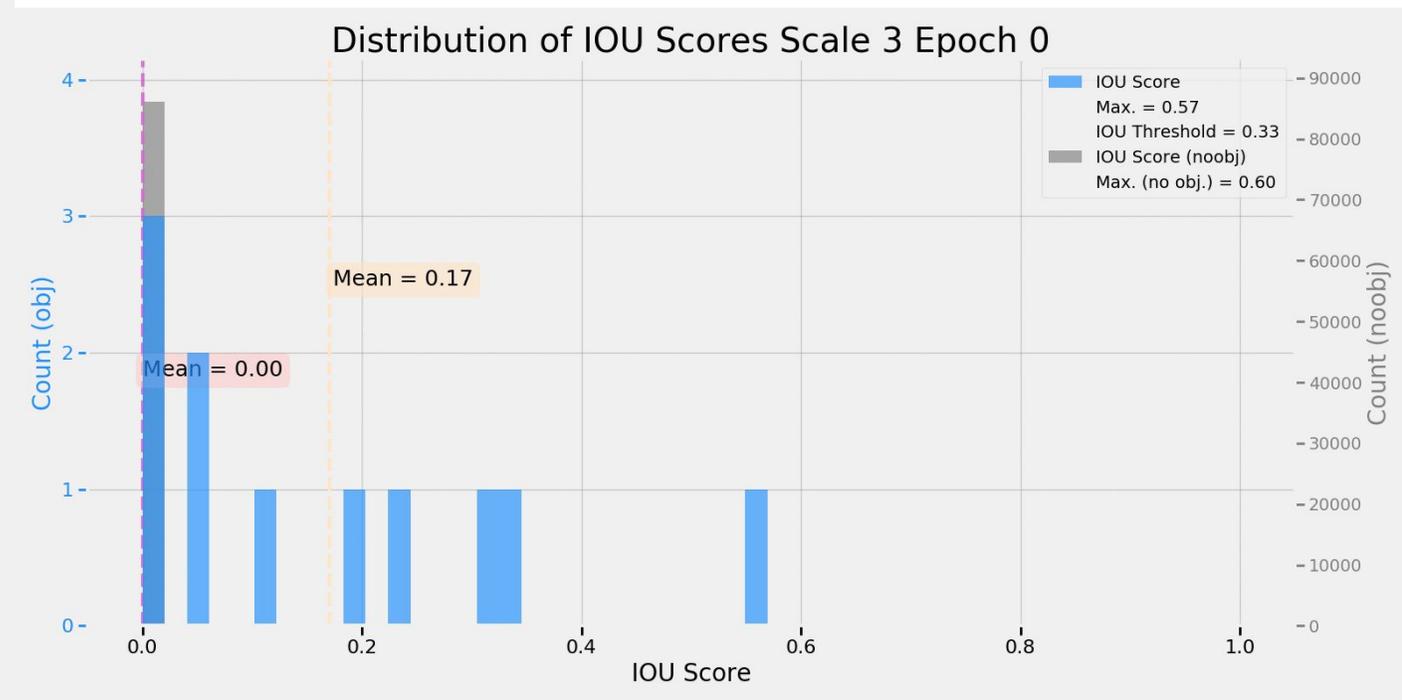
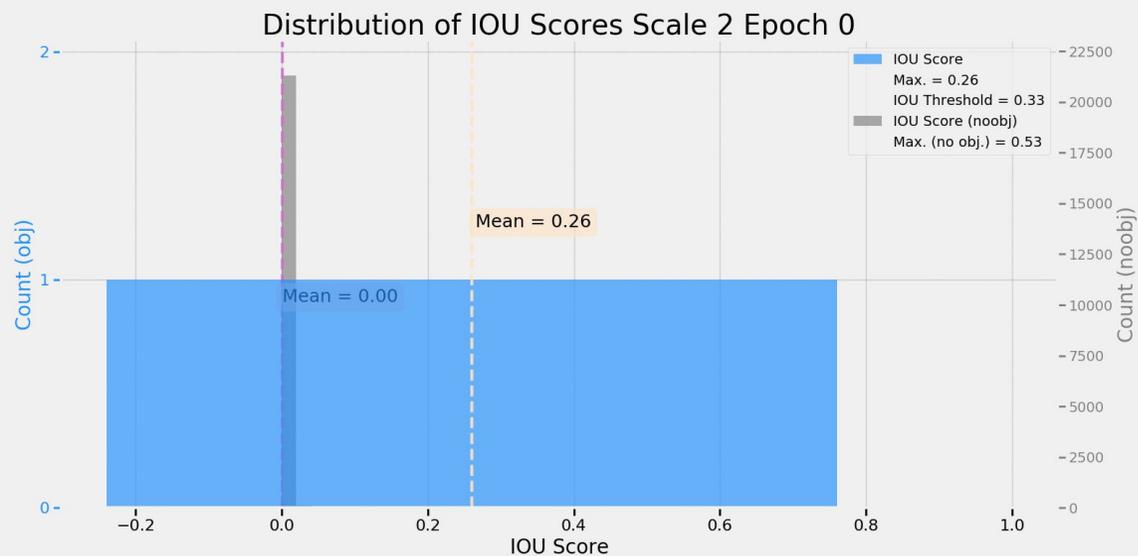
Distribution of OBJ Scores Scale 3 Epoch 0



Network Training On SNNu

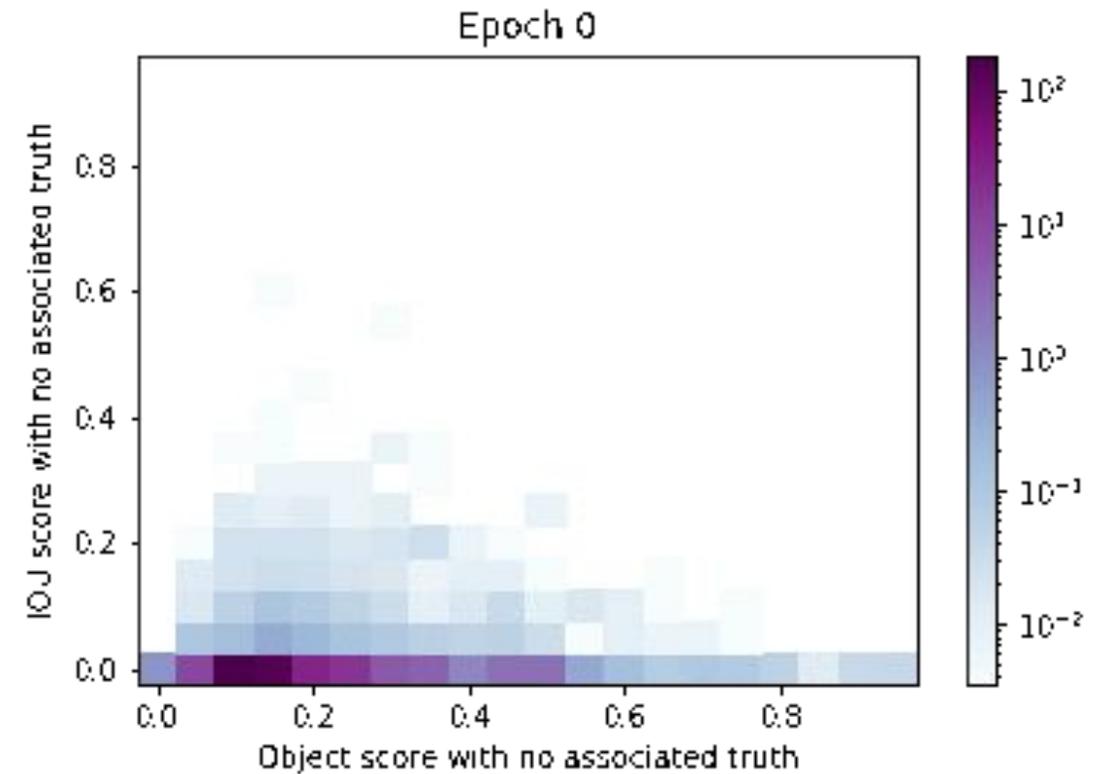
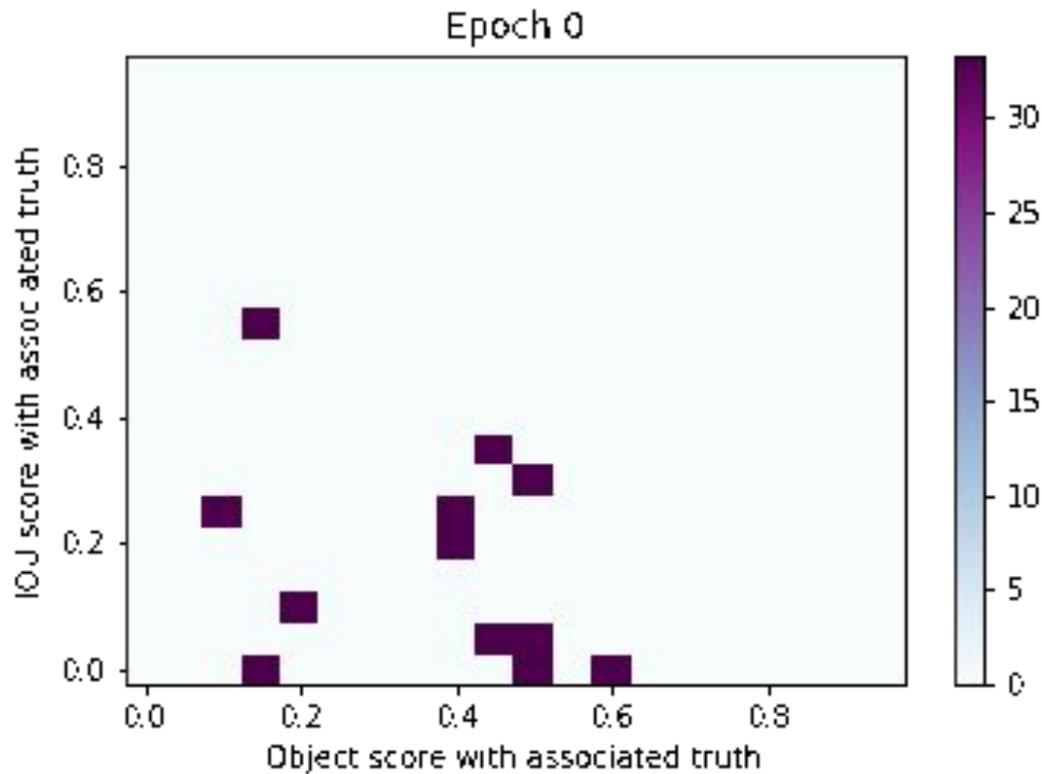


What do the distributions of object scores and iou scores look like as we train for predictions from each scale?



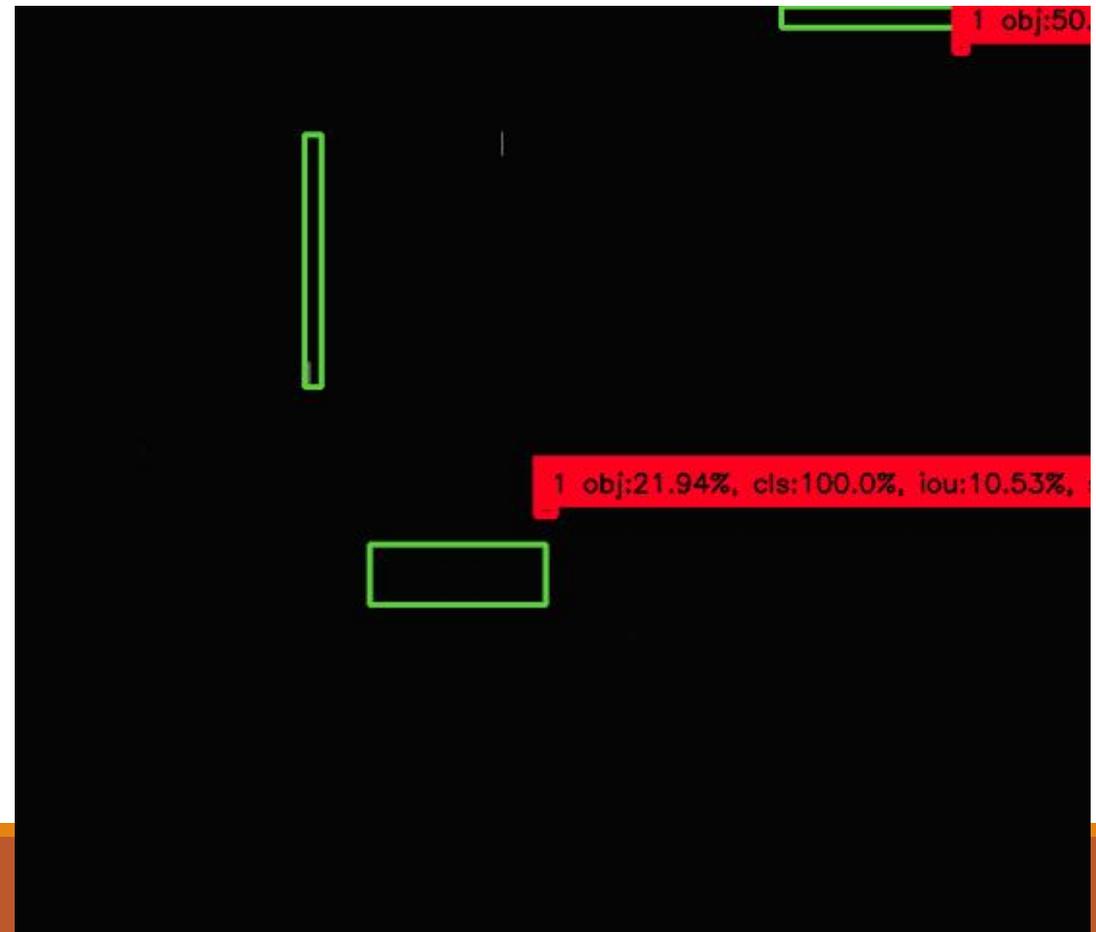
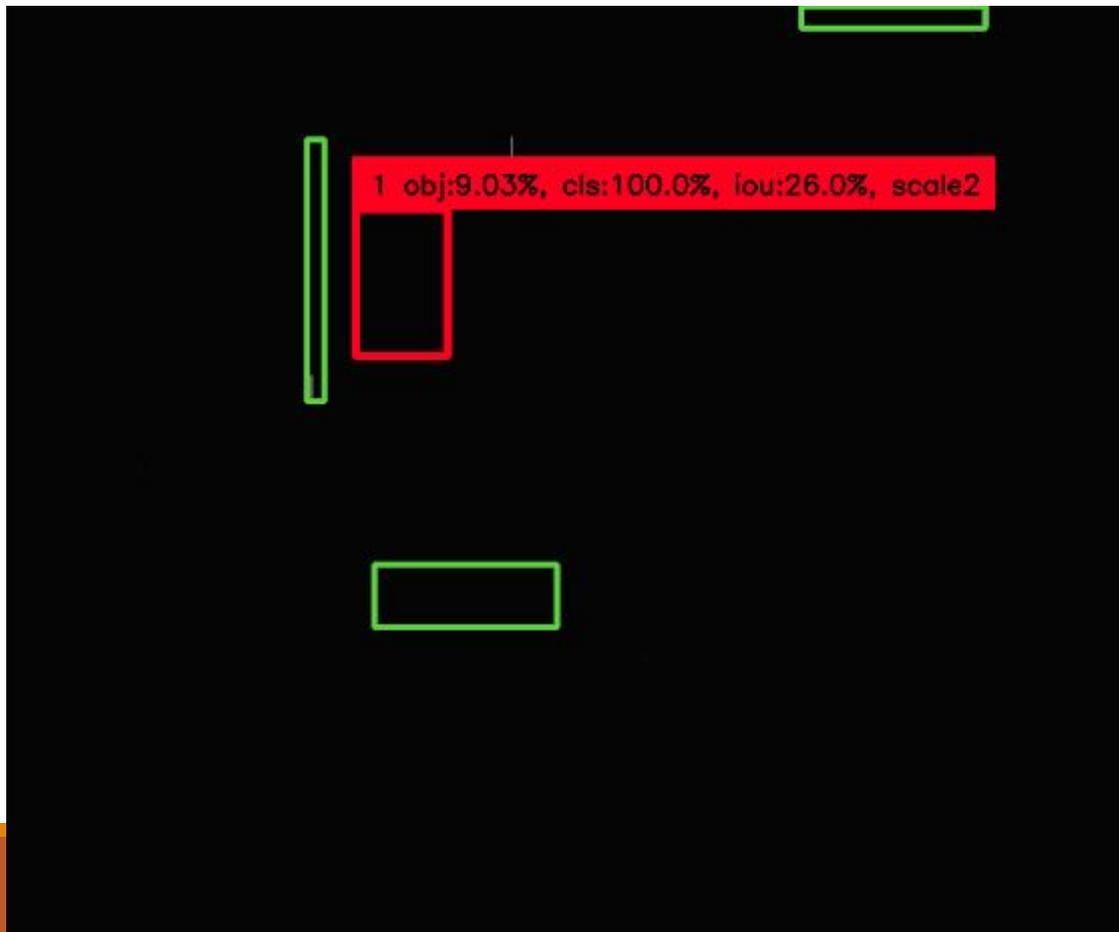
Network Training On SNNu

What is the spread of object scores and iou scores for all predictions from all scales?



Network Training On SNNu

Let's look at prediction boxes which have an associated truth box for an image which has 1 truth box associated with scale 2 and 2 truth boxes associated with scale 3



Network Training On SNNu

- The average OBJ scores for boxes with and without associated truth boxes rapidly decrease to be comparable to predictions with no associated truth box
- The OBJ scores for predicted boxes with associated truth boxes rapidly become similar to those from predictions without associated truth boxes for both scales 2 and 3
- The IOU scores for predictions with associated truth boxes have a higher average than those without for scales 2 and 3
- Plotting the OBJ score vs the IOU score, doesn't appear to give use anything else (at least with bin widths of 0.05).

Summary

- We've looked at how the object score and IOU scores change for predictions which do have an associated truth box and those which do not for raccoons and SNNu datasets.
- For both datasets our main problem is that the object scores for predictions which we know should produce high objectness scores (when the network is functioning/learning correctly) are very low.
- Even more confusingly, they start high. Perhaps we should try upweighting the contribution of the object score to the loss function.
- Our network's performance depends heavily on the object score. It doesn't bode well for performance if this isn't optimised during training. We could look at the FPR, precision and recall if we can identify a 'best epoch' which it would be worth doing this for.
- Will likely get better insight when we compute these metrics over many batches instead of just 1
- It is possible that IOU scores appear more reasonable than objectness scores for predictions which have an associated truth box just because they are guided by a grid cell location and a guide size (from the anchor box)

YOLOv3 Network - progress + to do

- Added calculation of IOU score for predicted boxes with no associated truth box
- Added distributions of prediction obj and iou with truth scores split by scale and whether a truth box is associated with the prediction
- Added output of the truth boxes and predicted boxes during each epoch to allow us to determine performance metrics external to our training code

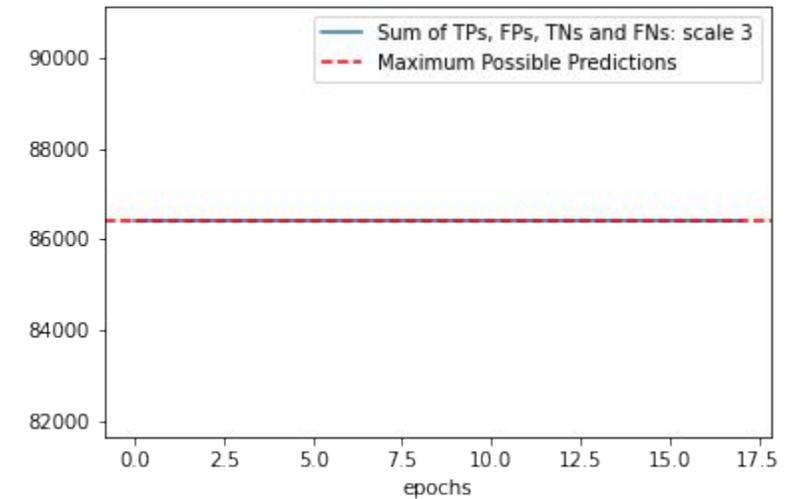
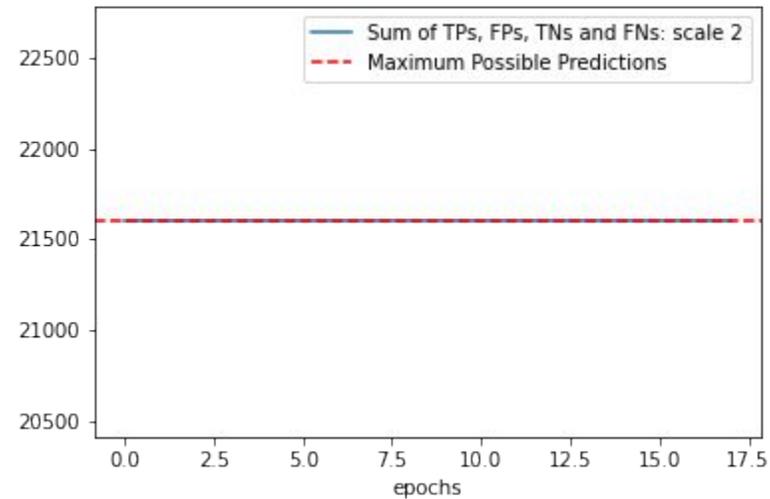
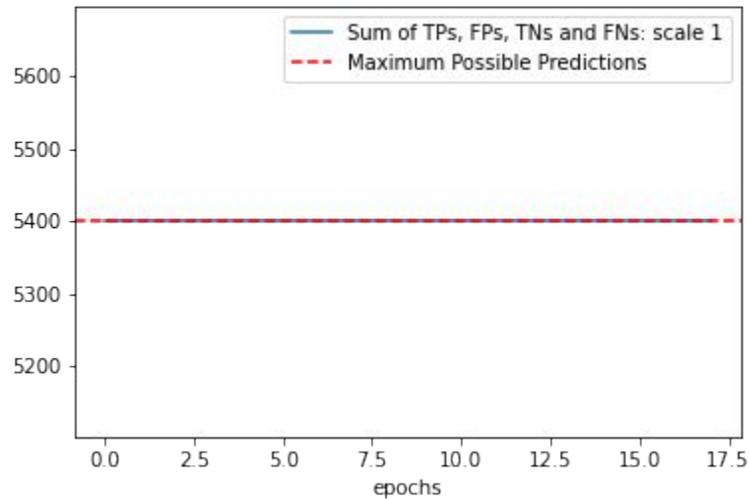
To do:

- Look at these metrics for the output of training on bottles dataset
- Use output truth and predicted boxes to determine FPR, precision and IOU for different object score thresholds and no other cuts
- Do the same but in addition for a range of thresholds on the IOU of predicted boxes when we discard overlapping predictions
- Add log scale (and single y axis with normalization) to obj and IOU distribution hists DONE
- Bug fix to allow saving of images of predictions made which have no associated truth box during each epoch of training
- Extend metrics callback over multiple batches
- Reproduce the same plots and distributions with higher statistics from multiple batches
- Investigate the magnitude of different contributions to loss during training

Network Training On Raccoons

Counting the total number of predictions made for each scale. We should have:

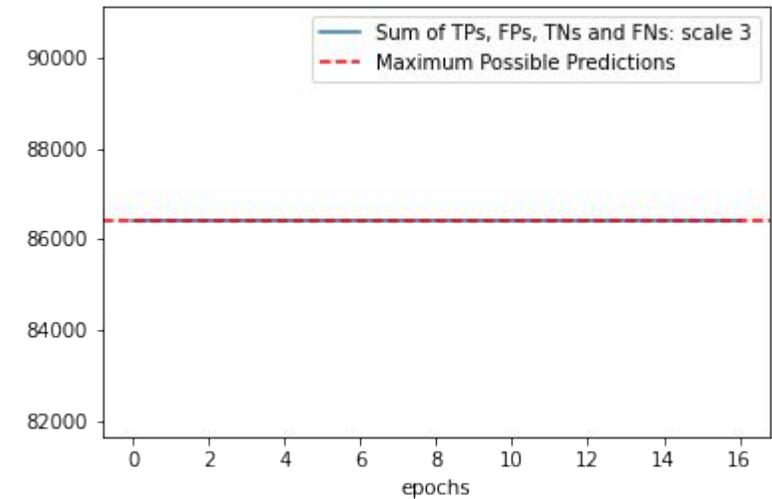
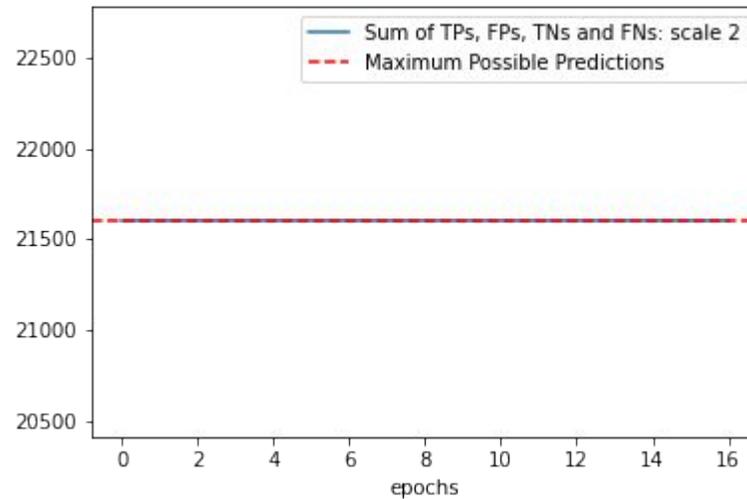
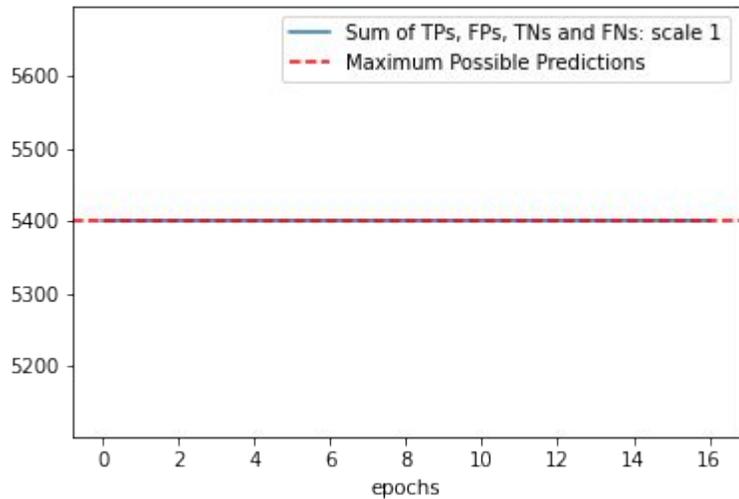
- 5400 for scale 1
- 21600 for scale 2
- 86400 for scale 3



Network Training On SNNu

Counting the total number of predictions made for each scale. We should have:

- 5400 for scale 1
- 21600 for scale 2
- 86400 for scale 3



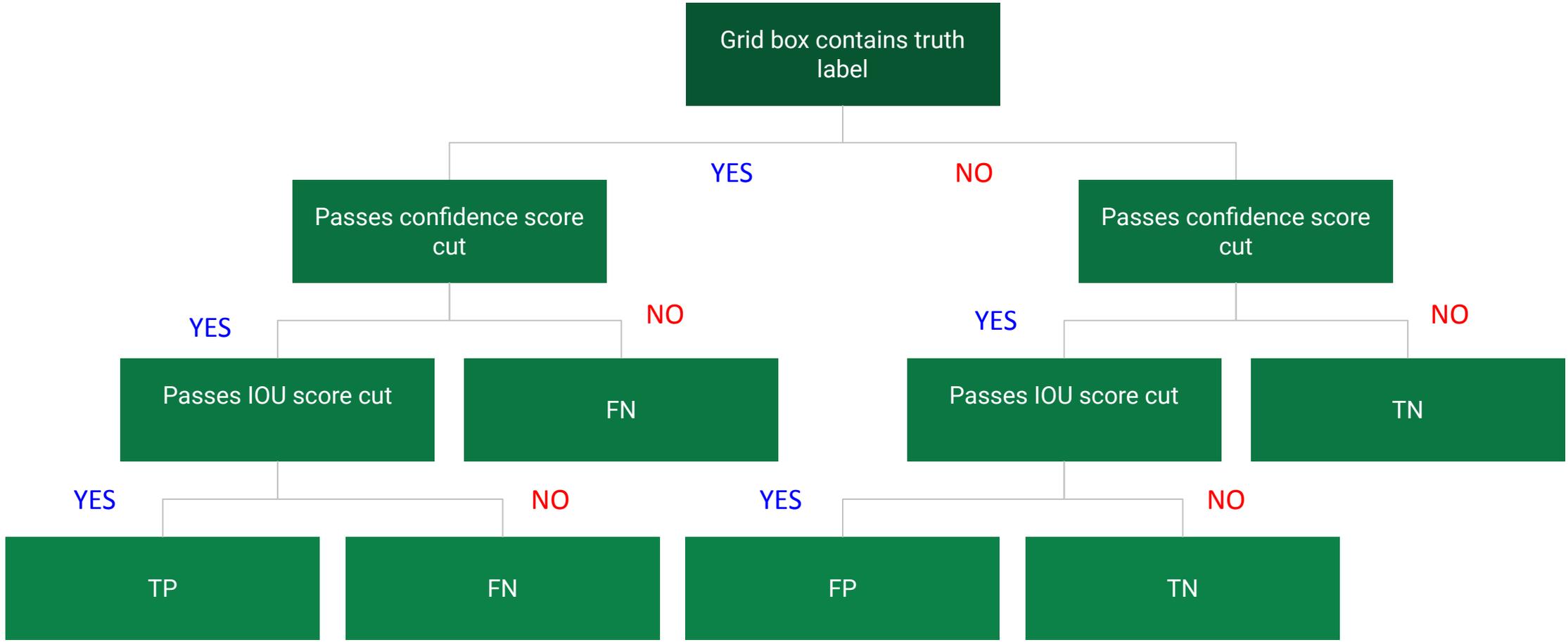
Network Training On Bottles

Network Training On Bottles

Counting the total number of predictions made for each scale. We should have:

- 5400 for scale 1
- 21600 for scale 2
- 86400 for scale 3

Metrics Flowchart



Predictions out of training

All boxes that passed the obj. conf. cut need to be ordered in descending order of obj. score

For each predicted box:

Passes object confidence score cut?

YES

NO

Is the box with highest obj. score of all unassigned boxes?

YES

NO

Overlap with unassigned truth box > 0?

YES

NO

Assign truth box to this predicted box

Passes IOU (with truth box) threshold?

YES

TP

NO

FP & go to start

IOU with better boxes less than NMS threshold?

YES

NO

Overlap with unassigned truth box > 0?

YES

More boxes exist to assign to truth?

YES

NO

FP & FN for unassigned truth box

NO

Discard box and look at next box

Box(es) exist

Go to start

NO boxes exist

Truth box(es) exist?

YES

NO

FN for each truth box

TN

Summary + Next Steps

We think we understand how the network is reading the truth information on a per-batch level. We have run our new callback with a small dataset of 8 images only to test it and have looked at the output metrics.

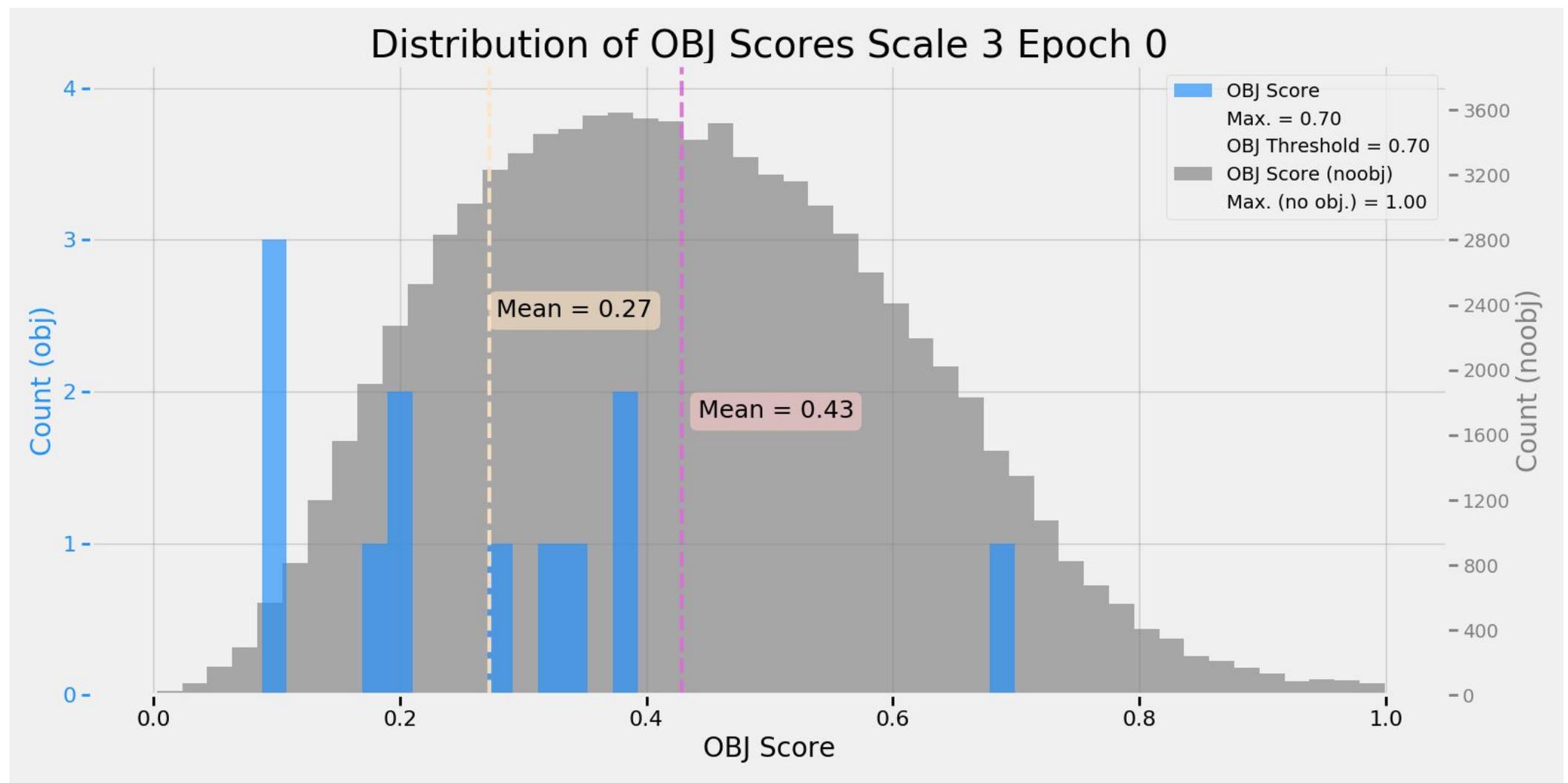
We are going to ensure we understand what is going on for individual images by plotting the predicted box and the truth box associated with it on the image for each epoch. We can then see how the network learns by visualising it

The weights learned by the network appear to function poorly for predicting the objectness score. Due to the callback using only 8 images, it's not really possible to tell if this is due to the one factor or another yet, but it doesn't correlate with the reported metrics from the training and validation sets which we can still access from the logging layer method.

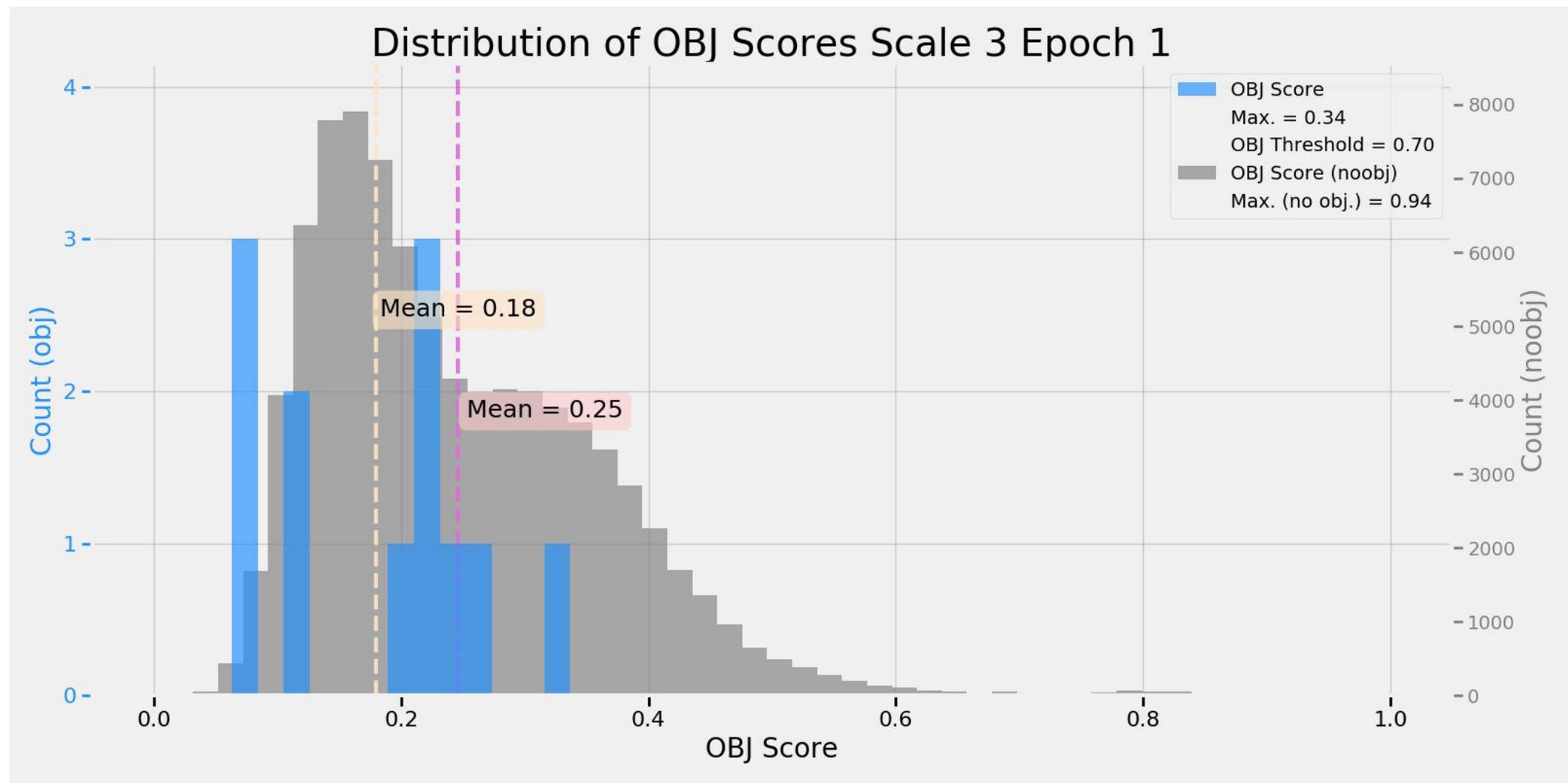
We are adding a callback which evaluates the same metrics but for the entire training/validation dataset. We are also running the metrics training on our bottles and SNNu dataset

In future, we will also be able to calculate more complex metrics combining the outputs from all 3 scales from within a callback in a similar manner.

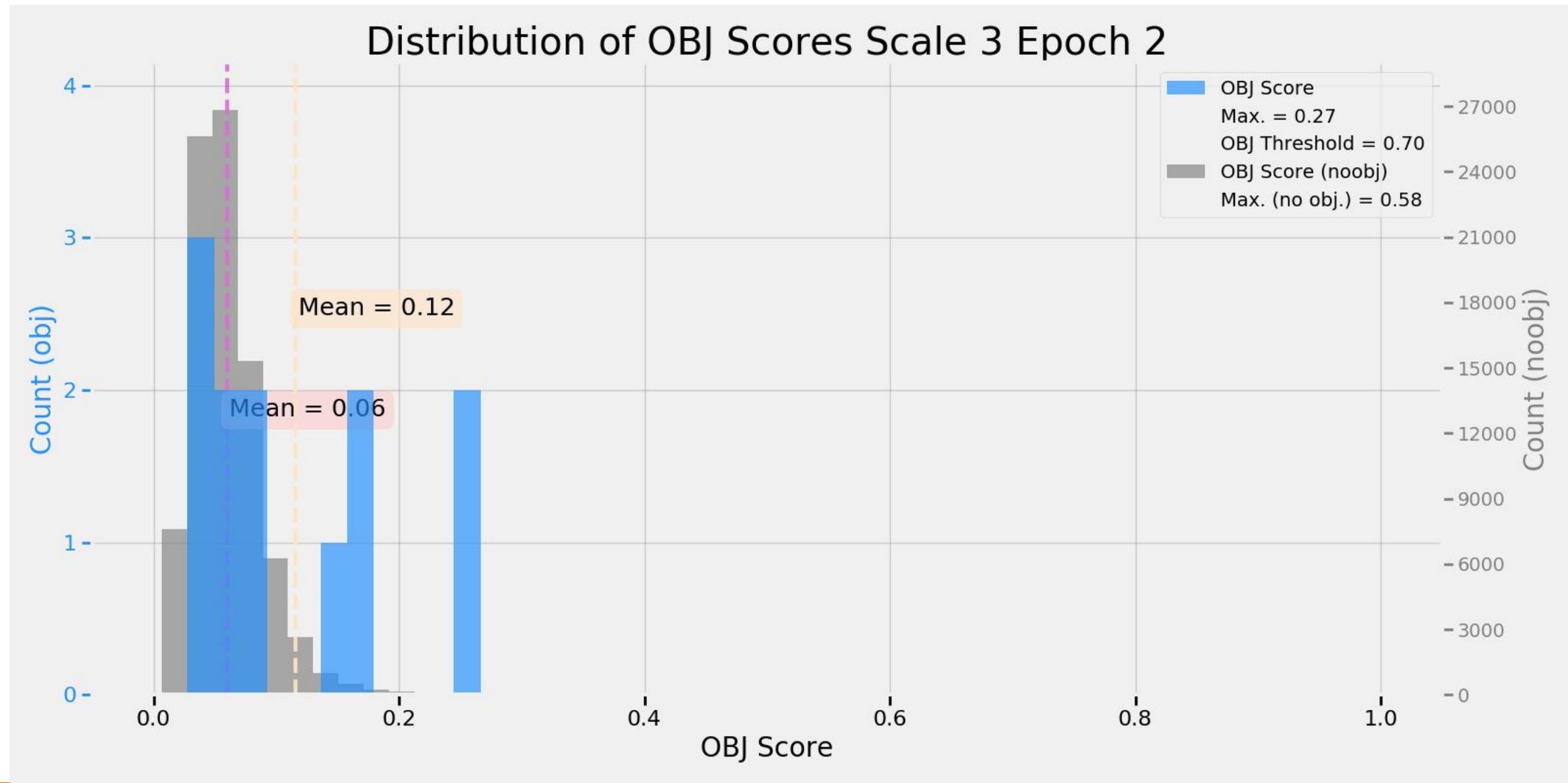
Network Training On SNNu



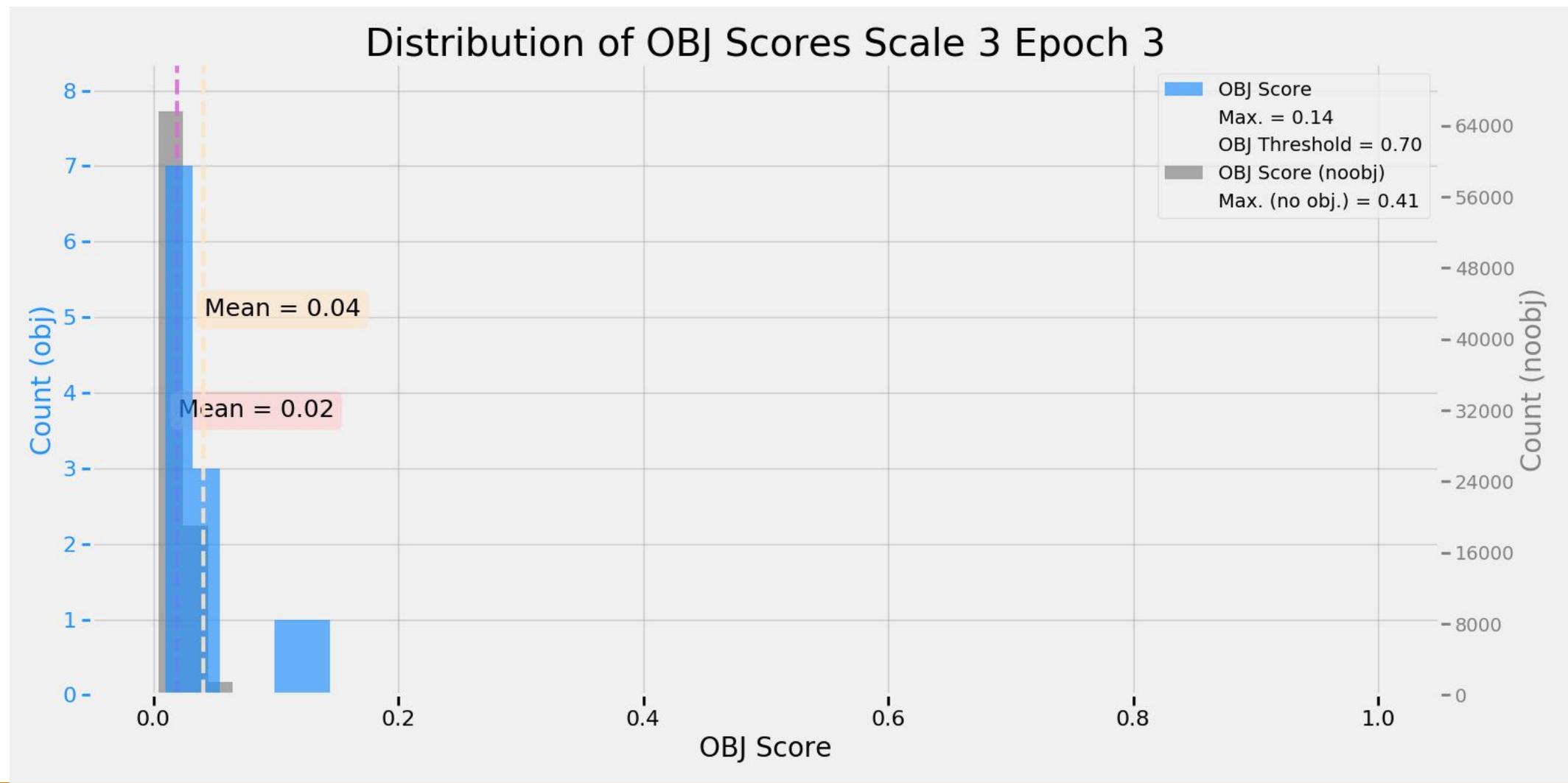
Network Training On SNNu



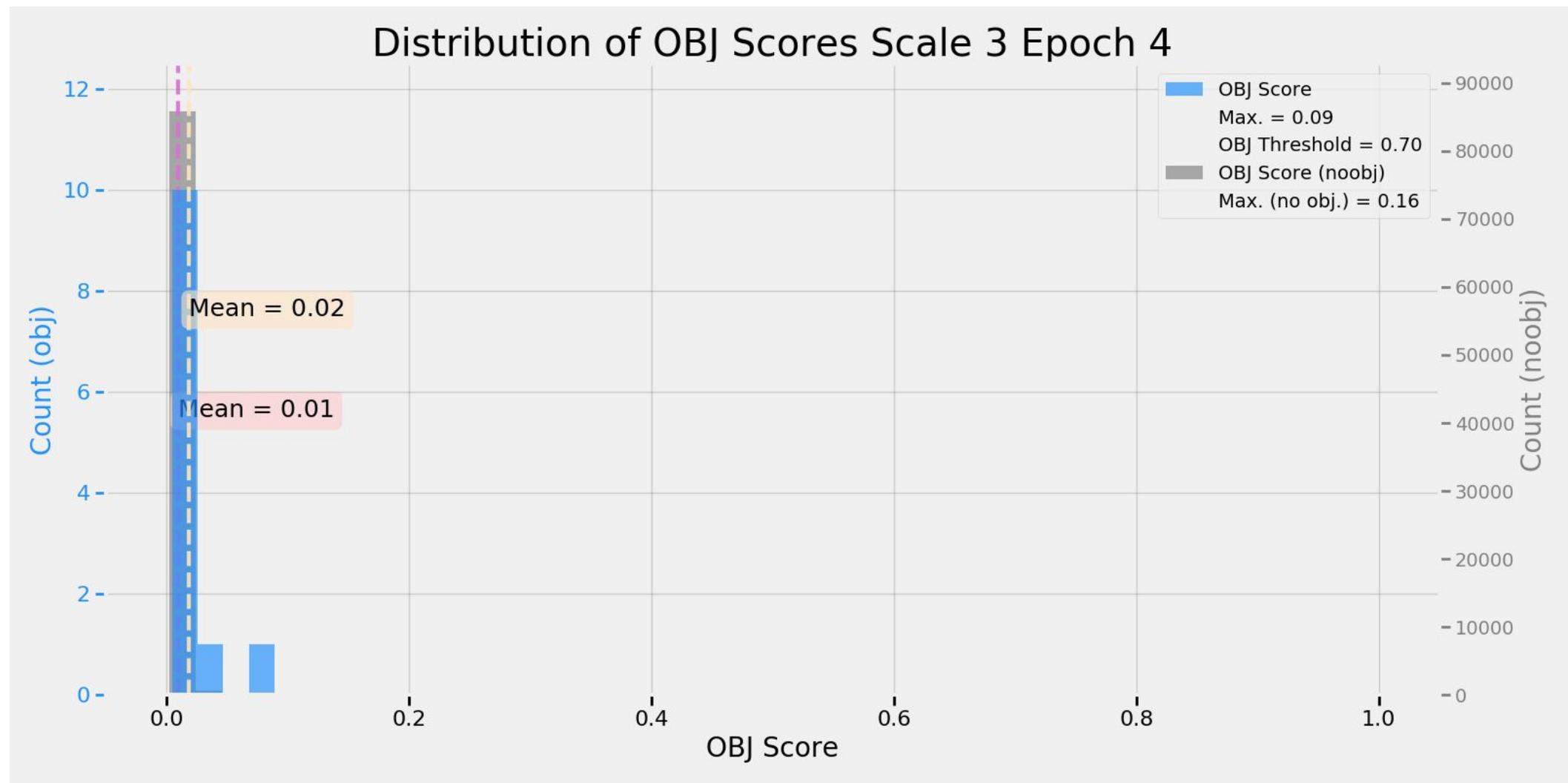
Network Training On SNNu



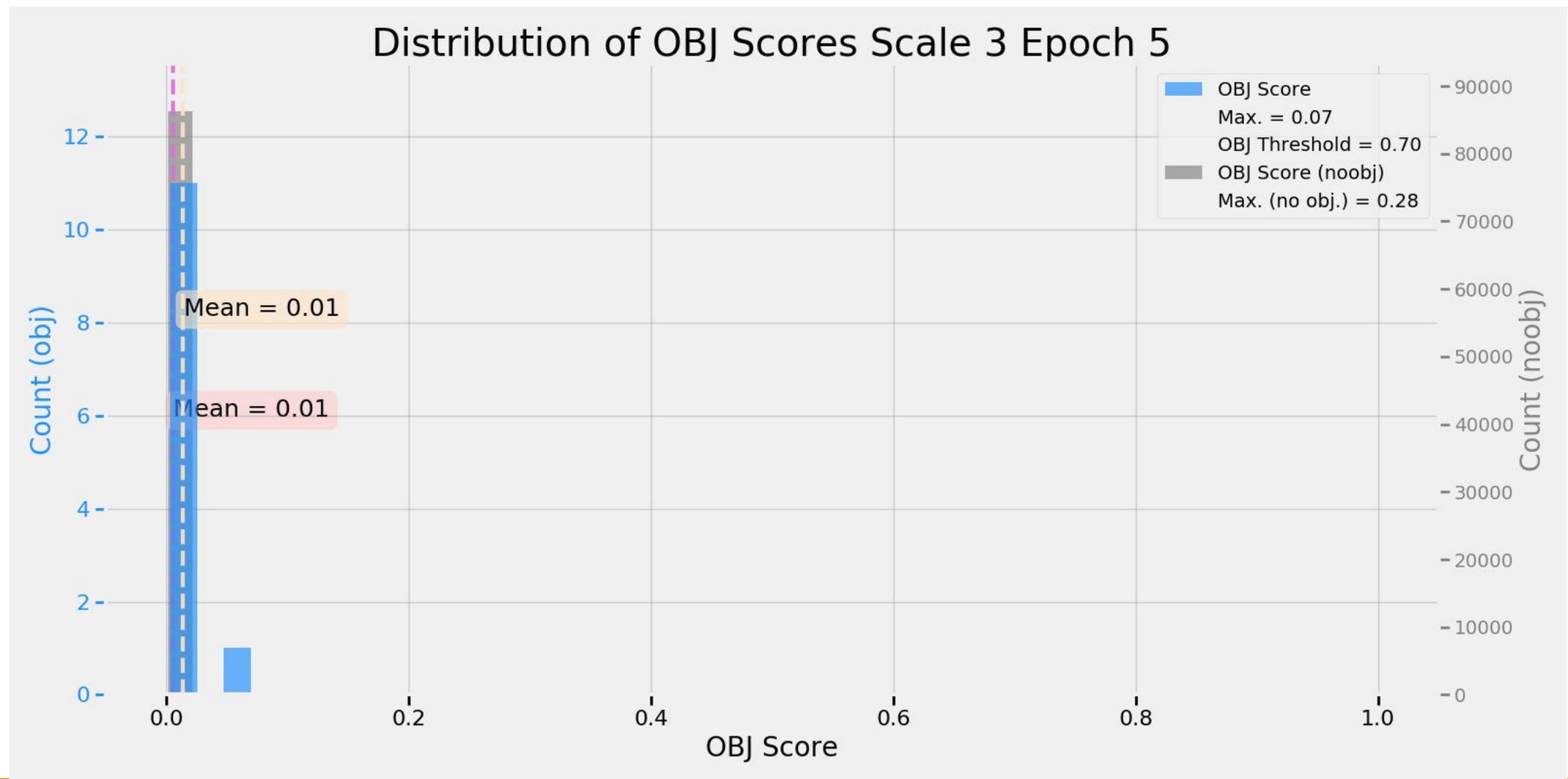
Network Training On SNNu



Network Training On SNNu



Network Training On SNNu



Network Training On SNNu

