1. Data Processing

(a)

i

Mean: [118.263, 118.004, 118.594]

Std: [66.307, 69.039, 75.332]

ii.

① By extracting the mean and standard deviation from each color channel, we can normalize each channel of the image. and thus reducing the possibility of overfitting.

Other data partitions are uncommon methods, and are useless for normalizing the image.

(2) processing the image with their mean and std is a linear process (shift the data by the value of mean and then scale it).

Therefore, the image still contains the original information. There's no information loss by processing the image with mean and standard deviation. And this will be beneficial for the model to take outliers into account. After normalization, outliers won't have great impact on the predicting result.

(b)

2. Convolutional Neural Networks

a.

Layer 1: 16 filters

each filter size 5×5×3

bias: 1

⇒ 16 × (5×5×3+1) = 16 × (76) = 1216

Layer 2: 0

Layer 3: 64 filters

filter size: 5×5×16

bias: 1

⇒ 64 × (5×5×16+1) = 25664

Layer 4: 0

Layer 5: 8 filters.

filter size: 5×5×64

bias 1

⇒ 8 × (5×5×64+1) = 12808

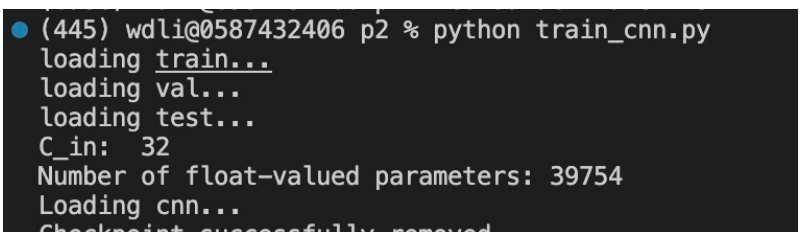
Layer 6: 1 bias: 32 inputs, 2 outputs

(32+1) ×2 = 66

In total: 1216 + 25664 + 12808 + 66

= 39754

The photo to the right verifies this conclusion.



(f)

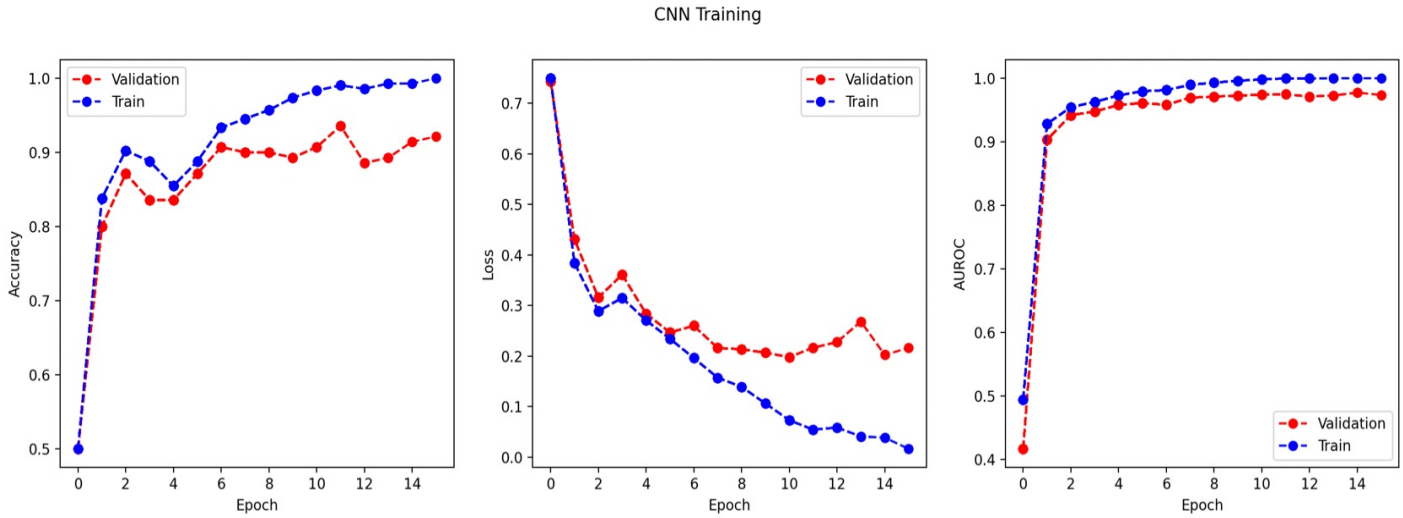
(i)

1. The reason why it doesn't monotonically decreasing is that the model may overfit when epoch is high, thus validation error doesn't always decrease when epoch increases

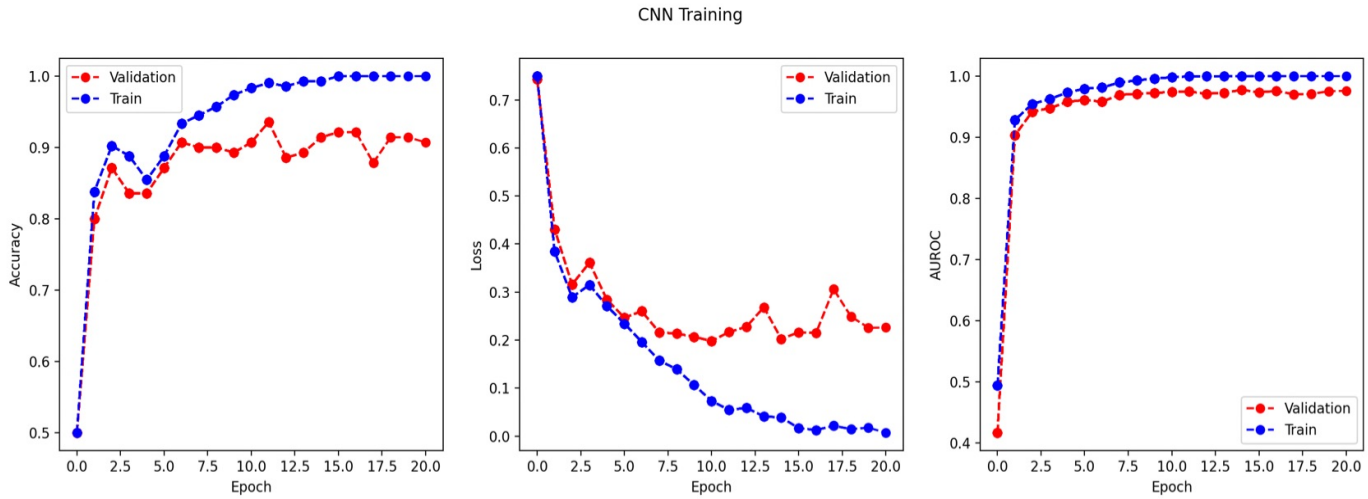
2. the dataset may contain some noise. when we apply mini-batch SGD to it, we may receive some random information. this will result in oscillation in model prediction.

(ii) when patience = 5 model stop training at 15

when patience = 10, model stop training at 20



The above figure is generated by using a patience of 5



The above figure is generated by using a patience of 10

Based on the training graph, which patience value works better and why?

Patience = 5 works better

Compared to loss value when patience = 10, the model that is generated by patience = 5 is less overfitting. And the value when validation loss equals training loss is less than that of model of patience 10.

And since both graph overfits when epoch is high, this shows it's not necessary to use patience=10 since the extra work is meaningless (e.g. no performance increase).

Both of these two model show quite similar validation performance in terms of accuracy, loss and AUROC, but the model of patience=5 needs less time to training(and patience=10 may aggravate overfitting). So, in terms of efficiency, model with patience = 5 is better.

In what scenario might increased patience be better?

When the model fluctuates quite intensely, and the value of the model may suddenly drop a lot due to the instability of the model. For example, when the model comes across a local minima, if the patience is not large enough, the figure will stop, leaving high error because the model doesn't reach global minimum. If the patience is higher, it will keep pending in the calculation for a longer time, which means it will be more possible to reach global minimum.

(iii): The new size is 64 * 2 * 2 = 256

	Epoch	Training AUROC	Validation AUROC
8 filters	10	0.9982	0.9743
64 filters	7	0.9993	0.9694

How does the performance vary as we increase the number of filters from 8 to 64? What might account for this change?

The validation performance decreases a little bit, but the training performance of 64 filters is better than that of 8 filters in terms of AUROC. When the number of filters is 64, we reach minimum loss when Epoch=7, but we reach minimum loss with Epoch=10 when using 8 filters. Although generally, the accuracy, loss and AUROC of both models are quite similar, the model with 64 filters always get lower maximum performance in terms of validation.

This might because of

1. the overfitting. Since the dataset has some noise. 64-sized filter lets the model receive too much “noise”, this may lead to overfitting.
2. too complex model. 64-sized model might be too complex structure for this dataset, so the model might overfit the data.

(g)

(i)

	Training	Validation	Testing
Accuracy	0.9833	0.9071	0.625
AUROC	0.9982	0.9743	0.6827

(ii)

Yes.
From the comparison between training performance and validation performance, we can notice that the training accuracy is much greater than validation accuracy (higher for about 0.075) and the AUROC of training model is also greater than that of validation model (for about 0.01). Although the AUROC score does not show far exceeding, the accuracy score of training and validation model changes much.

(iii)

Compare the validation and test performance.
From the table, we can notice that the testing performance is much lower than validation performance. They are not very similar.

What trend do you see here?
It seems the test performance is much lower than validation performance. And validation performance is lower than training performance. So, in terms of performance(accuracy & AUROC), from higher to lower: Training > Validation > Testing.

Hypothesize a possible explanation for such a trend.
This might due to the bias learned from the training dataset. The performance of the model shows that the training performance is slightly greater than validation performance. However, the training performance and validation performance is much greater than testing performance. This shows that our model learned some not important features from the training set that are not generalizable on testing set.

3. Visualizing what the CNN has learned

(a).

$$L^1 = \mathrm{ReLU}\left(\sum_{k=1}^{2} \alpha_k^1 A_k^1\right)$$

$$= \mathrm{ReLU}\left[\sum_{k=1}^{2} \left(\frac{1}{Z}\sum_i \sum_j \frac{\partial y^1}{\partial A_{ij}^k}\right) \cdot A_k^1\right]$$

$$\alpha_1^1 = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^1}{\partial A_{ij}^1} = \frac{1}{16}\cdot(-1+1-2-1-1+1+1+1+2+2)$$

$$= \frac{3}{16}$$

$$\alpha_2^1 = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^1}{\partial A_{ij}^2} = \frac{1}{16}\cdot(1+2+2+2+2+1+1-1-2-1)$$

$$= \frac{7}{16}$$

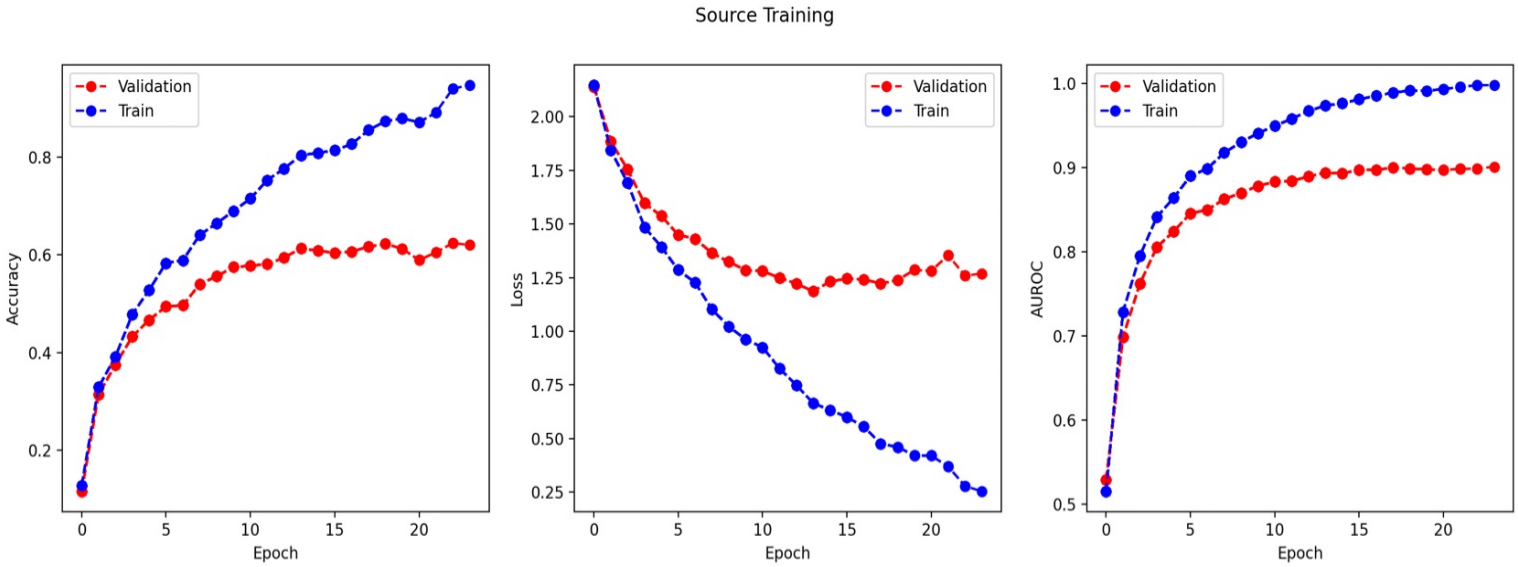
$$L^1 = \mathrm{ReLU}\left(\sum_{k=1}^{2} \alpha_k^1 A^k\right) = \alpha_1^1 A^1 + \alpha_2^1 A^2$$

$$= \begin{pmatrix} \frac{5}{8} & \frac{5}{8} & \frac{13}{16} & \frac{5}{8} \\ \frac{17}{16} & \frac{5}{4} & \frac{17}{16} & \frac{7}{8} \\ \frac{7}{8} & \frac{17}{16} & \frac{7}{16} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

(b)
From the problem, we are told that the lighter colored area of the image is more important for the model to distinguish. Because these lighter areas are generally distributed on blue objects like sky, blue clothes. Also, some lighter area is also distributed on places that transited from one object to another, like roof, column, and window of the building.

(c)
Yes.
From the picture, I notice that the lighter part does not have a very precise distribution on these buildings, most of lighter area even distributed on objects like sky and humans. Considering that the model has not bad validation performance, I can confirm that the model learned some bias that is not important (non-generalizable) features from training set. The blue objects can be considered as bias.

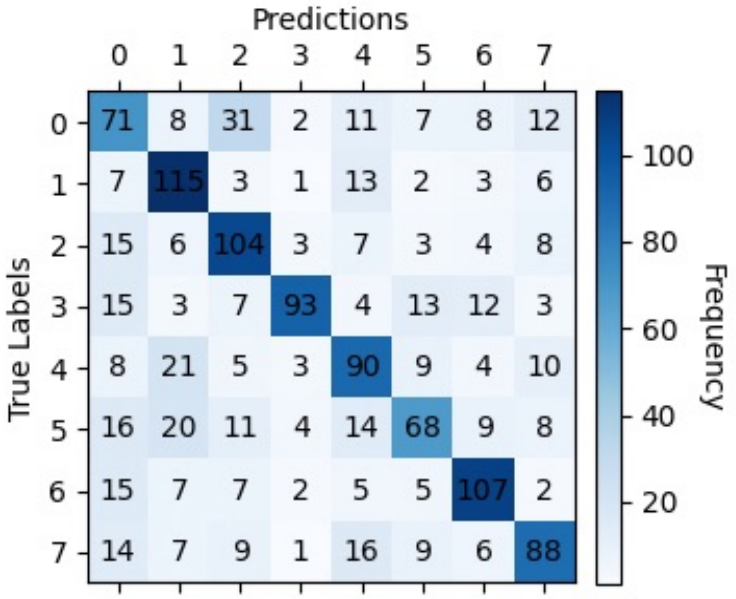
4.1 c



Epoch Number corresponding to the lowest validation loss: 13

4.1 d

0 - Colosseum
1 - Petronas Towers
2 - Rialto Bridge
3 - Museu Nacional d'Art de Catalunya
4 - St Stephen's Cathedral in Vienna
5 - Berlin Cathedral
6 - Hagia Sophia
7 - Gaudi Casa Batllo in Barcelona



1-Petronas Towers is predicted the most accurate, while 5-Berlin Cathedral is predicted the least accurate. I get the conclusion from the above graph. The formula for accuracy is

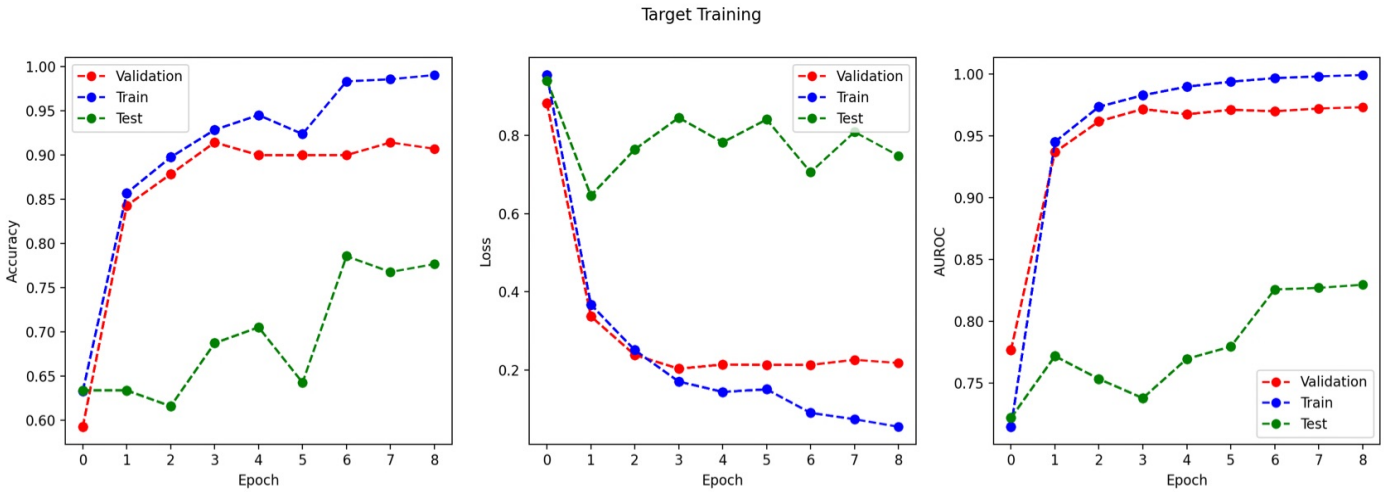
- Accuracy = True Predictions / All Predictions

Since for all predictions, the denominator is equal. To compare the accuracy of each landmark, we simply need to compare the number of true labels. From the graph, we know that the number on the diagonal means true predictions. Among them, 1-Petronas Towers has highest true prediction number while 5-Berlin Cathedral has lowest true prediction number.

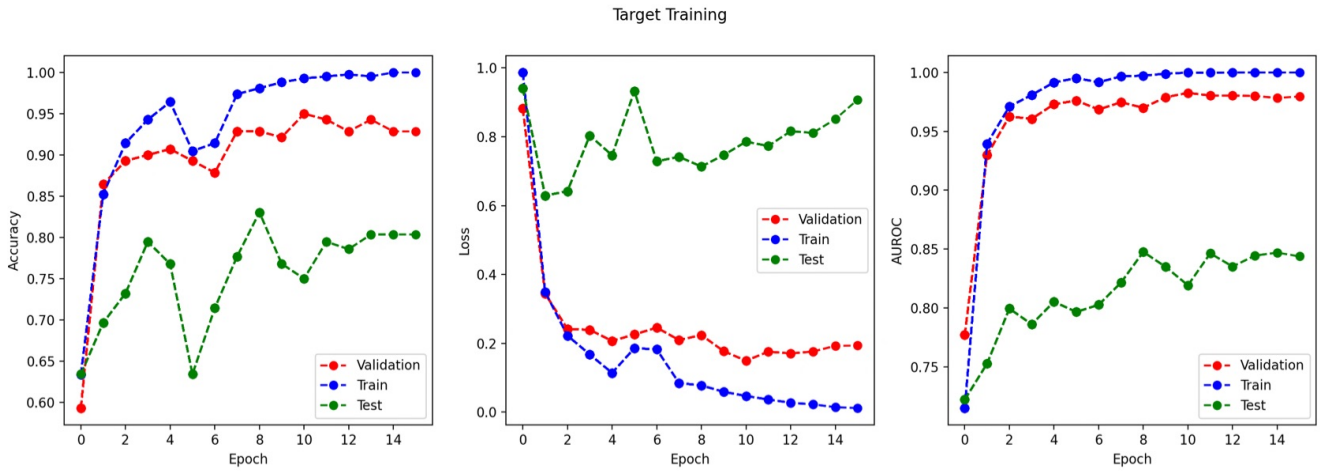
For the reason why this might be the case, I think (1) maybe Berlin Cathedral is not distinct from its background. Its roof is blue, which is quite close to the color of the sky. However the Petronas Towers has some distinct features like two identical buildings and one bridge in between. This distinguish itself from other buildings.(2) Maybe the training data for Berlin cathedral is not quite similar to that in the testing data set, so the data is not quite representative. So the model may learn some features that is not quite important in testing set.

4.1 f

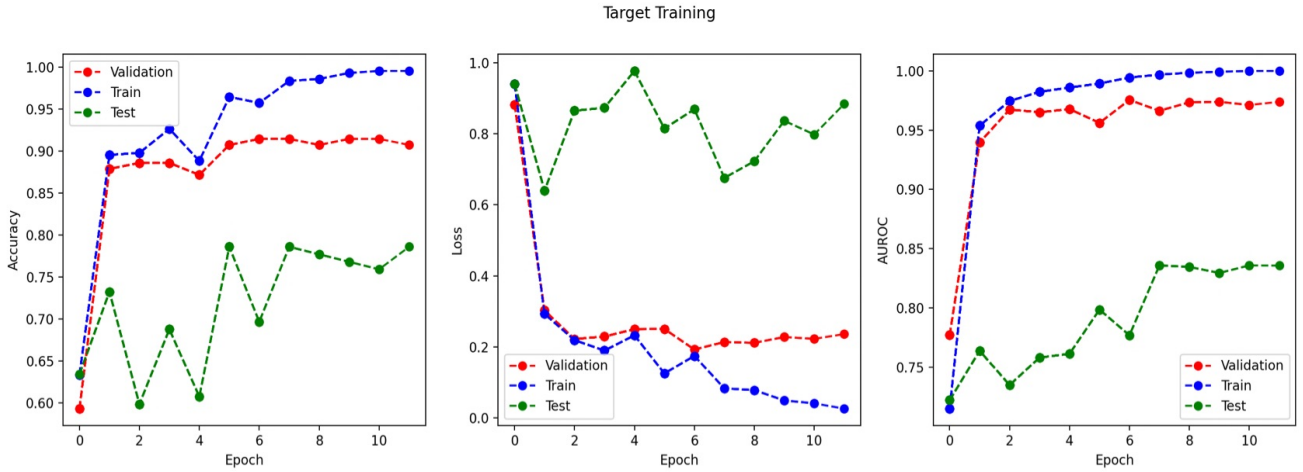
	AUROC		
	TRAIN	**VAL**	**TEST**
Freeze all CONV layers (Fine-tune FC layer)	0.9985	0.9747	0.8383
Freeze first two CONV layers (Fine-tune last CONV and FC layers)	0.9944	0.9755	0.7768
Freeze first CONV layers (Fine-tune last 2 conv, and fc layers)	0.9999	0.9827	0.8192
Freeze no layers (Fine-tune all layers)	0.983	0.9718	0.7379
No pertaining or Transfer Learning(Section 2(g) performance)	0.9982	0.9743	0.6827



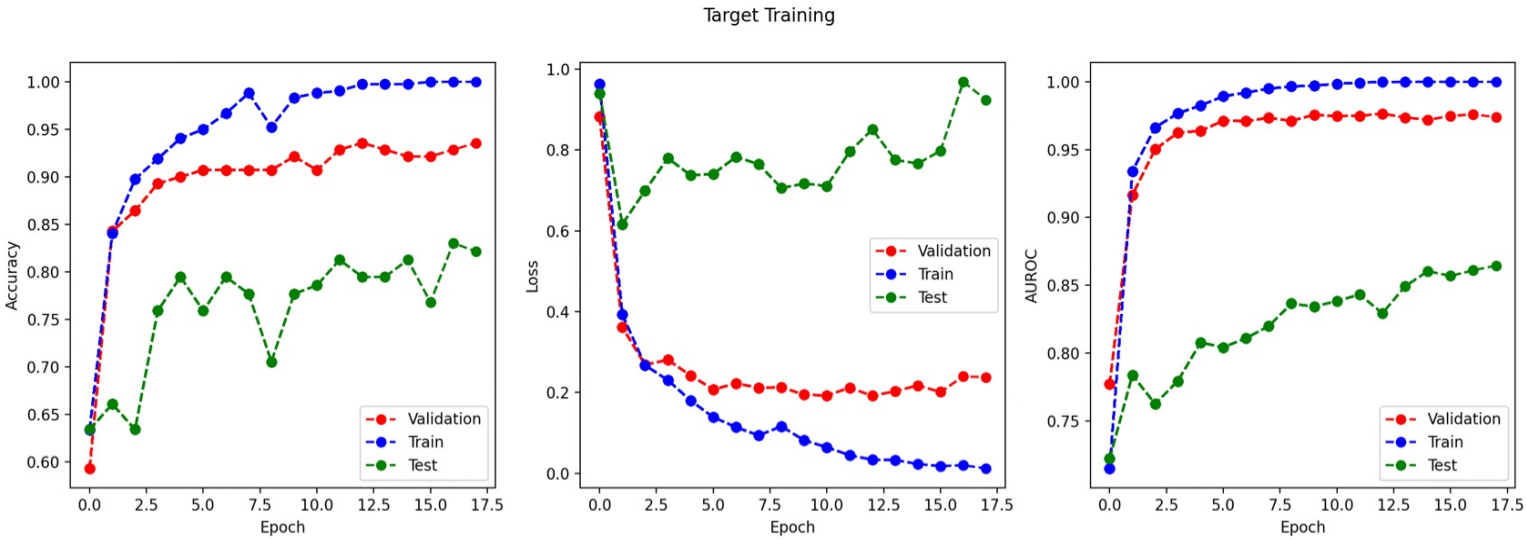
Freeze no layers



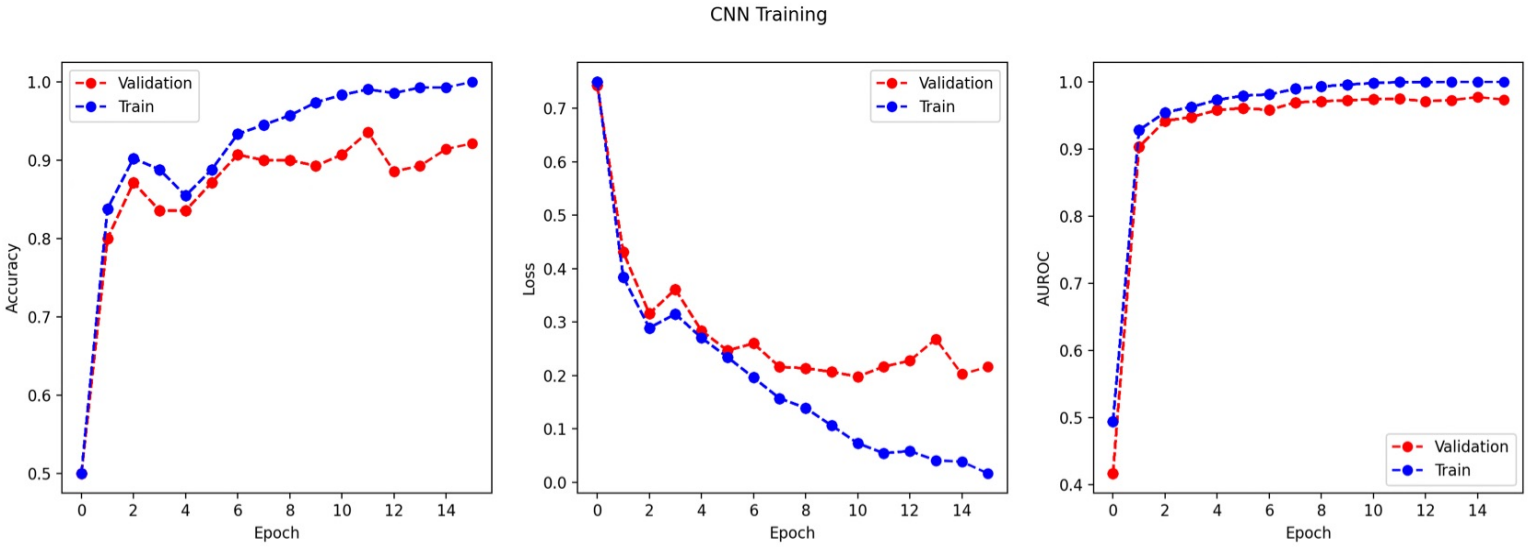
Freeze one layer



Freeze two layers



Freeze three layers



No pertaining or Transfer Learning(section 2(g))

Examine the results of using transfer learning

The performance is better than before. From the table, we can notice that the testing AUROC increases compared to the model that freezes no layers. And we can notice that testing AUROC increases when more layers are frozen (despite the case when first two layers are frozen). And all results of the first four rows are better than the last row, which uses the most primitive method and not using pertaining or transfer learning.

Does transfer learning help? Was the source task helpful? Why do you hypothesize it was (or wasn't) ?

From the table, we know that all first four rows have greater testing AUROC than the last row, which corresponds to the model that does not using transfer learning and source task. Although all four models have similar training and validation performance, the last model has worst testing performance, and this means transfer learning and source task are helpful.

How does freezing all convolutional layers compared to freezing just a subset, versus freezing none?

From the table, we find that the testing AUROC of the model that freezes all convolutional layers has best performance. This indicates freezing all convolutional layers achieves better performance than models that simply freeze a subset or nothing. And we can also compare the models that freeze a subset with the model that freezes nothing. The comparison shows that models that freeze just a subset still achieve better AUROC score than model that freezes nothing.

Why do you think this might occur?

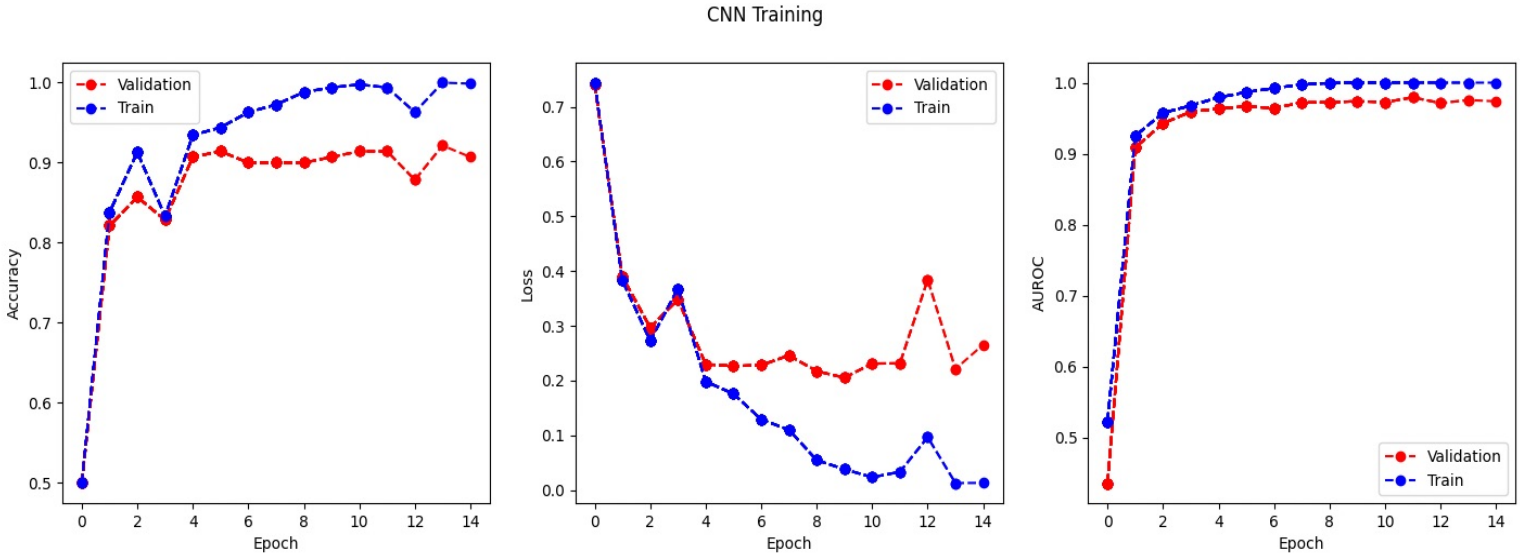
Because the source model is trained from the dataset that contains all 8 layers, the parameters in its convolutional layers might contain less bias than the model that trained on dataset that only has images of Pantheon and Homburg Imperial Palace. The more frozen convolutional layers, the more uesful and generalizable weights contained by the model. So, in the table, we can notice that for the testing AUROC, models that froze more layers have better performance when classifying data. In addition, if we freeze more layers, the model may contain fewer learnable parameters. This will reduce model complexity and can help solve overfitting to some extent.

However, we notice that the training error and validation error of the model whose convolutional was frozen is lower than that of the primitive model. This is because the primitive model tends to receive more information from training data and validation data and thus contains some bias in these two data sets.

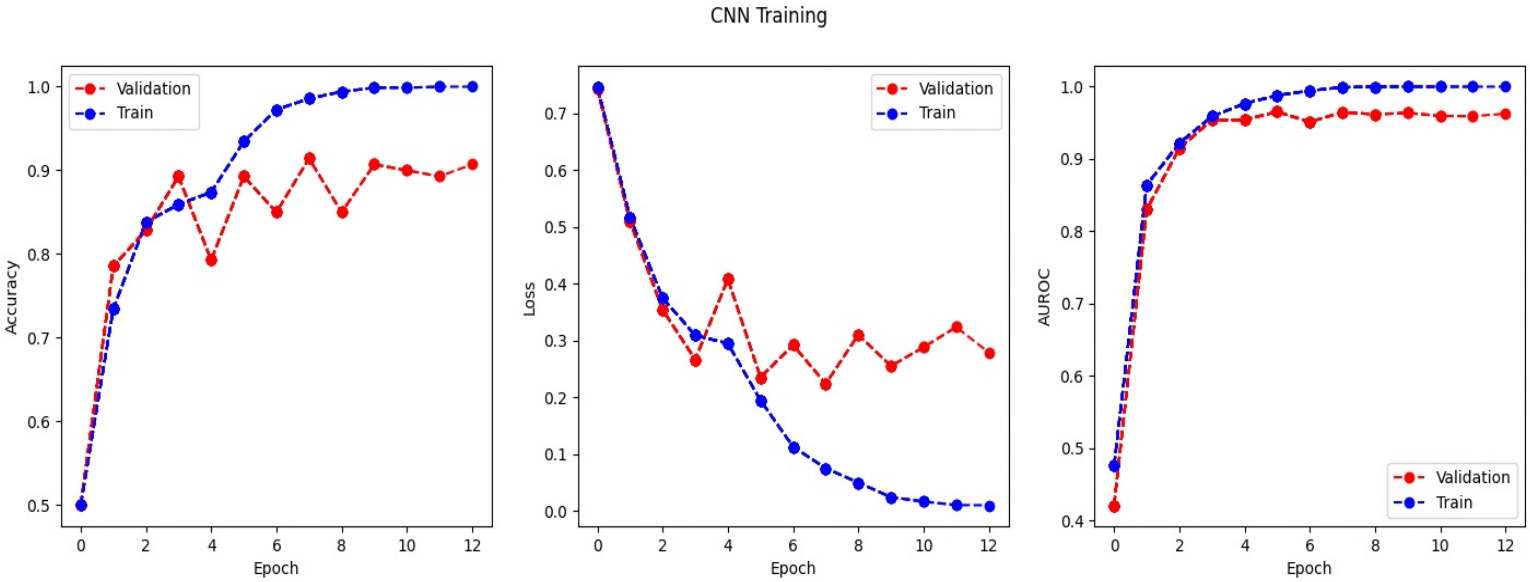
4.2 b

	AUROC		
	TRAIN	VAL	TEST
Rotation (keep original)	0.9998	0.9749	0.7602
Grayscale (keep original)	0.9994	0.9663	0.7618
Grayscale (discard original)	0.9946	0.911	0.8042
No augmentation	0.9982	0.9743	0.6827

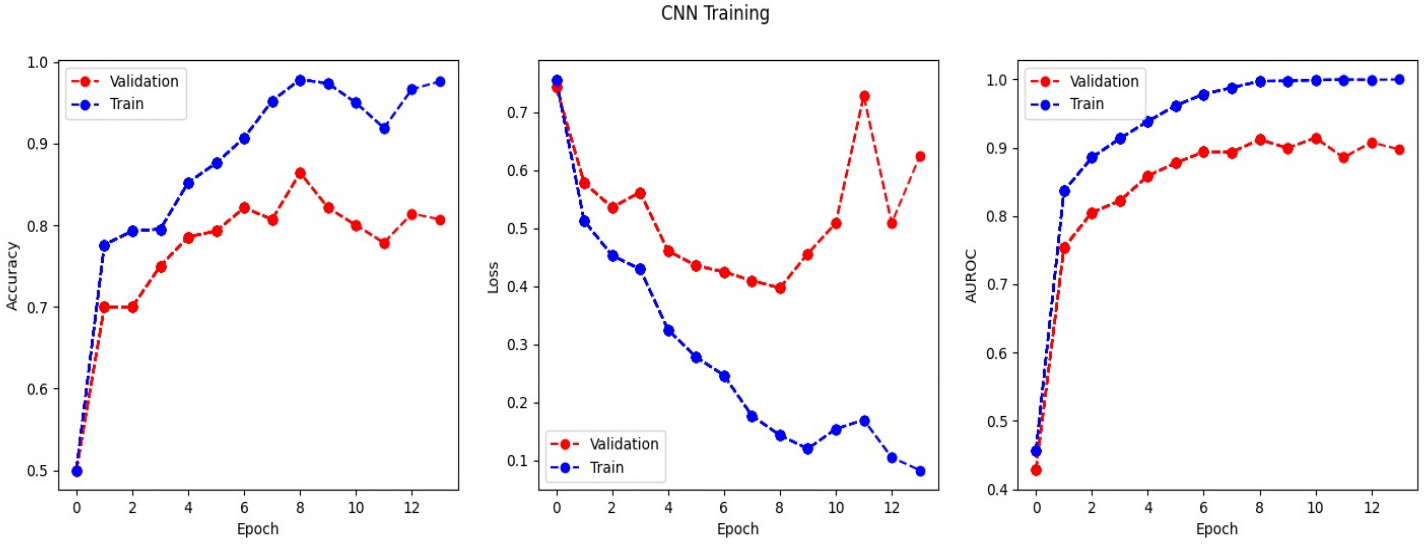
Rotation, keep original:



Grayscale, keep original:



Grayscale, discard original:



4.2 c

Compared to model that does not use any augmentation, The validation and training performances of the models that use rotation and grayscale are generally lower. For the training score, rotation, grayscale(keep original) has higher score than no augmentation. However, the model that uses grayscale but discards original images gets lowest training and validation score but highest testing score.

Maybe this is because the rotation and grayscale makes the model receive more information, instead of some misleading features like blue sky. Because of this, the bias in the training set is lower than the original training data set. The model receive less bias, so the model can detect more generalizable features. Therefore, the validation error is lower than the model that learned from the not-augmented dataset, but the testing error is higher than the primitive model.