# ELEC 475 Lab 4

William Dormer, 20176544

## Model information:

Due to time limitations after much testing, I had to resort to a model that I've trained for relatively few epochs. However, the accuracy is still quite good, so I'm satisfied.

Hardware information: training using GTX1070 GPU

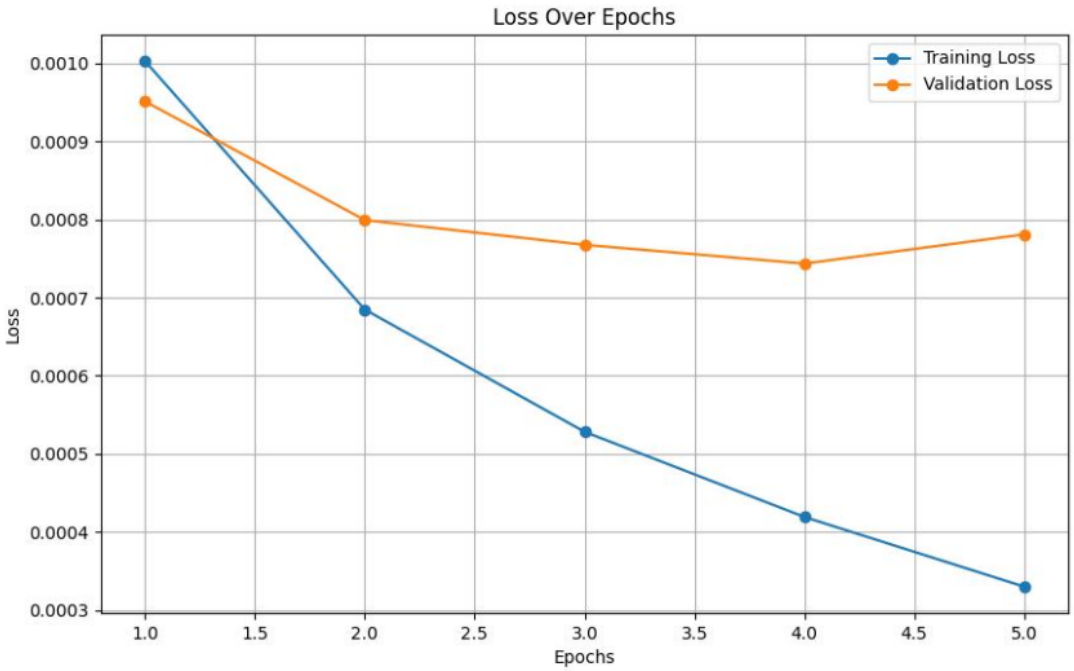
experiment name: "final_run"

optimizer: Adam
learning rate: 1e-4
weight decay: 1e-5
loss function: CrossEntropyLoss
scheduler: ExponentialLR (gamma 0.9)
batch size: 128
num epochs: 5
base model: resnet18 (with final layer modified to use 2 outputs instead of 1000 for binary classification)
weight initialization: using the default resnet18 weights for all except final layer
freezed layers: None (this model fine tuned the whole model, not just the FC layer)
transformations:

1. Resize (224 x 244)
2. Random Horizontal Flip
3. Color Jitter (brightness=0.2, contrast=0.2, saturation=0.2, hue=0.1)
4. Normalize (mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

In the lab manual it said to use the test set for validation. I was not sure if it wanted us to split the test set into validation and test, then use the validation partition for validation, and the test for testing, or to use the whole test partition to do both. I used the former because it seemed like a better approach.

## Loss plot:

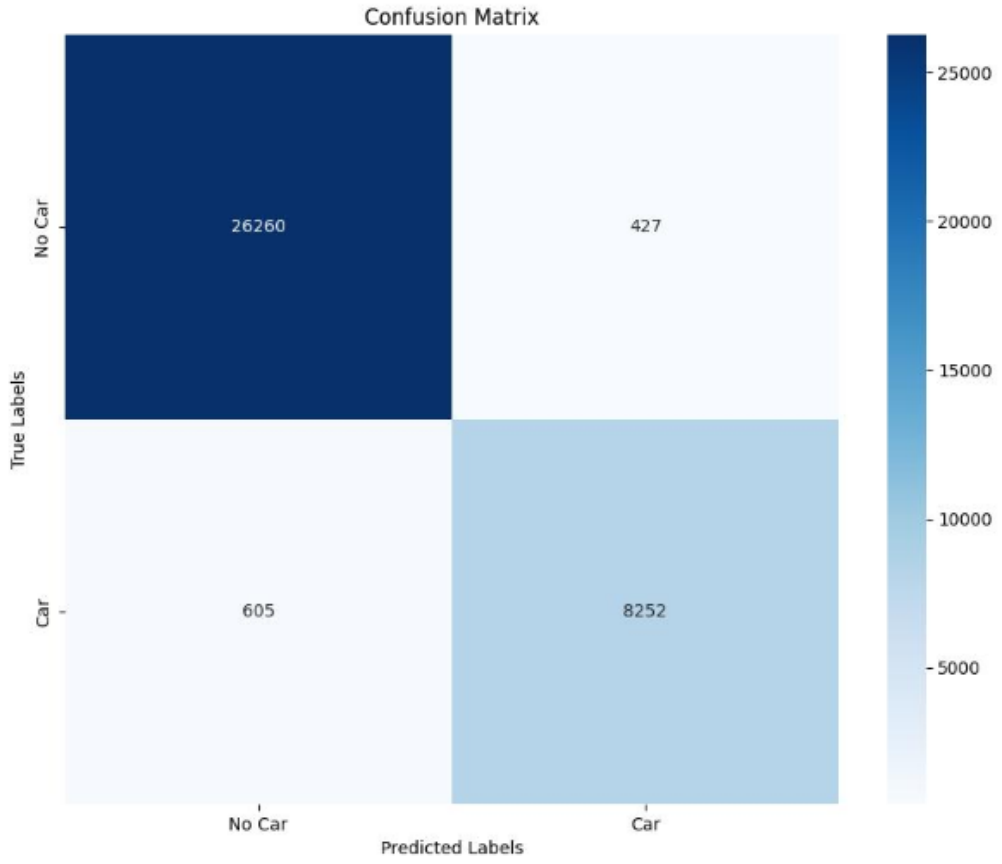



So the 4th epoch is the best model thus far.

accuracy: 97% on test, 97% on validation

confusion matrix:

---

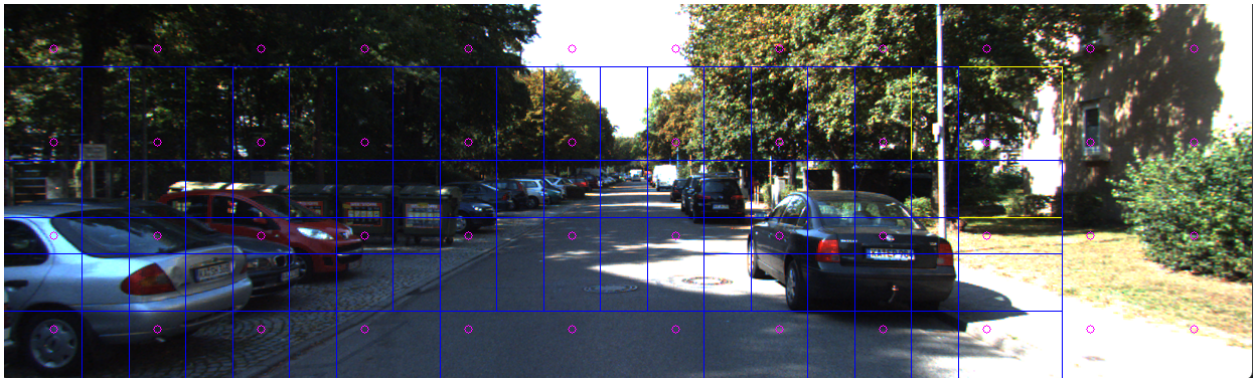



## did the method work?

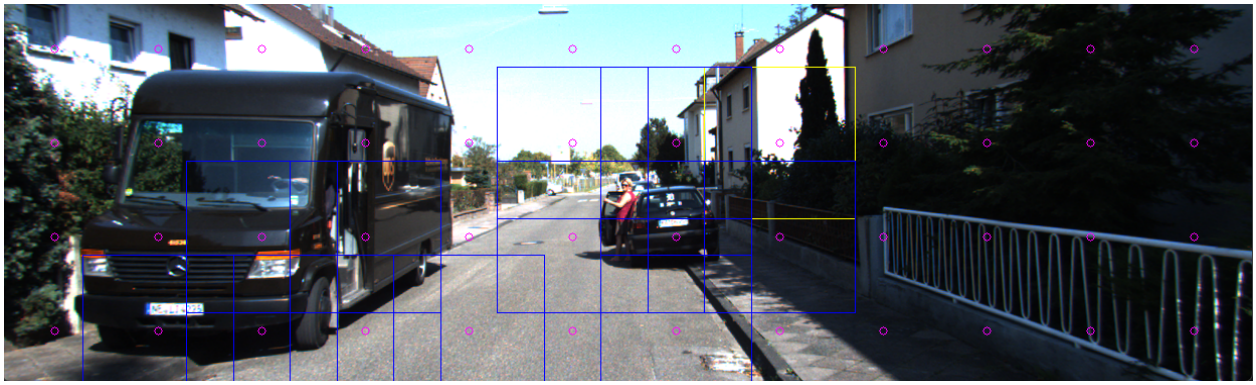
Yes. It seems that the misclassifications that the model is making are more likely issues with the ground truth labels, rather than errors in the model. I believe that any further training on this dataset would actually detract from the generalizability and utility of the model.

## kitti test images with overlaid car bounding boxes:

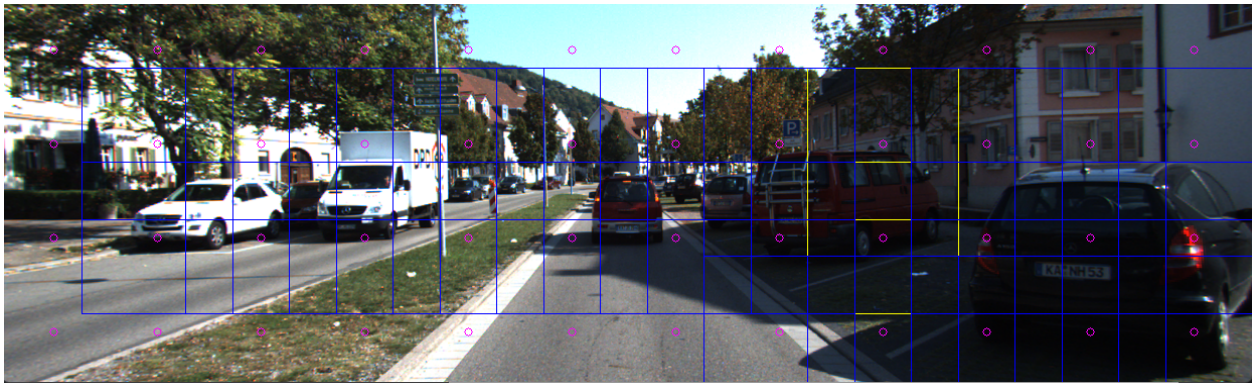
The yellow boxes are the original label boxes, and the blue boxes are the overlaid predictions of the model.



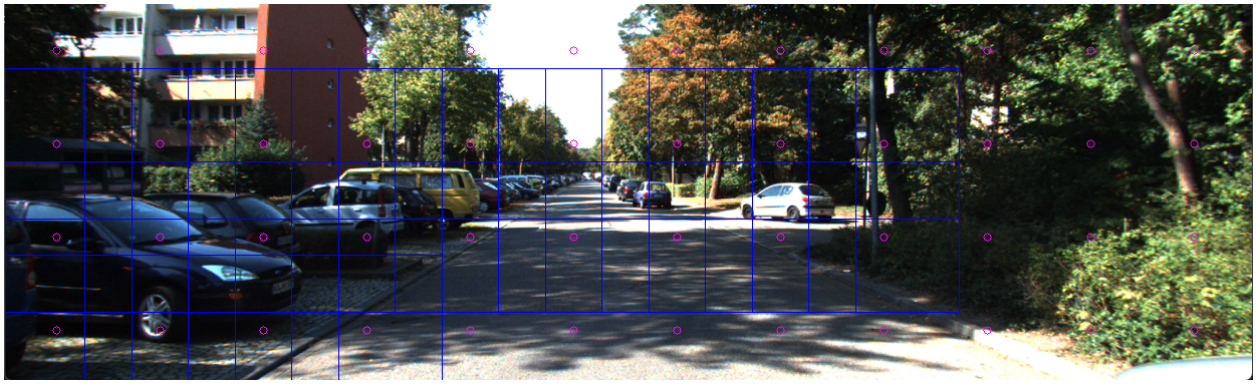
here it gets all the regions excluding one that is arguably not even including a car.



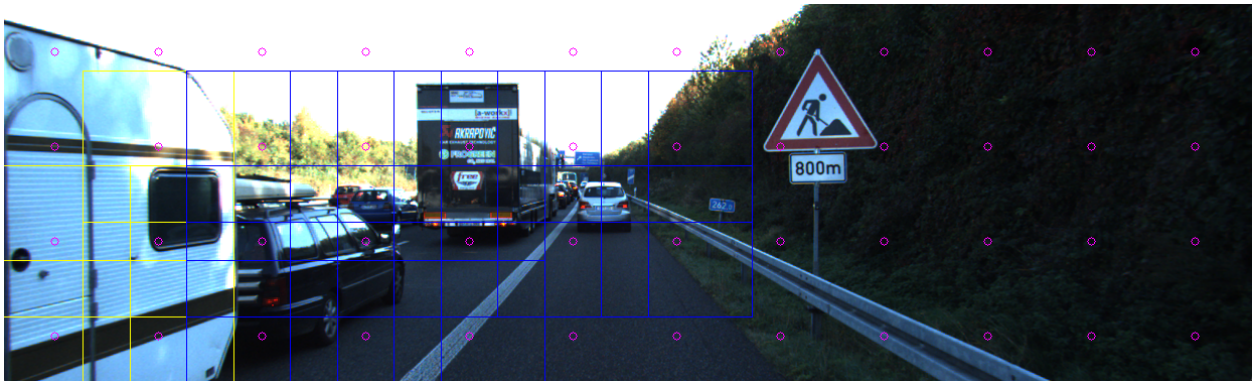
Here is picked up the car, and the van, which didn't count as a car in the original label, but has similar features in the regions that were selected. I suspect the lack of context for the regions contributes to this problem.



nearly perfect recollection of the regions excluding a single one.



perfect recognition of an image.



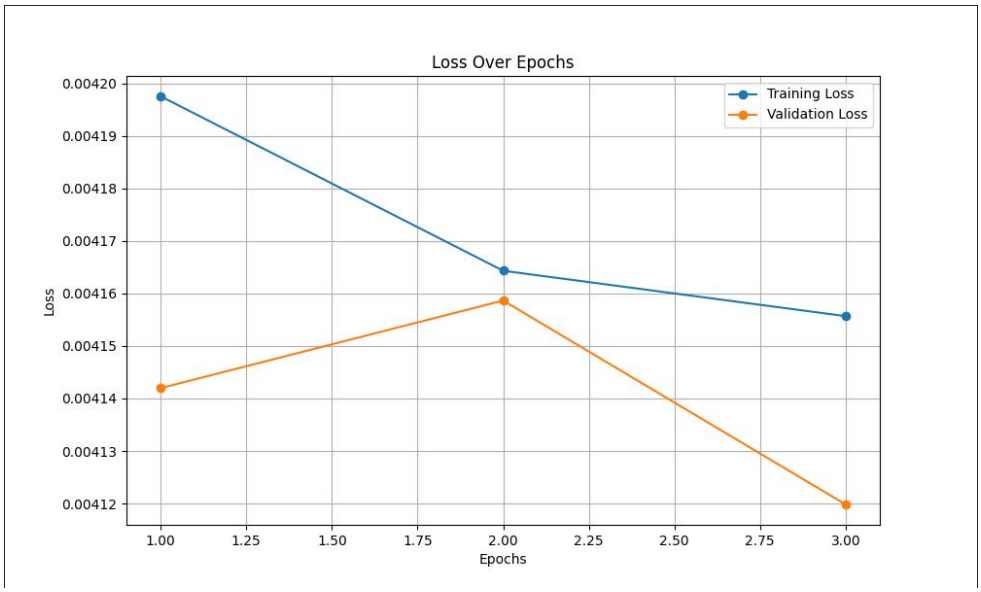

picked up the labeled regions excluding the trailer, which shouldn't have been labeled as a car.

## overall average IOU: 0.1572

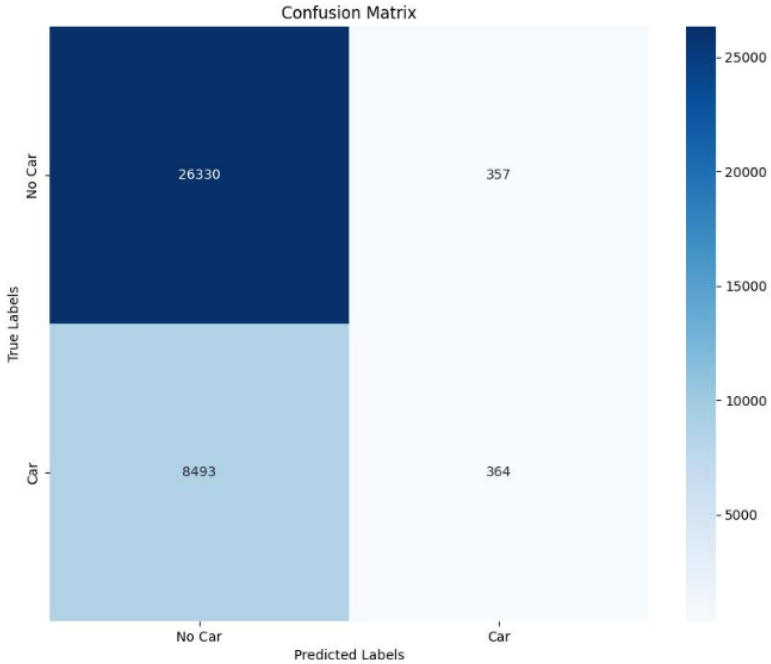
## Challenges:

I had a large number of challenges in this lab. It seemed that no matter what I tried my model either overfit, or would not learn effectively. I consulted with other students that had good performance, and they said they were taking a similar strategy to mine. However, I did eventually resolve the problem and get a reasonable model.

Here are some of my poor intermediate results

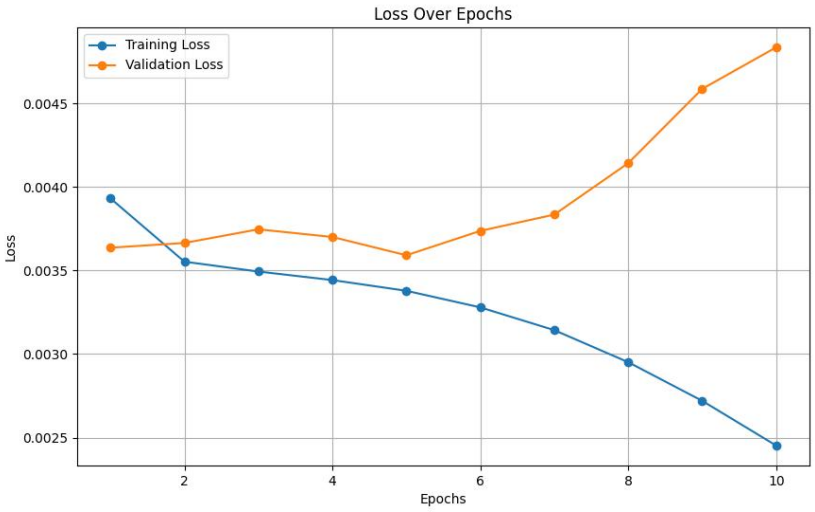


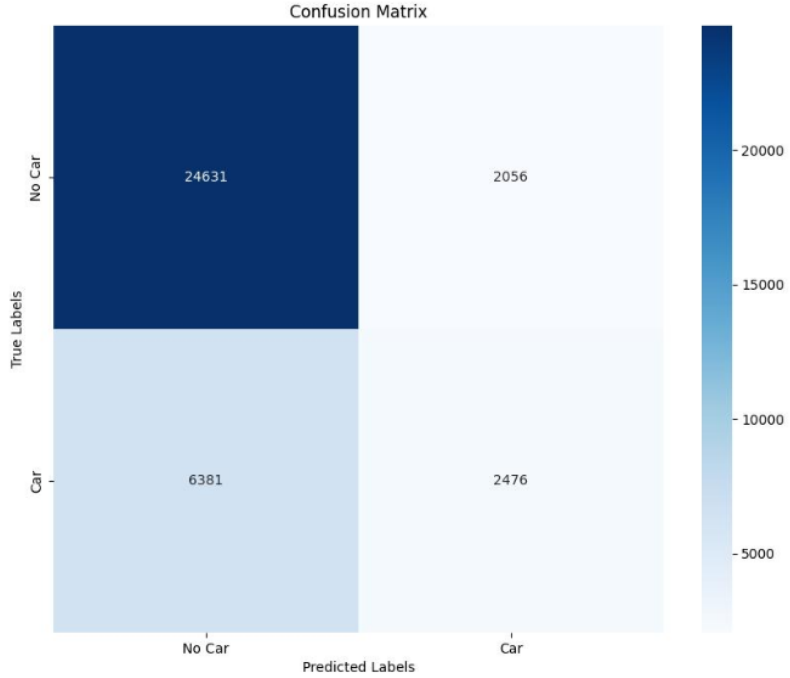

validation loss may have improved marginally, but I haven't run enough epochs on this particular model to know for certain if more epochs are going to improve performance. It still seems to have plateaued at around 75% accuracy, similar to my other model attempts.




As you can see it's clearly guessing most as simply not a car, and for the ones it's predicting to be a car it is barely better than random.

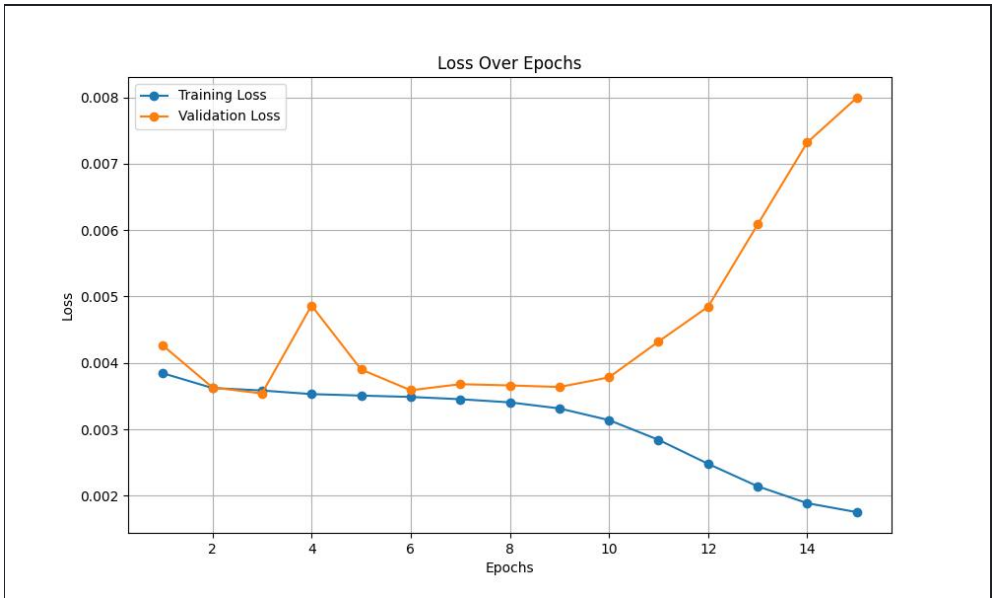
Shown below is a graph of a longer run did, in which the model seemingly overfit after the first epoch, despite having an accuracy only around 75% (same accuracy my model is getting still)




which as you can see is a more balanced prediction, and it is giving slightly higher values along the diagonal (correct predictions)

Another similar result for an even longer run with different hyperparameters:




**These are the things I tried that didn't resolve the issue:**

- tried adding data augmentations to prevent overfitting
    - affine transformations
    - rotations
    - resize
    - horizontal flips
    - color jitter
    - normalization
- tried adding weight decay to prevent overfitting
- tried various learning rates (1e-4, 1e3, 1e-2)
- tried various batch sizes (64, 128, 180)
- tried normalizing with different means and standard deviations (0.5 vs the recommended ones)
- tried freezing and unfreezing the weights for the resnet model
- tried using 2 outputs vs the standard 1000 for resnet (by replacing the last FC layer)
- tried different input image resizes (150 x 150, 224 x 224)
- tried different optimizers (SGD, ADAM)
- tried different schedulers (ExponentialLR, CosineAnnealingLR)
- tried training long times (no improvement/learning after first epoch)

**This is what eventually solved it:**

- Reworking the custom dataset code (I presume there was an error in the original method that I was using that led to incorrect labels)