Assignment 1: Weakly Supervised Object

Localization

Mangyin Mo (Matthew)

Oct 3rd, 2022 at 11:59pm EST



# 1 Task 0: Visualization and Understanding the Data Structures

**1.0.1 Q 0.1: What classes does the image at index 2020 contain (index 2020 is the 2021-th image due to 0-based numbering)? (3 pt)**

[18]. ”Train”

## 1.0.2 Q 0.2 Use Wandb to visualize the ground-truth bounding box and the class for the image at index 2020.(4 pt)

A picture containing text, outdoor, sky, grass

Description automatically generated

## 1.0.3 Q 0.3 Use Wandb to visualize the top ten bounding box proposals for the image at index 2020. (3 pt)

A picture containing text, grass, outdoor, sign

Description automatically generated

# Task 1: Is Object Localization Free? (50 points)

## Q 1.1 Fill in each of the TODO parts in ‘AlexNet.py‘. Next, fill the TODO parts in

**‘task˙1.py‘except the functions “metric1“, “metric2“ and “LocalizerAlexNetRobust“. You may need to refer to [1] for their choice of loss function and optimizer. As you may observe, the output of the above model has some spatial resolution. Make sure you read paper [1]**

**and understand how to go from the output to an image-level prediction (max-pool). (Hint:**

**This part will be implemented in “train()“ and “validate()“. For each of the TODO, describe the functionality of that part using appropriate comments. (2 pt)**

[Done]

## Q 1.2 What is the shape (NxCxHxW) of the output of the model? (3 pt)

torch.Size([32, 20, 29, 29])

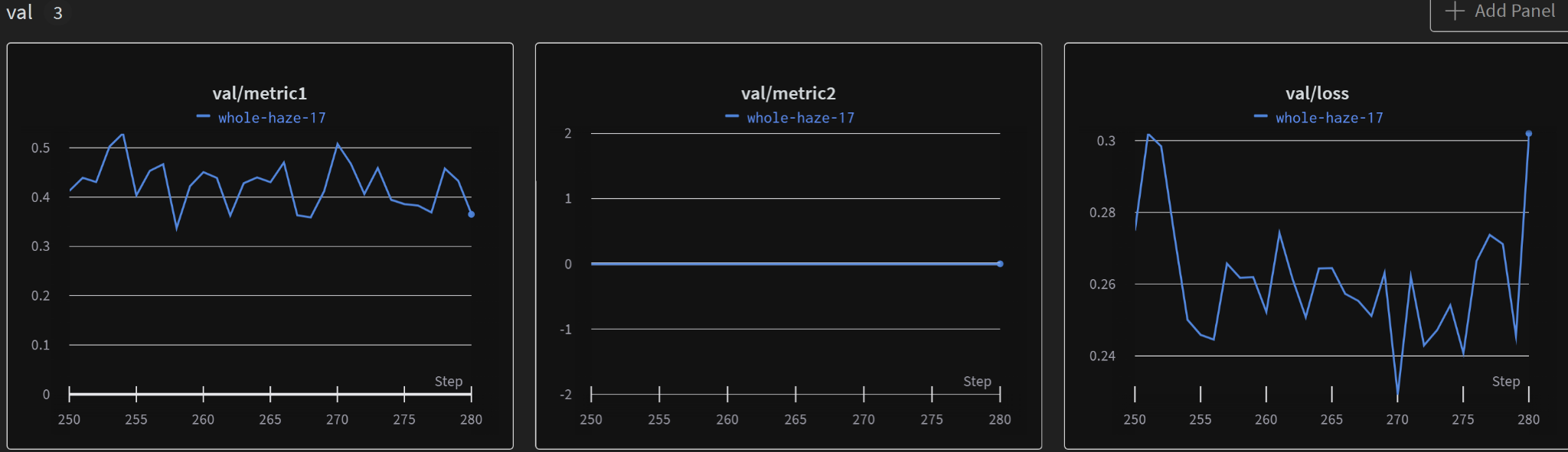
size = (32, 20, 29, 29)

## Q 1.3 Initialize the model from ImageNet (till the conv5 layer). Initialize the rest of layers with Xavier initialization and train the model using batchsize=32, learning rate=0.01, epochs=2 (Yes, only 2 epochs for now). (Hint: also try lr=0.1 - best value varies with implementation of loss) (10 pt)

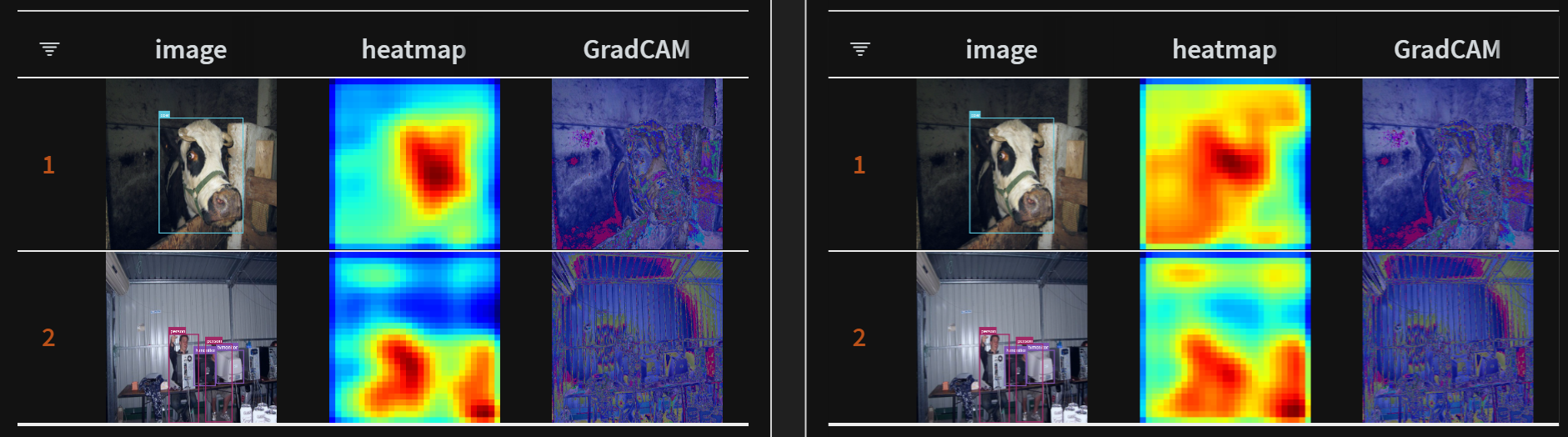
Training loss

## 

## Validation



1st and 2nd Epoch Heatmaps



## Q 1.4 In the first few iterations, you should observe a steep drop in the loss value. Why does this happen? (Hint: Think about the labels associated with each image). (2 pt)

A steep drop is usually observed at the start of the training due to high difference in the forwarded model classification output from randomly initiated weights, thus difference between random outputs and the groundtruth label (binary for each class) lead to higher gradients.

**Q 1.5 We will log two metrics during training to see if our model is improving progressively with iterations. The first metric is mAP, a standard metric for multi-label classification.**

## Write the code for this metric in the TODO block for “metric1“ (make sure you handle all the boundary cases). However, “metric1“ is to some extent not robust to the issue we identified in Q1.4. The second metric, Recall, is more tuned to this dataset. Even though there is a steep drop in loss in the first few iterations “metric2“ should remain almost constant. Implement it in the TODO block for “metric2“. (Make any assumptions needed like thresholds). Feel free to use libraries like “sklearn“. (3 pt)

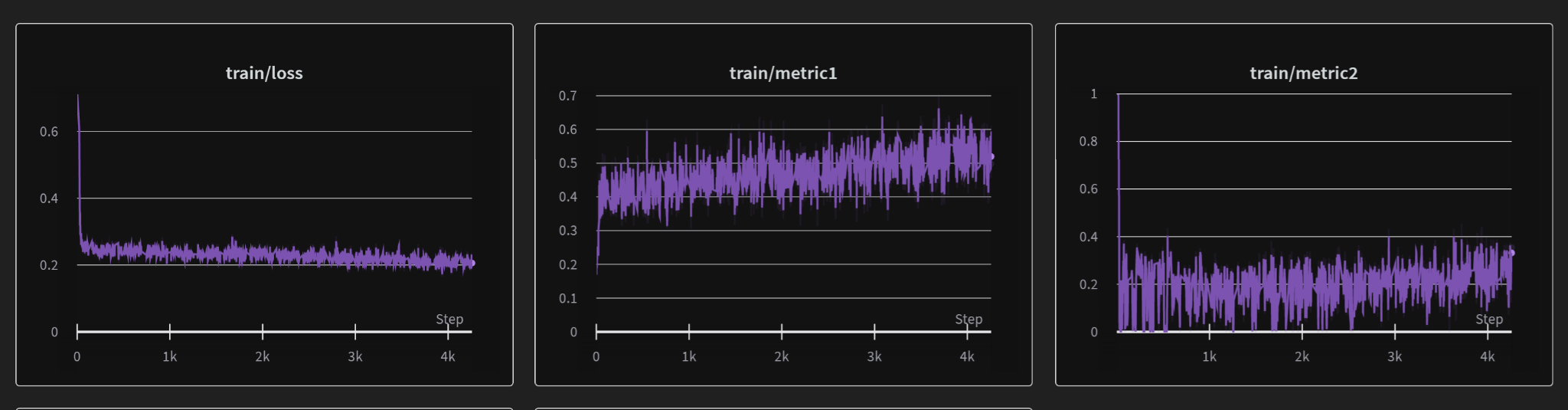
Assumption we make is that a threshold with greater than 0.5 is considered a ’true’ classification for the binary classification of any individual class. Also, it is important to note that within the batch, not all classes are present, therefore, calculating precision among all the classes will result in a very low score. In this implementation, only the classes that appeared in the batch are used, so we need to assume that there is relatively little class imbalance and the metric itself is potentially inflated but offer a good approximation of the improvement. Doing so also solved the boundary case where some classes did not have any classification in the batch at all,

[Code Done]

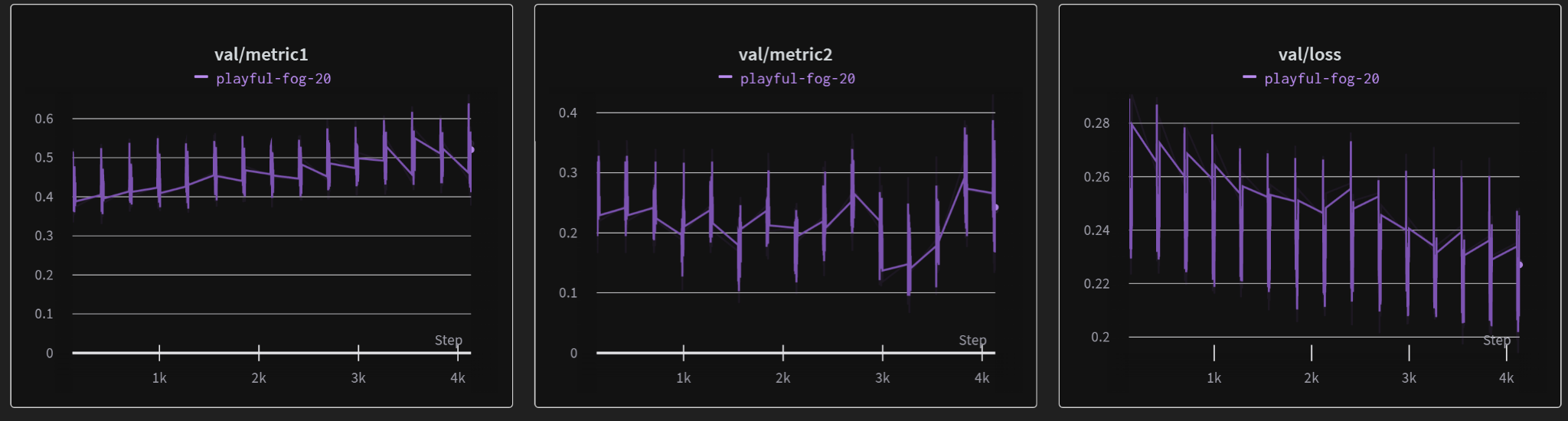
## Q 1.6 Initialize the model from ImageNet (till the conv5 layer), initialize the rest of layers with Xavier initialization and train the model using batchsize=32, learning rate=0.01, epochs=30. Evaluate every 2 epochs. (Hint: also try lr=0.1 - best value varies with implementation of loss) [Expected training time: 45mins-75mins]. (15 pt)

(The following is implemented with GradCAM for bonus (in all task 1))

Training plots

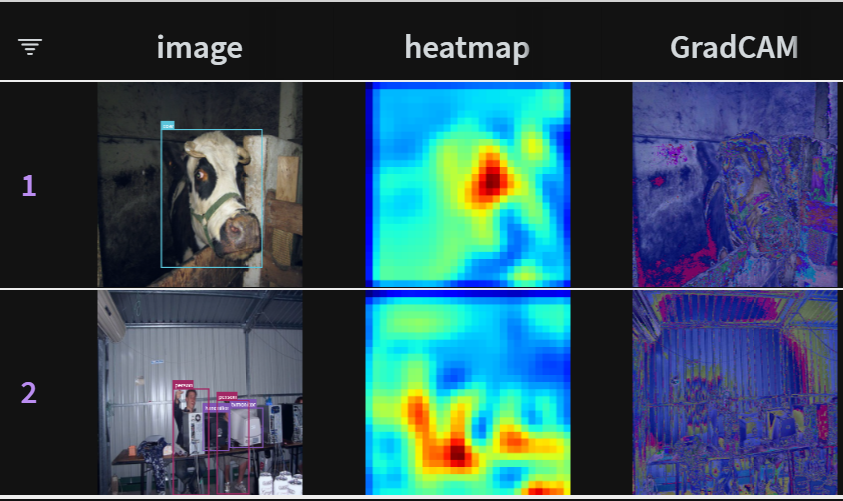


Validation plots

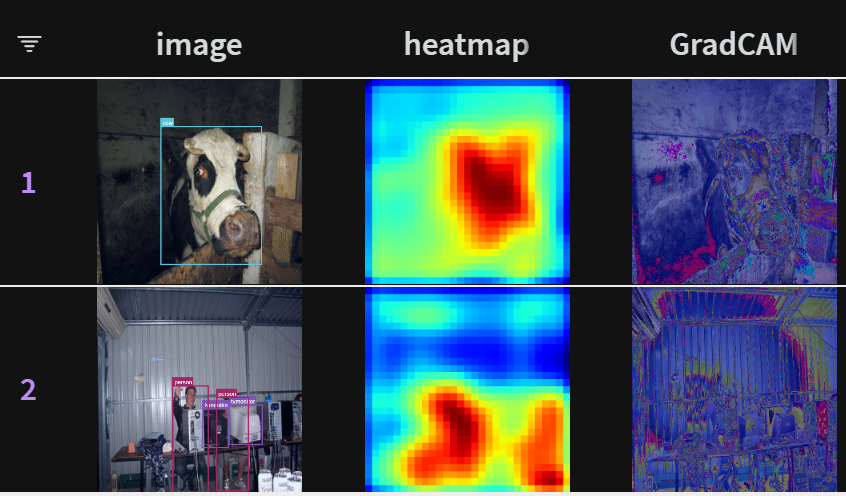


Heatmaps

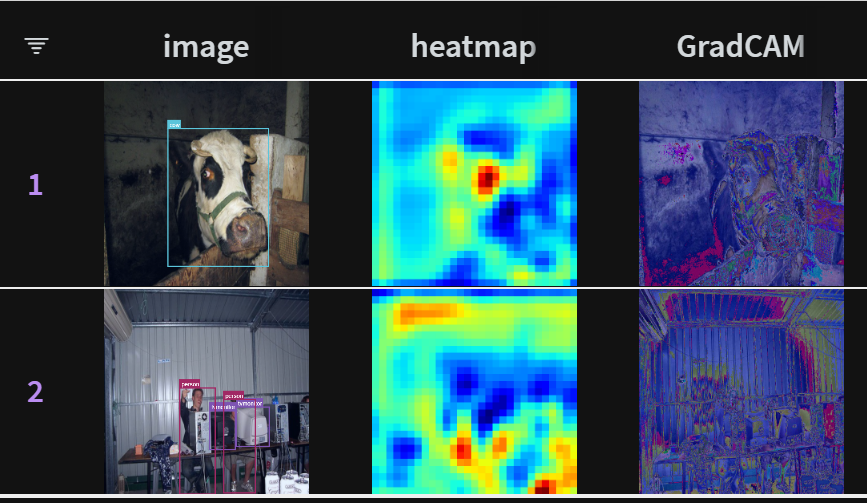
Epoch 0



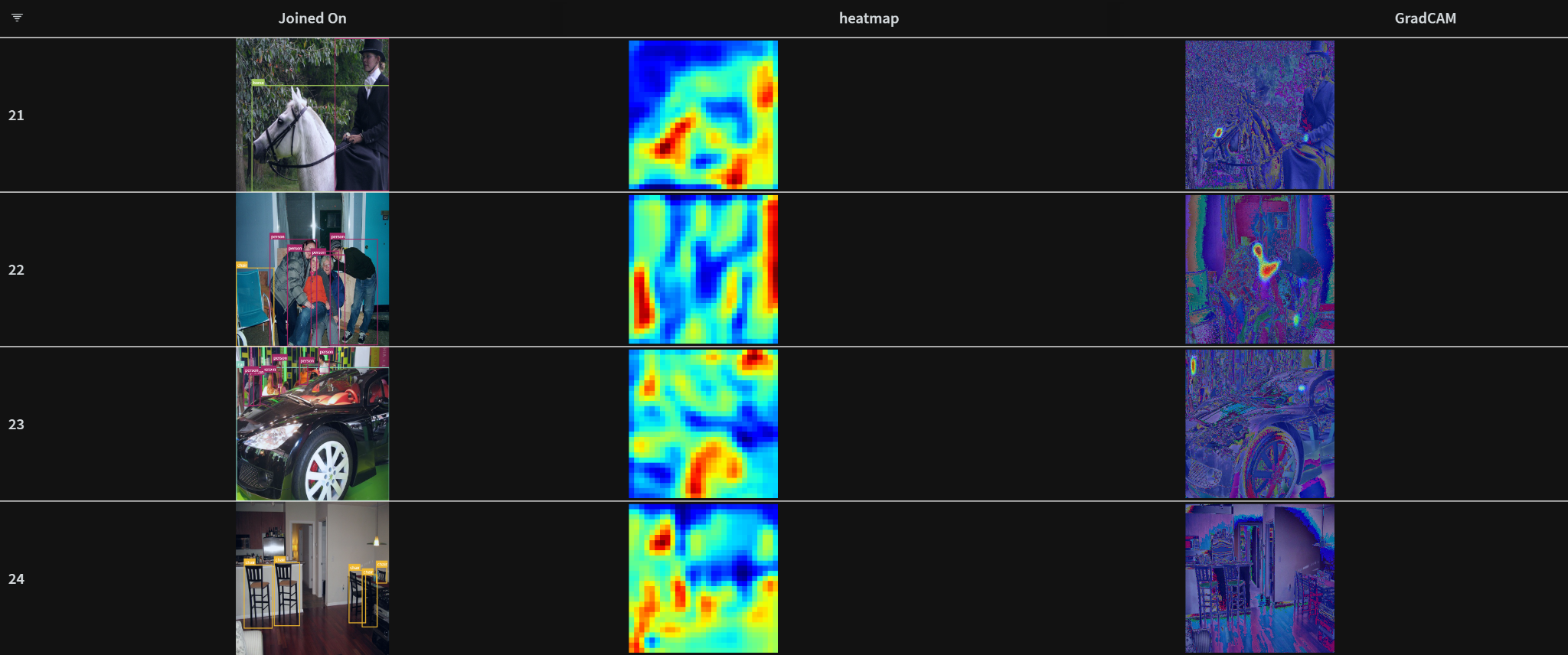
Epoch 15



Epoch 30



3 random validation



|  | Loss | Metric 1 | Metric 2 |
| --- | --- | --- | --- |
| Training | 0.21 | 0.48 | 0.25 |
| Validation | 0.23 | 0.45 | 0.2 |

[Code Done]

**Q 1.7 In the heatmap visualizations you observe that there are usually peaks on salient features of the objects but not on the entire objects. How can you fix this in the architecture of the model?**

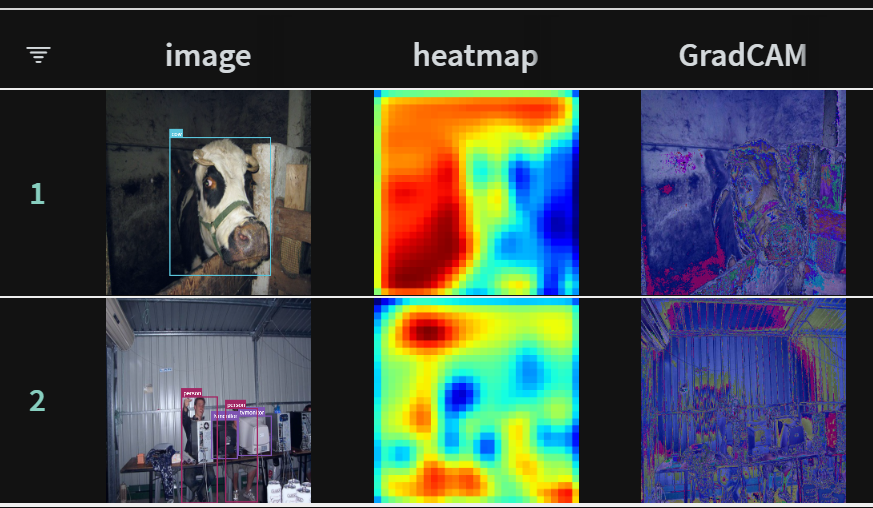
Peaks on salient features may be due to over-fitting as the model picks up on ”shortcuts” that are not relevant to generalization of the task. We can generally solve unwanted artifacts such as that by implementing regularization. We can add dropout layers, or perform some ’post-processing’ smoothing by average pooling or filtering. Here, several dropout layers with dropout=[0.3, 0.2,0.2] are added to the output of the last conv layer from model.features and then to the first 2 layers of the model.classification part. We see some improvement in this version of implementation.

Heatmaps

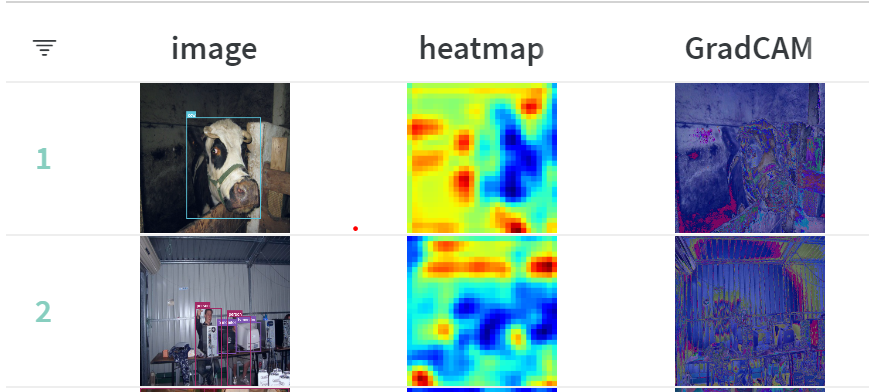
Epoch 0



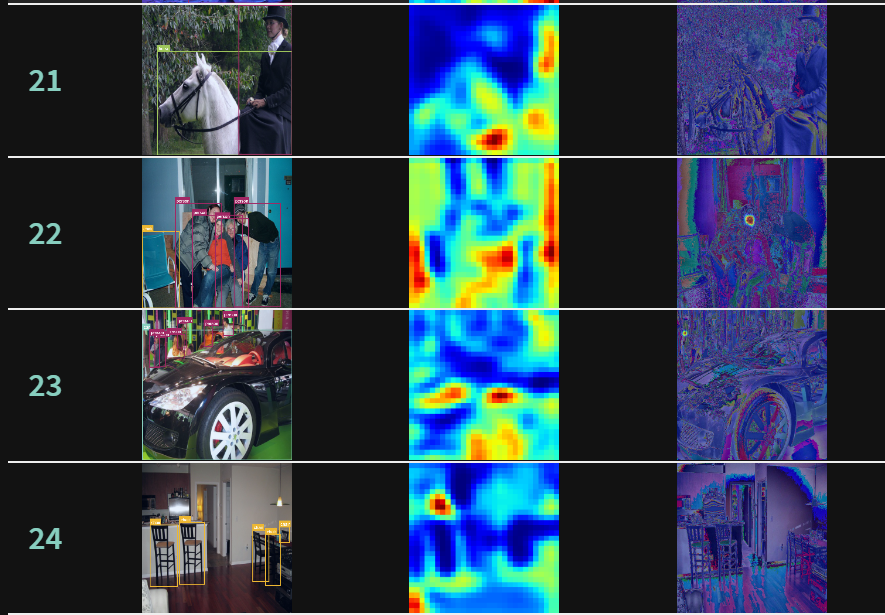
Epoch 15



Epoch 30



3 Random validation



|  | Loss | Metric 1 | Metric 2 |
| --- | --- | --- | --- |
| Training | 0.2 | 0.50 | 0.23 |
| Validation | 0.25 | 0.53 | 0.14 |

[Code Done]

# Task 2: Weakly Supervised Deep Detection Networks (40 points)

**Q2.1 In wsddn.py, you need to complete the ˙˙init˙˙, forward and build˙loss functions.**

The ˙˙init˙˙() function will be used to define the model. You can define 3 parts for the model

The feature extractor A ROI-pool layer (use torchvision.ops) A classifier layer, as defined in the WSDDN paper. The forward() function will essentially do the following:

**Q2.2 In task˙2.py you will first need to write the training loop.**

[Code Done]

**2.3 In task˙2.py, you now need to write a function to test your model, and visualize its predictions.**

Class-wise AP is computed by first iterating each class, then by filtering both ground-truth and predicted bounding boxes by their respective class (for prediction, take argmax of the probability as the assigned class (Hard assignment)), then we will filter the rois with Non-maximum suppression resulted in predicted boxes.

Now with predicted boxes and ground-truth boxes, we will iterate each ground-truth box to predicted boxes, not the other way around such that missing predicted boxes will be penalized. We will take the maximum IOU between the true box and predicted box for each pair of boxes then remove them iteratively.

Doing so for each class we average out the precision across the class to obtain the final mean AP per class of length [20].

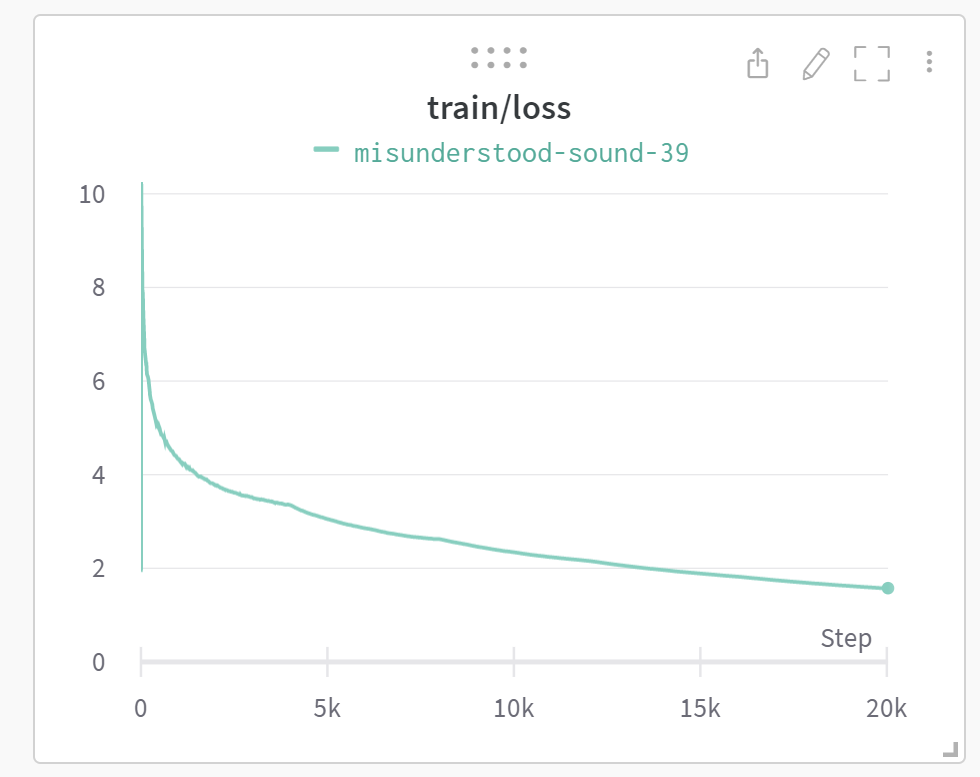
**Q2.4 In task˙2.py, there are places for you perform visualization (search for TODO). You need to perform the appropriate visualizations mentioned here:**

It is to be noted here that after a lot of iterations of trials, both SGD and Adam optimization may have little effect on training the model and could only bring loss down slightly and the model remains sub-performing. More time and optimization trials /experimentation are required to train the model optimally.

[Code Done]

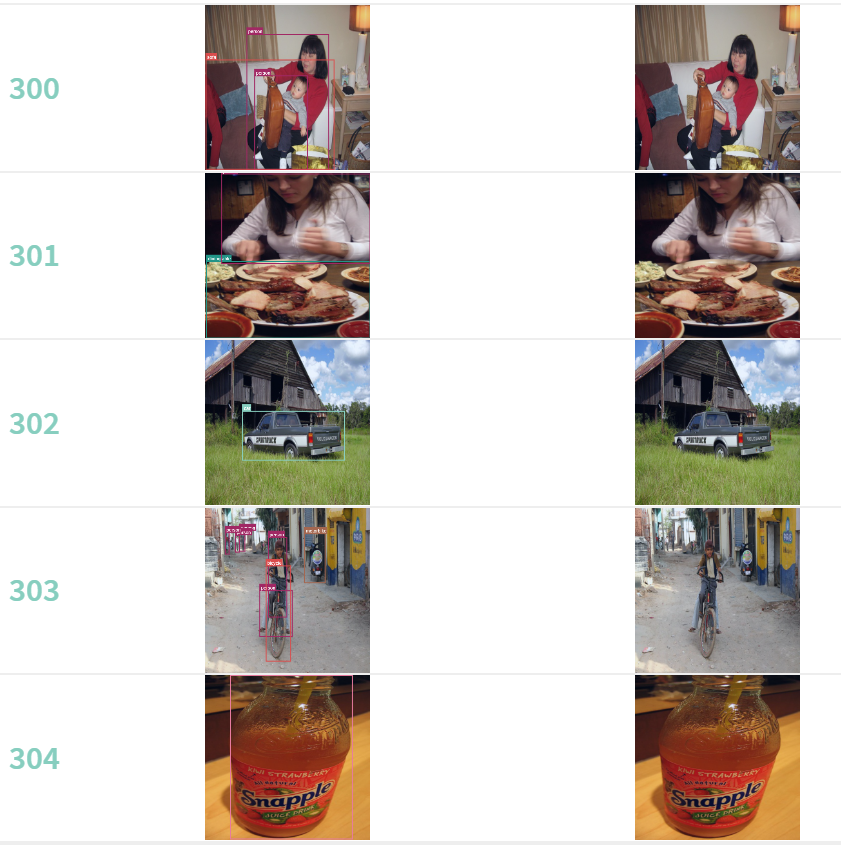
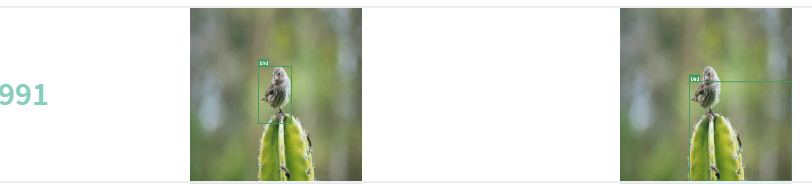
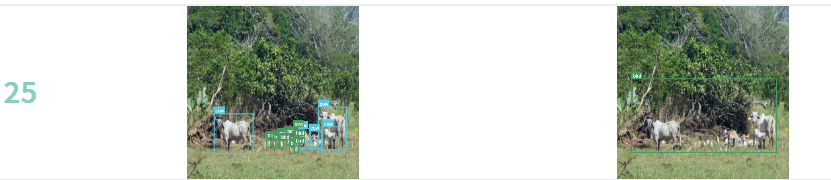
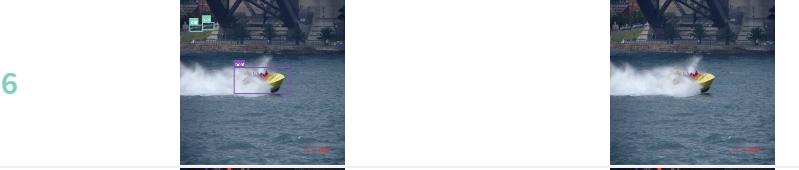
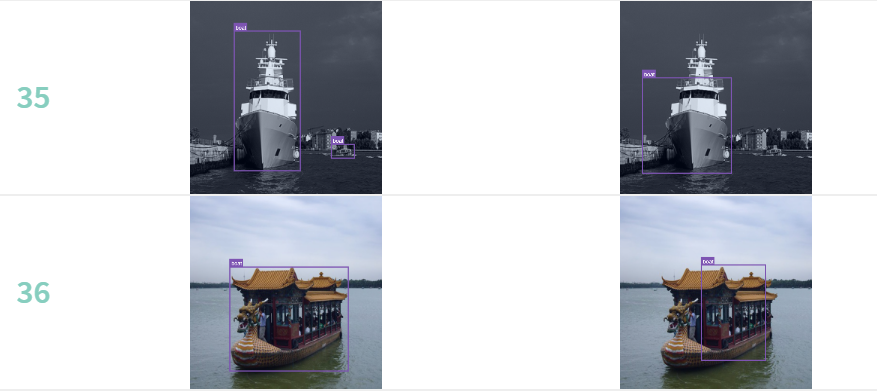
**Q2.5 Train the model using the hyperparameters provided for 5-6 epochs.**

Training loss

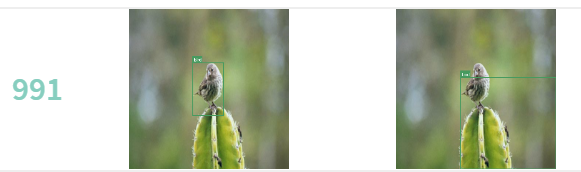
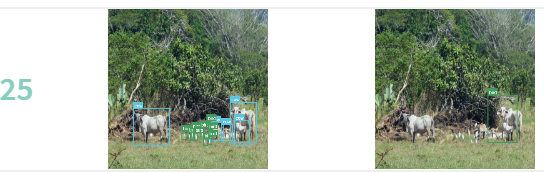
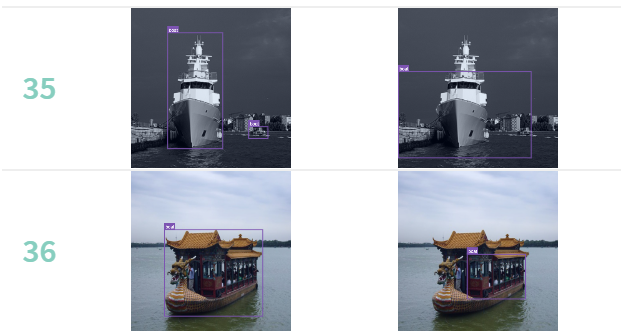


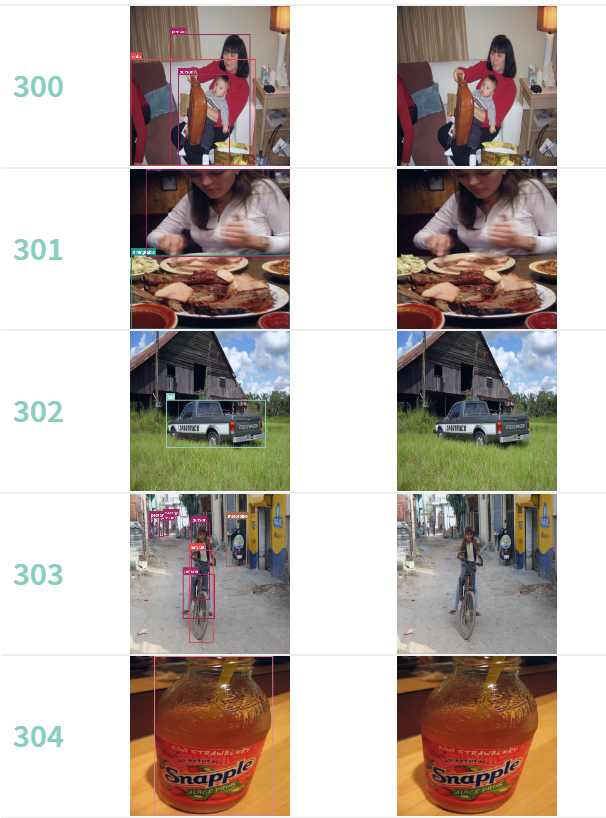
10 images Visualizations

Epoch 0



Epoch 5





[Code Done]

# Task 3: Visualizing Class Activation Maps (Extra Credit - 10 points)

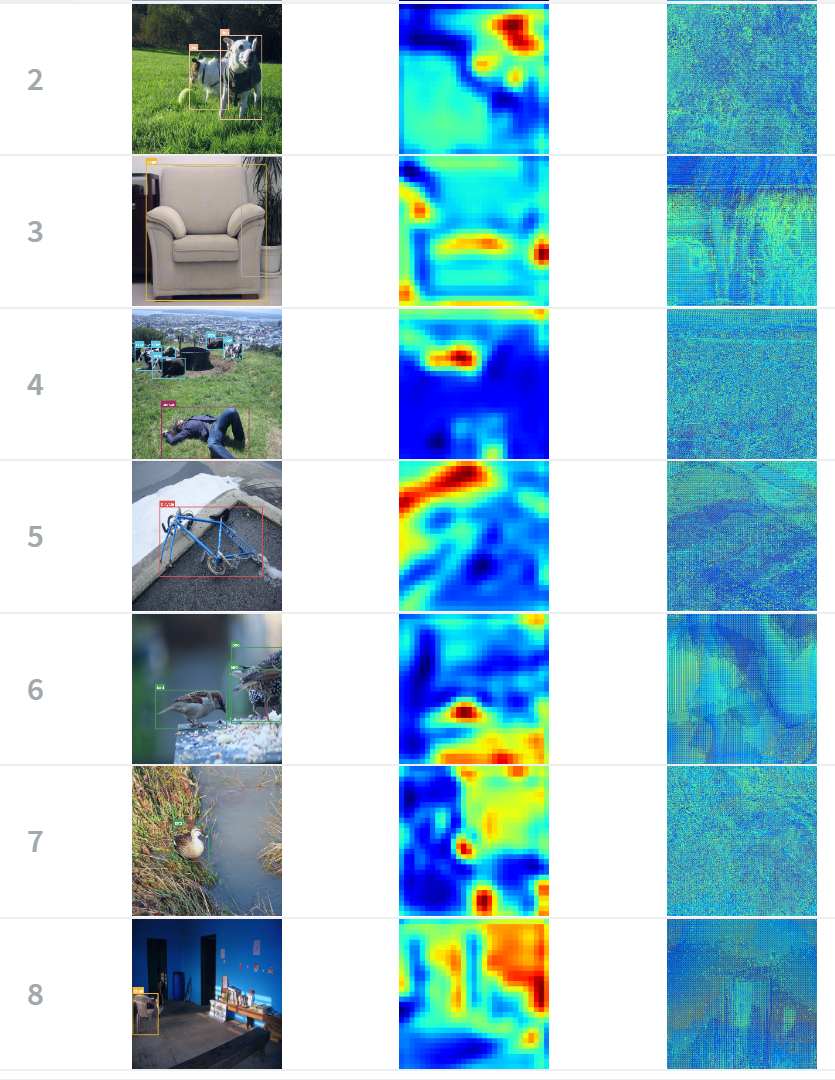
Another popular way to visualize how your network is learning is Class activation maps (as explained in the lecture)! In this task, we want you to experiment with state-of-the-art CAM methods (feel free to use existing

implementations such as https://github.com/frgfm/torch-cam). Using a network you trained (in either task 1 or task 2), apply one (or more) CAM methods to visualize how well your network is doing. In the report, explain the implementation of the method you chose in 2-3 sentences. Additionally, add 2-3 visualizations in the report and try comparing and contrasting with the inferences you made using heatmaps.

**Q3**.

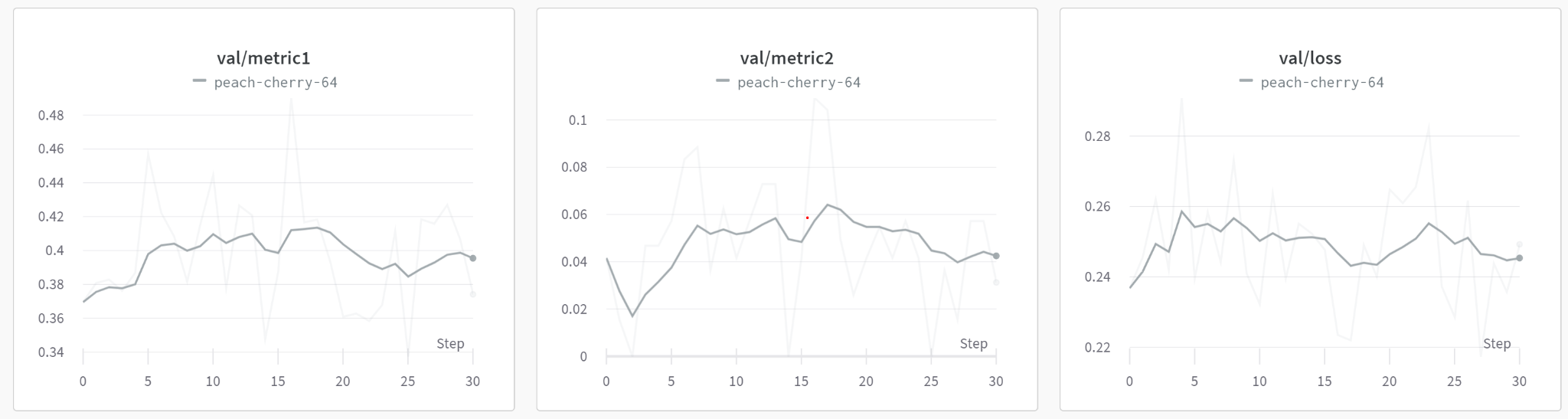
GradCAM provides a way for us to visualize the activation maps in convolution , specifically, it allows us to ‘dissect’ the activation map by visualizing in a ‘class-specific’ way. In detail, gradients from a specific target are used to pass into the final convolutional layer to produce a “coarse localization map highlighting important regions in the image for predicting the concept.” It is built on a technique called Class Activation Mapping (CAM) for identifying discriminative regions used by a restricted class of image classification CNNs which do not contain any fully-connected layers.

For convenience, GradCam Activation map overlays have been added to task1 in the visualization loop, where the gradient weighted map is overlaid on the original image. In addition , the following several images attached from using the gradient weighted activation map in the forward propagation loop :



Upon using GradCAM to weight the activation map before making the inference, it is noticed that the attention heatmap for the particular object becomes more succinct and helps the classification inference perform better for the respective class but not for all the classes.

The evaluation of gradient-weighted activation maps but only using the weight for one of the classes for all predictions.



# References

# Submission Checklist

In all the following tasks, coding and analysis, please write a short summary of what you tried, what worked (or didn't), and what you learned, in the report. Write the code into the files as specified. Submit a zip file (ANDREWID.zip) with all the code files, and a single REPORT.pdf, which should have commands that TAs can run to re-produce your results/visualizations etc. Also mention any collaborators or other sources used for different parts of the assignment.

## Report

### ~~Task 0~~

* ~~Answer Q0.1~~
* ~~wandb screenshot for Q0.2~~
* ~~wandb screenshot for Q0.3~~

### ~~Task 1~~

* ~~Q1.1 describe functionality of the completed TODO blocks with comments~~
* ~~Answer Q1.2~~
* ~~Q1.3~~
  + ~~Add screenshot of training loss~~
  + ~~Screenshot of wandb showing images and heat maps for the first logged epoch~~
  + ~~Screenshot of wandb showing images and heat maps for the second logged epoch~~
* ~~Answer Q1.4~~
* ~~Answer Q1.5 and mention the assumptions~~
* Q1.6
  + ~~Add screenshot of metric1, metric2 on the training set~~
  + ~~Add screenshot of metric1, metric2 on the validation set~~
  + ~~Screenshot of wandb showing images and heat maps for the first logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~Screenshot of wandb showing images and heat maps for the 15th logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~Screenshot of wandb showing images and heat maps for the last logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~wandb screenshot for 3 randomly chosen validation images and heat maps \*show image and heatmap side-by-side\*.~~
  + ~~Report training loss, validation metric1, validation metric2 at the end of training~~
* ~~Q1.7~~
  + ~~Screenshot of wandb showing images and heat maps for the first logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~Screenshot of wandb showing images and heat maps for the 15th logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~Screenshot of wandb showing images and heat maps for the 30th logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~Screenshot of wandb showing images and heat maps for the last logged epoch \*show image and heatmap side-by-side\*.~~
  + ~~wandb screenshot for 3 randomly chosen validation images (but same images as Q1.6) and heat maps \*show image and heatmap side-by-side\*.~~
  + ~~Report training loss, validation metric1, validation metric2 at the end of training~~

### Task 2

* ~~Q2.3 detailed code comments on how classwise AP and mAP are calculated~~
* ~~Q2.4 wandb downloaded image of training loss vs iterations~~
* ~~Q2.4 wandb downloaded image of test mAP vs iterations plot~~
* Q2.4 screenshot for class-wise APs vs iterations for 5 classes
* ~~Q2.4 screenshot of 10 images with predicted boxes for the first logged epoch~~
* Q2.4 screenshot of 10 images with predicted boxes for the last logged epoch (~5 epochs)
* Q2.4 report final classwise APs on the test set and mAP on the test set

### ~~Task 3~~

* ~~Description of CAM method,~~
* ~~2-3 images to compare similarities/differences with previous tasks~~