UNIVERSITY OF MICHIGAN
Department of Electrical Engineering and Computer Science
EECS 445 — Introduction to Machine Learning
Winter 2021

Project 2 Solutions

Note: For all coding problems, Please refer to the solution code posted on Canvas.

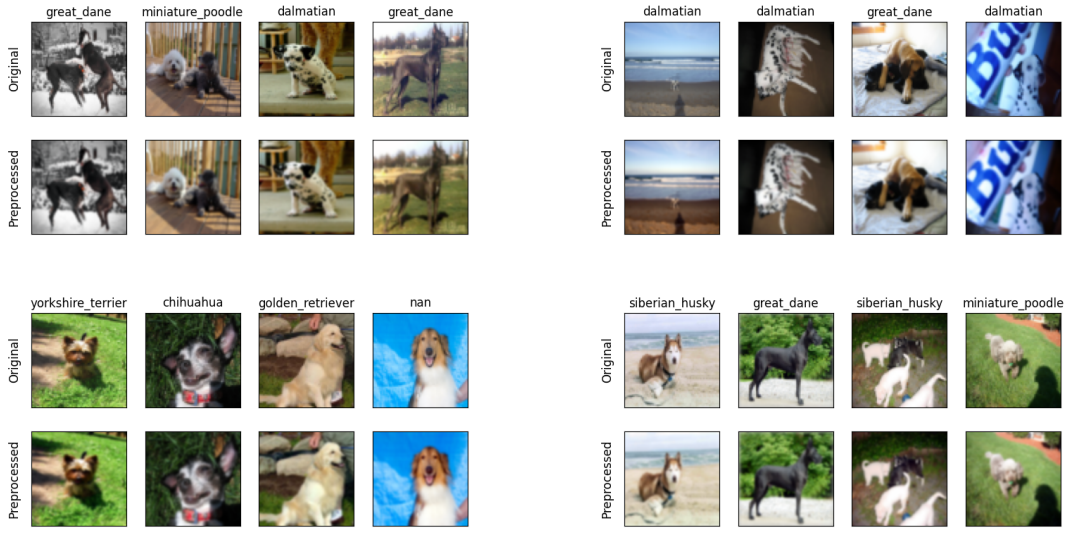
1 Data Preprocessing [10 pts]

(a) Short answers:

i. Training Mean: $[123.084, 117.488, 93.312]$, Training Std: $[62.729, 59.178, 61.434]$

ii. Rationale for standardizing with respect to only the training set:
We want our validation and test performance to be indicative of the ability of our model to generalize to new, unseen data. By taking the mean and std of only the training set, we ensure that we are not capturing any of the structure of the data partitions we are using for evaluation.

(b) Some sample images are shown below.



Below are some of the visible effects induced by the preprocessing:

- Slight attenuation in high frequency details (i.e. image is slightly blurry). This is actually likely due to denormalization and bicubic interpolation in matplotlib's imshow() and not any of our preprocessing.
- The shape and color of the dogs is still recognizable.
- Small variations in the brightness/color balance in some images.

- There is not a strong difference between original and preprocessed images.

2 Convolutional Neural Networks [30 pts]

(a) Number of learnable float-valued parameters:

The general formula for counting model paratemeters (layer weights) are:

- Convolutional layer: filter_width $\times$ filter_height $\times$ n_input_channels $\times$ n_output_channels
- Linear layer: n_input $\times$ n_output

Layer	Weights	Biases	Total
1: Conv_1	$5 \times 5 \times 3 \times 16$	16	$1,216$
2: Pool_1	0	0	0
3: Conv_2	$5 \times 5 \times 16 \times 64$	64	$25,664$
4: Pool_2	0	0	0
5: Conv_3	$5 \times 5 \times 64 \times 8$	8	$12,808$
6: FC_1	32×2	2	66

The above total to $\boxed{39,754}$ float-valued parameters.

(b) `Target` class: Solution code is ungraded.

(c) `predictions()` function: Solution code is ungraded.

(d) `train_cnn.py`: Solution code is ungraded

(e) Early stopping: Solution code is ungraded

(f) i. Training plot: Due to non-determinism of weight initialization, your plot might not look exactly the same, but it should follow the general trend, and the final validation accuracy should be around 90%.

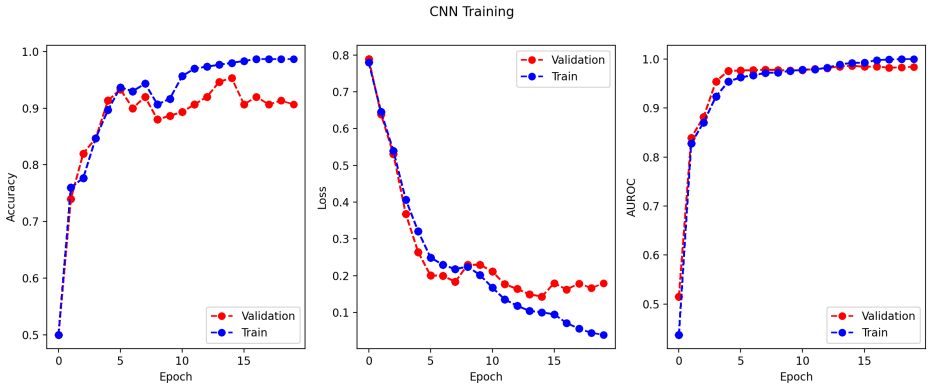


Figure 1: CNN training plot

Sources of noise in validation loss:

- From data: the data itself has multiple sources of noise: luminance (i.e., a mean different from the mean of the training data) or color balance.
- From the optimization procedure: ADAM, a variant of SGD is stochastic. Therefore, each update step will not necessarily decrease the training loss.
- Random shuffling training data at the end of each epoch.
- We directly optimize for training loss, but not validation loss. Therefore, even if we used true gradient descent on training data, validation loss has no guarantees of monotonic behavior.

ii. Effect of changing the patience:

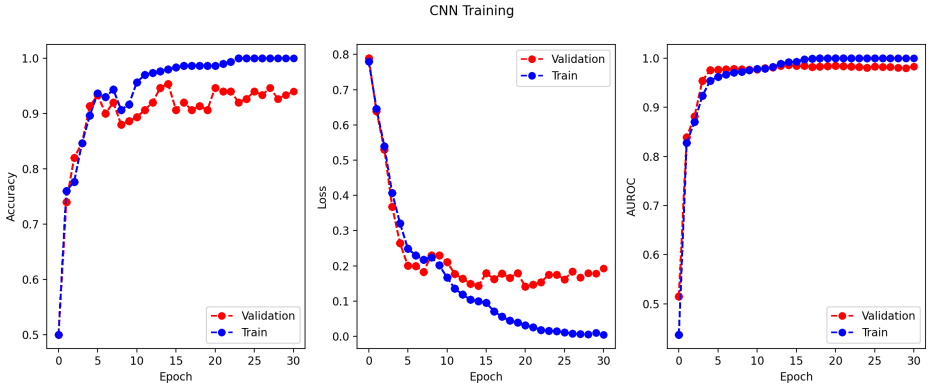


Figure 2: CNN training plot with increased patience

- Stopping epoch for patience of 5: **19**
- Stopping epoch for patience of 10: **30**
- A patience of 10 seems to work better for this dataset since we see better convergence (more of a plateau shape) in all three graphs. We can see that the loss graph for a patience of 5 looks like it is still decreasing so increasing the patience ensures that we have a model that has better predictions as it has optimized its objective function more.
- Answers will vary. For example, any time you see your objective function (such as minimizing loss) has not converged to a value such that it can still be optimized, you should increase your patience.

iii. Effect of changing the number of filters:

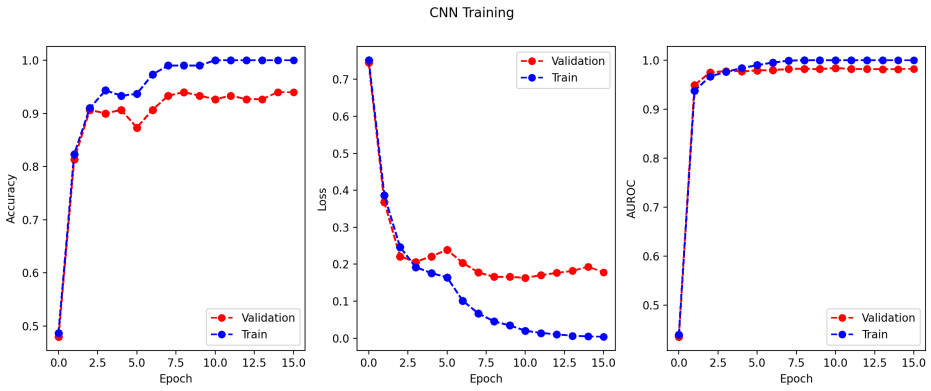


Figure 3: CNN training plot with increased filters

Could also look like this if weight initialization was not changed for this question:

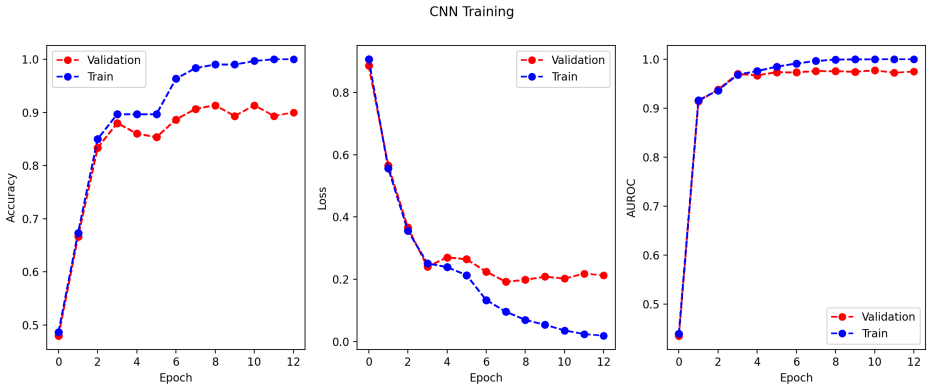


Figure 4: CNN training plot with increased filters

	Epoch	**Training AUROC**	**Validation AUROC**
8 filters	14 (may vary between 12-20)	0.9923	0.9865
32 filters	10 (may vary between 7-11)	1.0	0.984

- New size of input to FC layer: **128**
- Some observations about the changes in the 32 filter graphs
  - Slightly larger gap between train and validation performance (in accuracy and loss)
  - Smoother curves
  - Training set converges to AUROC of 1 quicker
  - Trained for fewer epochs with same patience value
- Some reasoning for each of the above changes
  - Model has more parameters now so it is more likely to slightly overfit to training data and not generalize as well to the validation data, thus we get a larger gap in performance.
  - We are converging quicker with more parameters so curves have less fluctuations and appear smoother.

- Again, more parameters means we can overfit to the training data and converge quicker.
- Converging quicker means early stopping will kick in sooner so we train for fewer epochs with more parameters.

(g) Evaluating on the test set:

	Training	**Validation**	**Testing**
Accuracy	0.98	0.9953	0.55
AUROC	0.9923	0.9865	0.6284

i.

- The training and validation performance is very similar so there is no immediate evidence of overfitting.

ii.

- The testing performance is far lower than the validation performance. This means our validation set was **not** a good representation of our testing set. Many possible hypothesis answers. For example, the validation set may include some bias that is not present in the test set, i.e. the validation images all have some feature that is not heavily present in the testing set - like green backgrounds!

3 Visualizing what the CNN has learned [10 pts]

(a) Calculating L^1:

$$\alpha_1^1 = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}\frac{\partial y^1}{\partial A_{ij}^{(1)}} = \frac{3}{16}$$

$$\alpha_2^1 = \frac{1}{16}\sum_{i=1}^{4}\sum_{j=1}^{4}\frac{\partial y^1}{\partial A_{ij}^{(2)}} = \frac{7}{16}$$

$$L^1 = ReLU(\alpha_1^1 * A^{(1)} + \alpha_2^1 * A^{(2)})$$

$$L^1 = \begin{bmatrix} 10 & 10 & 13 & 10 \\ 17 & 20 & 17 & 14 \\ 14 & 17 & 7 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \frac{1}{16}$$

(b) Visualizing plot:
The model seems to be making classification decisions using background features, and often times fails to activate on the dog at all. Many of the images are segmented such that the collie classification is influenced almost entirely by the grass background.

(c) Revisiting 2 (g):
Since the model is making decisions based on the background and not the actual dog features, we can hypothesize that the background is highly correlated to the label so our model has learned to use this

to classify the dogs. However, since our test set performance is so much lower, it seems likely that this shortcut does not generalize well to the test set, possibly because the backgrounds from the training/validation set are not representative of the test set. In other words, since our model did not actually learn to distinguish breeds and instead learned to classify based on background, it is not generalizable to other scenarios. This confirms our hypothesis that the training/validation set are not representative of the test set, since our learned shortcut does not transfer well

4 Transfer Learning Data Augmentation [30 pts]

4.1 Transfer Learning

1. Implement Source architecture: Code is ungraded.
2. Fill in `train_source.py`: Code is ungraded.
3. Train Source model:

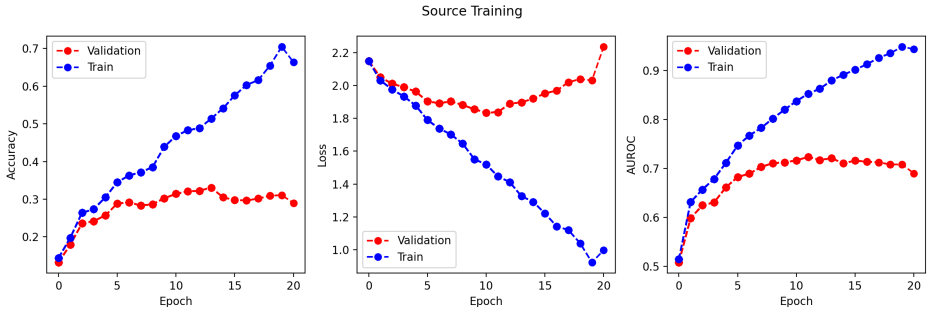


Figure 5: Source Training Plot

4. Create confusion matrix.

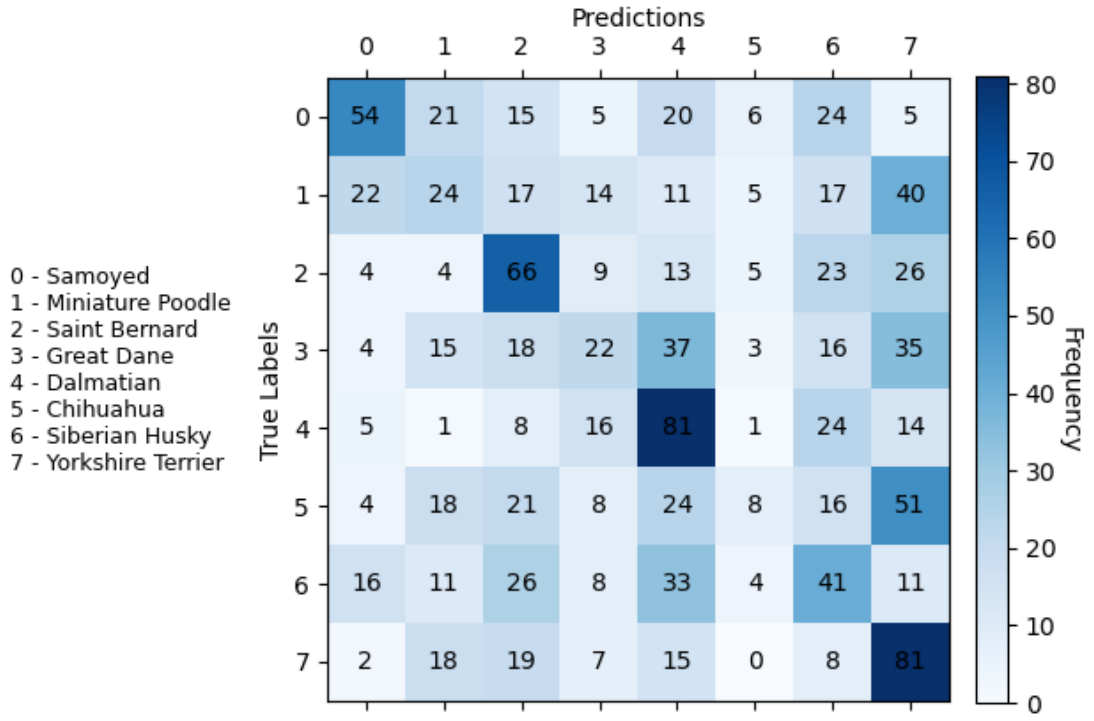


Figure 6: Confusion Matrix

Yorkshire terrier and Dalmatians were the most accurate, while Chihuahuas were the most incorrectly classified, mostly as Yorkshire terriers. Our model tended to predict most images as either a Dalmatian or Yorkshire terrier, which led to the accuracy for two classes. It seems that most smaller dogs such as miniature poodle or chihuahua were predicted to be a terrier, and bigger dogs like Great Dane or Husky were predicted to be a Dalmatian, another larger dog. Dalmatians also have very distinct coloration, whereas chihuahuas have a more generic color and shape, leading to the model being accurate for Dalmatians and not chihuahuas.

5. Implement freezing layers: Code is ungraded.

6. Compare classifiers:

	AUROC		
	TRAIN	**VAL**	**TEST**
Freeze all CONV layers (Fine-tune FC layer)	0.838	0.859	0.7704
Freeze first two CONV layers (Fine-tune last CONV and FC layers)	0.9889	0.9774	0.7664
Freeze first CONV layer (Fine-tune last 2 conv. and fc layers)	0.9912	0.981	0.722
Freeze no layers (Fine-tune all layers)	0.994	0.9799	0.7356
No Pretraining or Transfer Learning (Section 2 performance)	0.9923	0.9865	0.6284

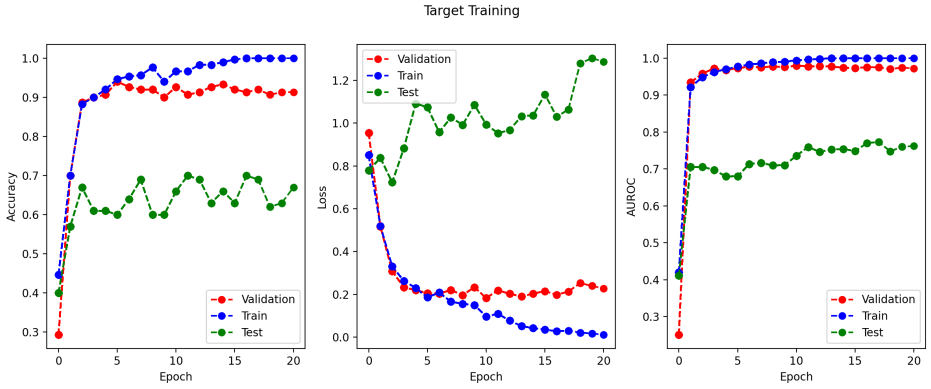


Figure 7: No Layers Frozen

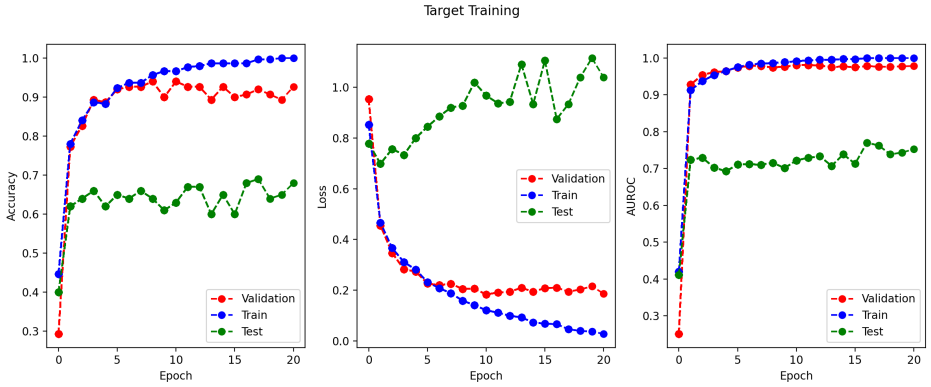


Figure 8: First CONV layer frozen

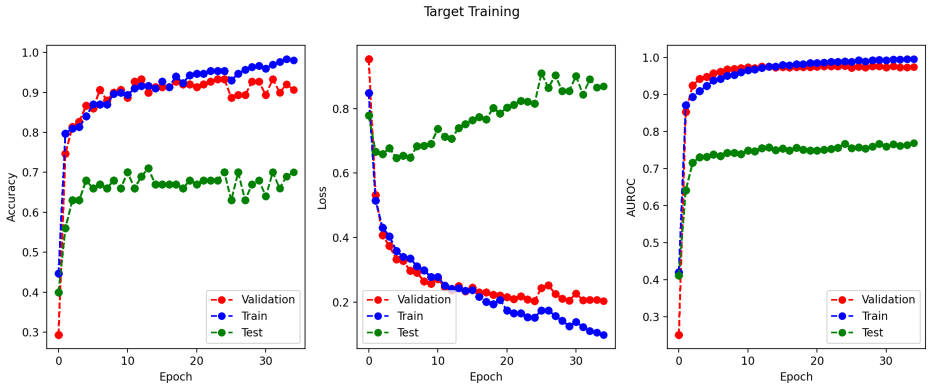


Figure 9: First two CONV layers frozen

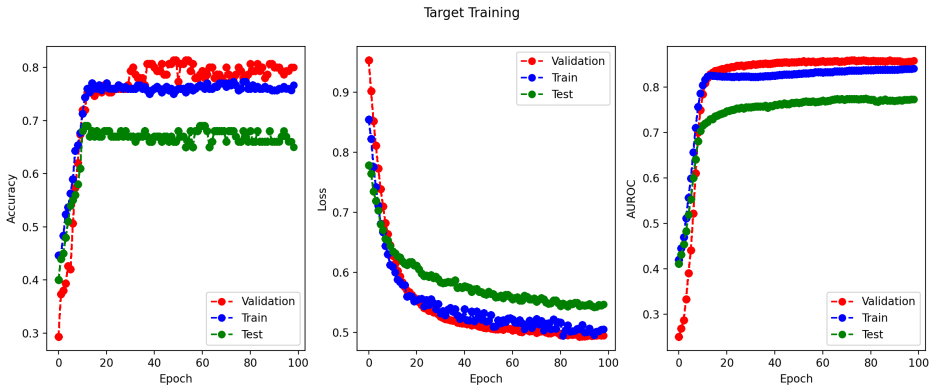


Figure 10: All CONV layers frozen

As we freeze more layers, the test AUROC generally increases. This is due to the model no longer being able to learn the bias of the grass in classification. The train and validation AUROC tend to decrease, but still hover at very high values since the model is already trained on similar task of differentiating other dog breeds. The transfer learning produced better results, even with no layers frozen, than the section 2 performance in test AUROC, indicating that the initialization itself actually helped performance.

4.2 Data Augmentation

1. Complete data augmentation code: Code is ungraded.

2. Experiment with different data augmentation settings: See Figure 11 for example training plots. Exact curves may vary due to random initialisations.

	AUROC		
	TRAIN	**VAL**	**TEST**
Rotation (keep original)	0.9861	0.9737	0.7408
Grayscale (keep original)	0.9799	0.9796	0.7292
Grayscale (discard original)	0.9807	0.8567	0.7516
No augmentation (Section 2 performance)	0.9923	0.9865	0.6284

3. How do validation and training plots change?

   Both the Rotation and Grayscale (keep original) training plots should look similar to the original CNN training plots (i.e. the validation and training loss are both high) because they both still allow for learning the shortcut. The Grayscale (discard original) training plot should have significantly worse validation performance compared to training performance because the shortcut can no longer be learned, so the model must learn to actually discern between the dogs.

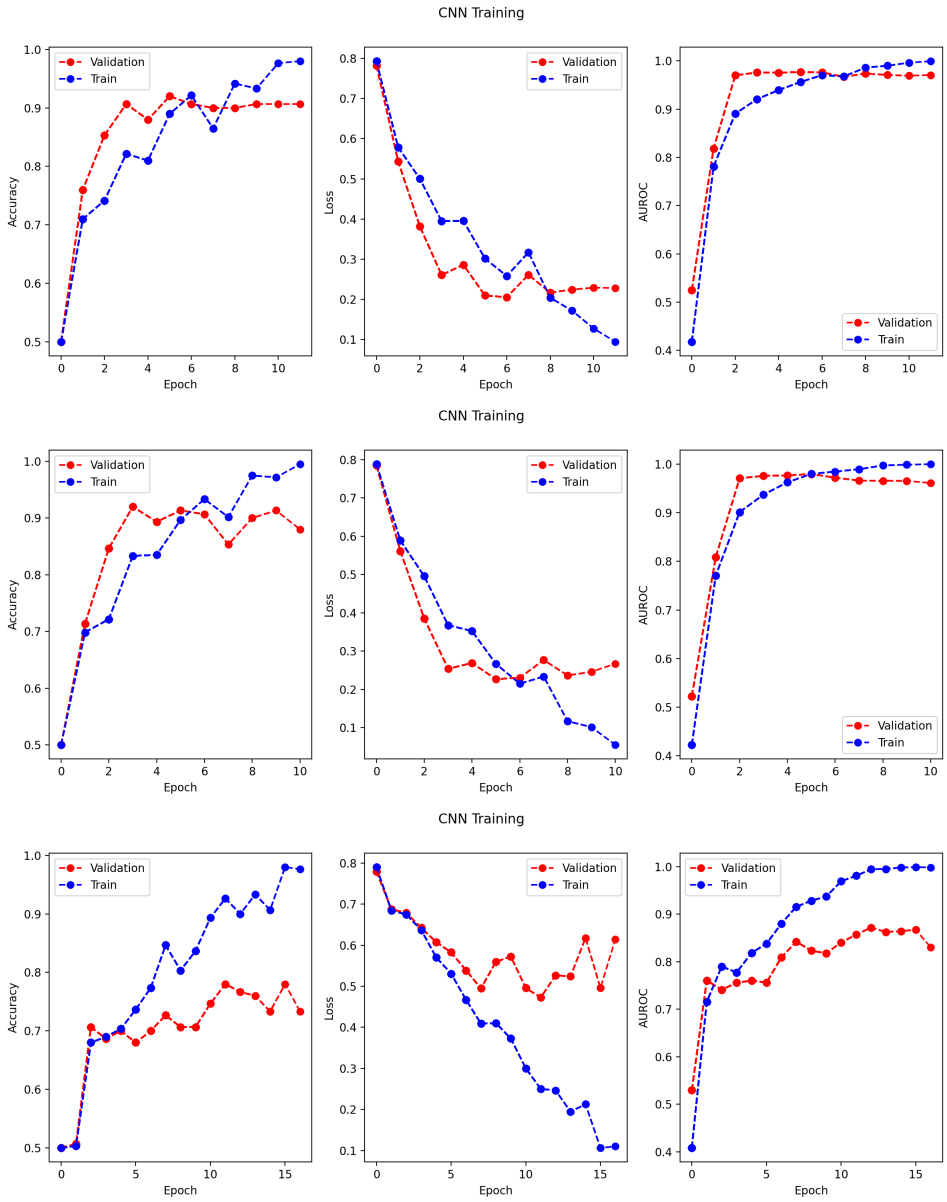


Figure 11: From top to bottom: Rotate, Gray (keep), Gray (discard)

5 Challenge [20 pts]

This part is open-ended.