Project2 Write Up

1(a)

i.

```
[(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason_wu$ python ]
 dataset.py
 loading train...
 loading val...
 loading test...
 Train:   300
 Val:     150
 Test:    100
 Mean: [123.968 117.887  93.942]
 Std:  [63.033 59.667 61.949]
```

ii.

If statistics are extracted from a validation or test set, data leakage is introduced, causing the model to behave on the test set in a "preference" for the statistical distribution of that data set, which does not accurately reflect the generalization ability of the model. Therefore, using the mean and standard deviation of the training set helps to ensure that the validation set and the test set remain "independent," thus testing the model's performance on real, unseen data.

1(b)

2(a)

Max Pooling layers don’t have parameters.

Layer 1: Convolutional Layer 1

Input channels: 3

Output channels: 16

Parameters per convolution kernel: 5x5=25

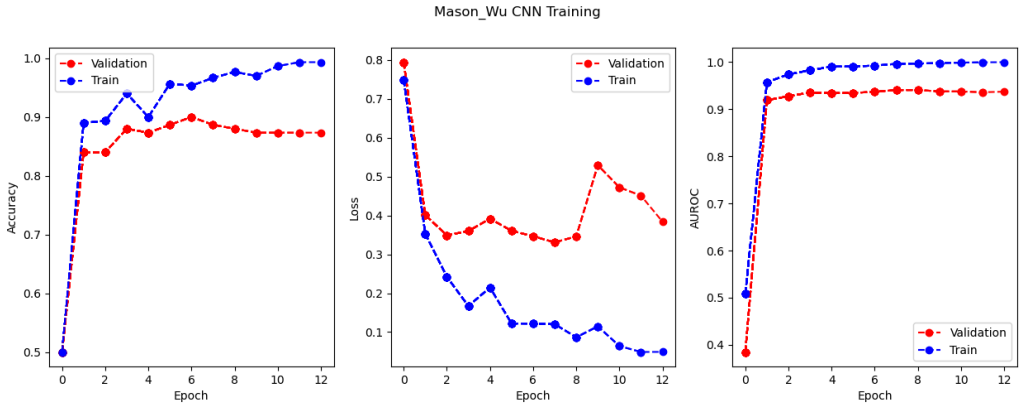
Number of weight parameters: 3x16x25=1200

Number of bias parameters: 16

Total parameters: 1200+16=1216
Layer 3: Convolutional Layer 2
Input channels: 16
Output channels: 64
Parameters per convolution kernel: 5x5=25
Number of weight parameters: 16x64x25=25600
Number of bias parameters: 64
Total parameters: 25600+64=25664
Layer 5: Convolutional Layer 3
Input channels: 64
Output channels: 8
Parameters per convolution kernel: 5x5=25
Number of weight parameters: 64x8x25=12800
Number of bias parameters: 8
Total parameters: 12800+8=12808
Layer 6: Fully Connected Layer 1 (Output Layer)
Input size: 32 (output from previous layer 8×2×2, the two 2 here is because we reduce the image size by 2 through pooling twice and we need to resize it back)
Output size: 2 (two categories)
Number of weight parameters: 32x2=64
Number of bias parameters: 2
Total parameters: 64+2=66
In total, we have 1216+25664+12808+66=39754 parameters.
2(b)
2(c)
2(d)
2(e)
2(f)
i.

```
(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason
_wu$ python train_cnn.py
loading train...
loading val...
loading test...
Number of float-valued parameters: 39754
Loading cnn...
No saved model parameters found
Epoch 0
```

```
Epoch 0
        Validation Accuracy:0.5
        Validation Loss:0.7946
        Validation AUROC:0.3849
        Train Accuracy:0.5
        Train Loss:0.7496
        Train AUROC:0.5083
Epoch 1
        Validation Accuracy:0.84
        Validation Loss:0.4011
        Validation AUROC:0.9191
        Train Accuracy:0.89
        Train Loss:0.353
        Train AUROC:0.9572
Epoch 2
        Validation Accuracy:0.84
        Validation Loss:0.349
        Validation AUROC:0.9276
        Train Accuracy:0.8933
        Train Loss:0.2423
        Train AUROC:0.974
Epoch 3
        Validation Accuracy:0.88
        Validation Loss:0.36
        Validation AUROC:0.9355
        Train Accuracy:0.94
        Train Loss:0.1669
        Train AUROC:0.9827
Epoch 4
        Validation Accuracy:0.8733
        Validation Loss:0.3918
        Validation AUROC:0.9342
        Train Accuracy:0.9
        Train Loss:0.2147
        Train AUROC:0.9904
Epoch 5
        Validation Accuracy:0.8867
        Validation Loss:0.3607
        Validation AUROC:0.9346
        Train Accuracy:0.9567
        Train Loss:0.1222
        Train AUROC:0.9906
Epoch 6
        Validation Accuracy:0.9
        Validation Loss:0.3466
        Validation AUROC:0.9371
        Train Accuracy:0.9533
        Train Loss:0.1218
        Train AUROC:0.9926
Epoch 7
        Validation Accuracy:0.8867
        Validation Loss:0.3308
        Validation AUROC:0.9406
        Train Accuracy:0.9667
        Train Loss:0.1204
        Train AUROC:0.9963
Epoch 8
        Validation Accuracy:0.88
        Validation Loss:0.3462
        Validation AUROC:0.9408
        Train Accuracy:0.9767
        Train Loss:0.0866
        Train AUROC:0.997
Epoch 9
        Validation Accuracy:0.8733
        Validation Loss:0.5295
        Validation AUROC:0.9378
        Train Accuracy:0.97
        Train Loss:0.1147
        Train AUROC:0.998
Epoch 10
        Validation Accuracy:0.8733
        Validation Loss:0.4725
        Validation AUROC:0.9377
        Train Accuracy:0.9867
        Train Loss:0.0641
        Train AUROC:0.9988
Epoch 11
        Validation Accuracy:0.8733
        Validation Loss:0.4511
        Validation AUROC:0.936
        Train Accuracy:0.9933
        Train Loss:0.0487
        Train AUROC:0.9996
Epoch 12
        Validation Accuracy:0.8733
        Validation Loss:0.3841
        Validation AUROC:0.9372
        Train Accuracy:0.9933
        Train Loss:0.0489
        Train AUROC:0.9998
Finished Training
```



- Mini-batch Variability

Within each epoch, the training process splits the data set into multiple mini-batches for training. Each small batch of data does not fully represent the entire data distribution, which leads to a certain randomness in the update of model parameters. Therefore, the validation loss obtained by the model after each epoch may fluctuate, showing that the validation loss is not always monotonically decreasing. The randomness of small batches introduces inconsistent data samples, making the model parameters slightly different after each epoch, and this randomness causes validation losses to occasionally rise.

- Learning-rate Adjustment

In the training process, the model will use the set learning rate to update parameters, and a higher learning rate will make the model's parameter update amplitude larger, which will cause the loss to fluctuate in the process of decline (especially in the initial training stage). When the learning rate is high, the model may produce overshooting during optimization, resulting in a certain degree of fluctuation in validation loss.

It stops at epoch 12.

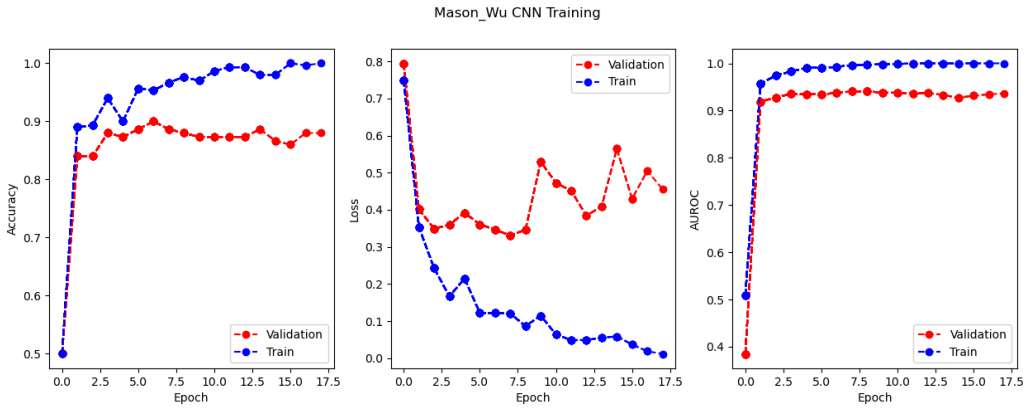
ii.

To be fair, I also start from epoch 0 in this case.

```
(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason_wu$ python train_cnn.py
loading train...
loading val...
loading test...
Number of float-valued parameters: 39754
Loading cnn...
Which epoch to load from? Choose in range [0, 12]. Enter 0 to train from scratch.
>> 0
Checkpoint not loaded
Checkpoint successfully removed
Epoch 0
        Validation Accuracy:0.5
        Validation Loss:0.7946
        Validation AUROC:0.3849
        Train Accuracy:0.5
        Train Loss:0.7496
        Train AUROC:0.5083
Epoch 1
        Validation Accuracy:0.84
        Validation Loss:0.4011
        Validation AUROC:0.9191
        Train Accuracy:0.89
        Train Loss:0.353
        Train AUROC:0.9572
Epoch 2
        Validation Accuracy:0.84
        Validation Loss:0.349
        Validation AUROC:0.9276
        Train Accuracy:0.8933
        Train Loss:0.2423
        Train AUROC:0.974
Epoch 3
        Validation Accuracy:0.88
        Validation Loss:0.36
        Validation AUROC:0.9355
        Train Accuracy:0.94
        Train Loss:0.1669
        Train AUROC:0.9827
Epoch 4
        Validation Accuracy:0.8733
        Validation Loss:0.3918
        Validation AUROC:0.9342
        Train Accuracy:0.9
        Train Loss:0.2147
        Train AUROC:0.9904
Epoch 5
        Validation Accuracy:0.8867
        Validation Loss:0.3607
        Validation AUROC:0.9346
        Train Accuracy:0.9567
        Train Loss:0.1222
        Train AUROC:0.9906
Epoch 6
        Validation Accuracy:0.9
        Validation Loss:0.3466
        Validation AUROC:0.9371
        Train Accuracy:0.9533
        Train Loss:0.1218
        Train AUROC:0.9926
Epoch 7
        Validation Accuracy:0.8867
        Validation Loss:0.3308
        Validation AUROC:0.9406
        Train Accuracy:0.9667
        Train Loss:0.1204
```

```
        Train AUROC:0.9963
Epoch 8
        Validation Accuracy:0.88
        Validation Loss:0.3462
        Validation AUROC:0.9408
        Train Accuracy:0.9767
        Train Loss:0.0866
        Train AUROC:0.997
Epoch 9
        Validation Accuracy:0.8733
        Validation Loss:0.5295
        Validation AUROC:0.9378
        Train Accuracy:0.97
        Train Loss:0.1147
        Train AUROC:0.998
Epoch 10
        Validation Accuracy:0.8733
        Validation Loss:0.4725
        Validation AUROC:0.9377
        Train Accuracy:0.9867
        Train Loss:0.0641
        Train AUROC:0.9988
Epoch 11
        Validation Accuracy:0.8733
        Validation Loss:0.4511
        Validation AUROC:0.936
        Train Accuracy:0.9933
        Train Loss:0.0487
        Train AUROC:0.9996
Epoch 12
        Validation Accuracy:0.8733
        Validation Loss:0.3841
        Validation AUROC:0.9372
        Train Accuracy:0.9933
        Train Loss:0.0489
        Train AUROC:0.9998
Epoch 13
        Validation Accuracy:0.8867
        Validation Loss:0.4092
        Validation AUROC:0.9328
        Train Accuracy:0.98
        Train Loss:0.0557
        Train AUROC:0.9996
Epoch 14
        Validation Accuracy:0.8667
        Validation Loss:0.5644
        Validation AUROC:0.9271
        Train Accuracy:0.98
        Train Loss:0.0588
        Train AUROC:1.0
Epoch 15
        Validation Accuracy:0.86
        Validation Loss:0.4295
        Validation AUROC:0.9316
        Train Accuracy:1.0
        Train Loss:0.0371
        Train AUROC:1.0
Epoch 16
        Validation Accuracy:0.88
        Validation Loss:0.5049
        Validation AUROC:0.9344
```

```
        Train Accuracy:0.9967
        Train Loss:0.0187
        Train AUROC:1.0
Epoch 17
        Validation Accuracy:0.88
        Validation Loss:0.4557
        Validation AUROC:0.9364
        Train Accuracy:1.0
        Train Loss:0.0119
        Train AUROC:1.0
Finished Training
```



It stops at epoch 17.

A patience value around 5 would be good: The loss curve of the verification set increases slightly at some epochs, but generally shows a downward trend. Therefore, if we choose an excessively small patience value (e.g. 2-3), we may cause the model to stop training prematurely, missing the opportunity for subsequent verification that the loss drops again. The fluctuation frequency of the verified loss curve in the chart is not high. A good patience number can be 5 or 6. This allows the model to continue training during a brief uptick in validation losses to see if subsequent validation losses drop again. If the patience value is too low, the opportunity to verify the overall decrease in set losses may be missed.

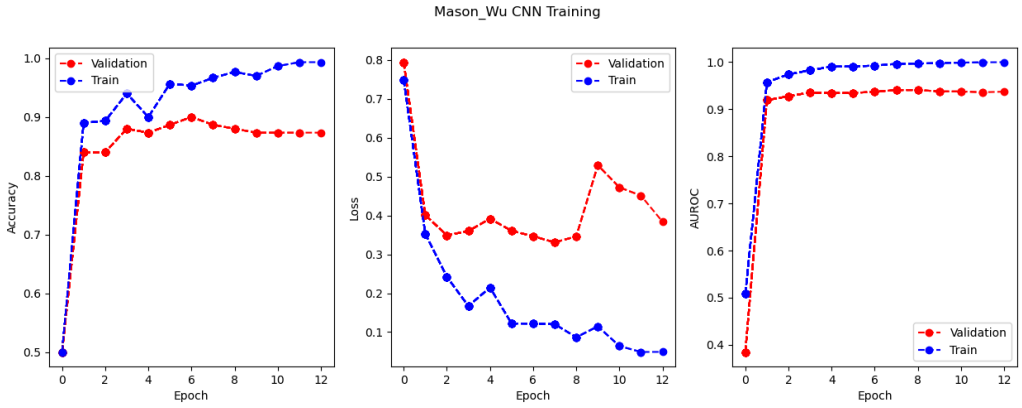
A larger patience value is appropriate for the following:

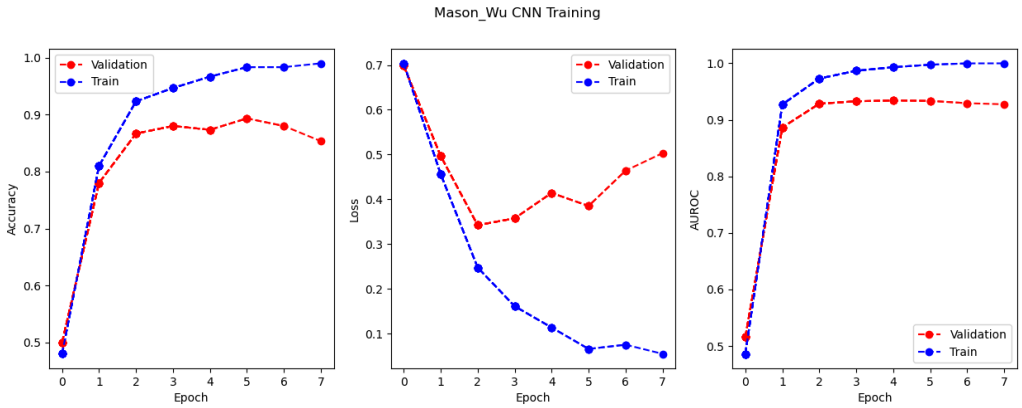
High noise data: If the fluctuations in validation losses are particularly frequent , a large patience value can prevent the model from stopping prematurely due to short-term fluctuations.

Long time training: When we have enough computing resources and time, we can increase patience and give the model longer time to find the optimal solution, thereby improving the final performance of the model.

iii.

The new size of the input to the fully connected layer is 64×2×2=256.





	Epoch	Training AUROC	Validation AUROC
8 filters	7	0.9963	0.9406
64 filters	2	0.974	0.9276

If you're asking about the variance of the performance in this table:

As the number of filters in the convolutional layer increases from 8 to 64, we can observe the following performance changes:

1. Training AUROC: When the number of filters is increased to 64, the training AUROC is still very high (0.9975), close to 1, indicating that the model has a strong ability to distinguish on the training set.
2. Validation AUROC: Validation AUROC also decreased slightly, from 0.936 for 8 filters to 0.9335 for 64 filters. This shows that with the increase of the number of filters, the generalization ability of the model on the verification set does not improve significantly, and even slightly decreases.

One possible cause is that as the number of filters increases, so does the number of parameters in the model. This allows the model to fit more detail on the training set, but also increases the risk of overfitting. Therefore, although the training AUROC approaches 1, the validation AUROC does not increase or even decrease slightly.

If you're asking about the variance of the performance the whole graph:

- Training set performance:

8 filters: Training loss drops faster and more significantly at first and declines gradually later with a few rise-again. The accuracy and AUROC curve of the training set tend to be stable, the AUROC is around 1.
64 filters: Training loss drops less significantly with no rise-again, and the training set's AUROC is closer to 1. This shows that the model fits the training data better.
- Validation set performance:
8 filters: The validation loss leveled off after declining over the first few (about 8) epochs, but the validation AUROC hovered around about 0.9 without a significant improvement. The accuracy and AUROC of the validation set were stable.
64 filters: The validation loss decreased less, but fluctuated slightly in the middle and late stages, showing some instability. The maximum value of the validation set AUROC is also closer to 0.93.

Change reason analysis:
Model capacity increases: By increasing the number of filters from 8 to 64, the capacity of the model becomes larger and can capture more complex features, resulting in training loss and a slight decrease in AUROC. More filters make the model perform better on the training set, but may also lead to the risk of overfitting.
More stable performance of the validation set: The loss of the model with 64 filters on the validation set fluctuates less, indicating that as the model fits more features, these features generalize to the validation set, leading to a good fit on it.
Overall, increasing the number of filters improved the performance of the training set, but the performance of the validation set was not significantly improved.

```
(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason
_wu$ python train_cnn.py
loading train...
loading val...
loading test...
Number of float-valued parameters: 129858
Loading cnn...
Which epoch to load from? Choose in range [0, 17]. Enter 0 to train
from scratch.
>> 0
Checkpoint not loaded
Checkpoint successfully removed
Epoch 0
        Validation Accuracy:0.5
        Validation Loss:0.6985
        Validation AUROC:0.5161
        Train Accuracy:0.48
        Train Loss:0.7038
        Train AUROC:0.4862
Epoch 1
        Validation Accuracy:0.78
        Validation Loss:0.4969
        Validation AUROC:0.8866
        Train Accuracy:0.81
        Train Loss:0.4564
        Train AUROC:0.9272
Epoch 2
        Validation Accuracy:0.8667
        Validation Loss:0.3424
        Validation AUROC:0.9285
        Train Accuracy:0.9233
        Train Loss:0.2478
        Train AUROC:0.9731
Epoch 3
        Validation Accuracy:0.88
        Validation Loss:0.3576
        Validation AUROC:0.9328
        Train Accuracy:0.9467
        Train Loss:0.1615
        Train AUROC:0.9868
Epoch 4
        Validation Accuracy:0.8733
        Validation Loss:0.4143
        Validation AUROC:0.9342
        Train Accuracy:0.9667
        Train Loss:0.1139
        Train AUROC:0.9931
Epoch 5
        Validation Accuracy:0.8933
        Validation Loss:0.3853
        Validation AUROC:0.9335
        Train Accuracy:0.9833
        Train Loss:0.066
        Train AUROC:0.9975
Epoch 6
        Validation Accuracy:0.88
        Validation Loss:0.4641
        Validation AUROC:0.9294
        Train Accuracy:0.9833
        Train Loss:0.0755
        Train AUROC:0.9998
Epoch 7
        Validation Accuracy:0.8533
        Validation Loss:0.5032
        Validation AUROC:0.9275
        Train Accuracy:0.99
        Train Loss:0.0551
```

```
        Train AUROC:0.9999
Finished Training
```

2(g)

```
[(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason]
_wu$ python test_cnn.py
loading train...
loading val...
loading test...
Loading cnn...
Which epoch to load from? Choose in range [0, 12]. Enter 0 to train
from scratch.
>> 7
Loading from checkpoint ./checkpoints/target/epoch=7.checkpoint.pth.
tar?
=> Successfully restored checkpoint (trained for 7 epochs)
Epoch 7
        Validation Accuracy:0.8867
        Validation Loss:0.3308
        Validation AUROC:0.9406
        Train Accuracy:0.9667
        Train Loss:0.1044
        Train AUROC:0.9963
        Test Accuracy:0.65
        Test Loss:1.31
        Test AUROC:0.6836
```

i.

	Training	Validation	Testing
Accuracy	0.9667	0.8867	0.65
AUROC	0.9963	0.9406	0.6836

ii.

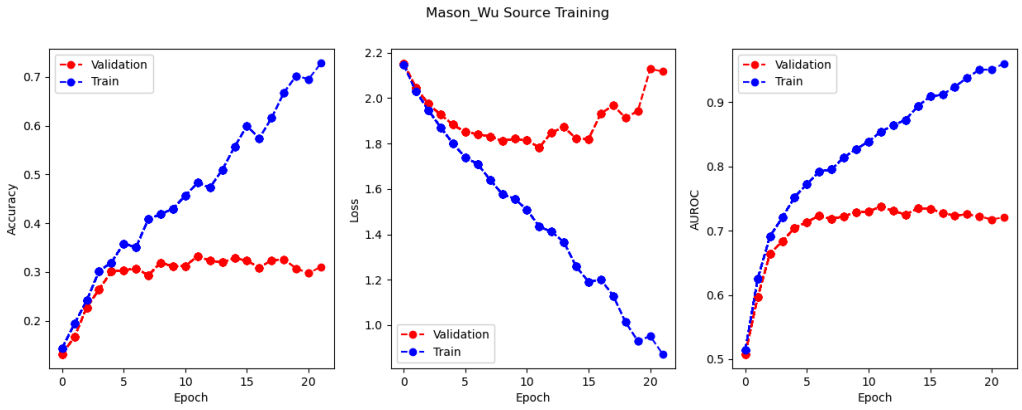
We have evidence of overfitting here. (The training accuracy is 8 percentage points higher than the validation accuracy, the training AUROC is about 5.57 percentage points higher than the validation AUROC, indicating a significant gap.)

iii.

As you can see from the images, the data for the Positive label seems to be mostly collies (in Carly's kennel house), while the data for the Negative label is golden retrievers (in Paco's kennel house). This suggests that the labeling of positive and negative samples in the dataset is actually based on different dog breeds (Collie vs. Golden Retriever) to be tagged. Therefore, the model may learn the characteristics of different breeds of dogs, rather than other general characteristics.

Verification set performance: The accuracy of verification set was 87.33%, and the AUROC was 0.936, which performed relatively well.
Test set performance: The accuracy of the test set dropped to 64% with an AUROC of 0.6728, significantly lower than the performance of the validation set.
The model performs well on the training set and the validation set, but significantly poorly on the test set.
The probable causes of it are:
Inconsistent data distribution: The data distribution of the test set may be different, containing more uncommon variants or other breeds, resulting in a model that is difficult to identify correctly and performance deteriorates.
Overfitting: The model may overfit the features of the training set and the validation set during training, resulting in poor performance on test sets with large distribution differences.

3(a)
3(b)
3(c)



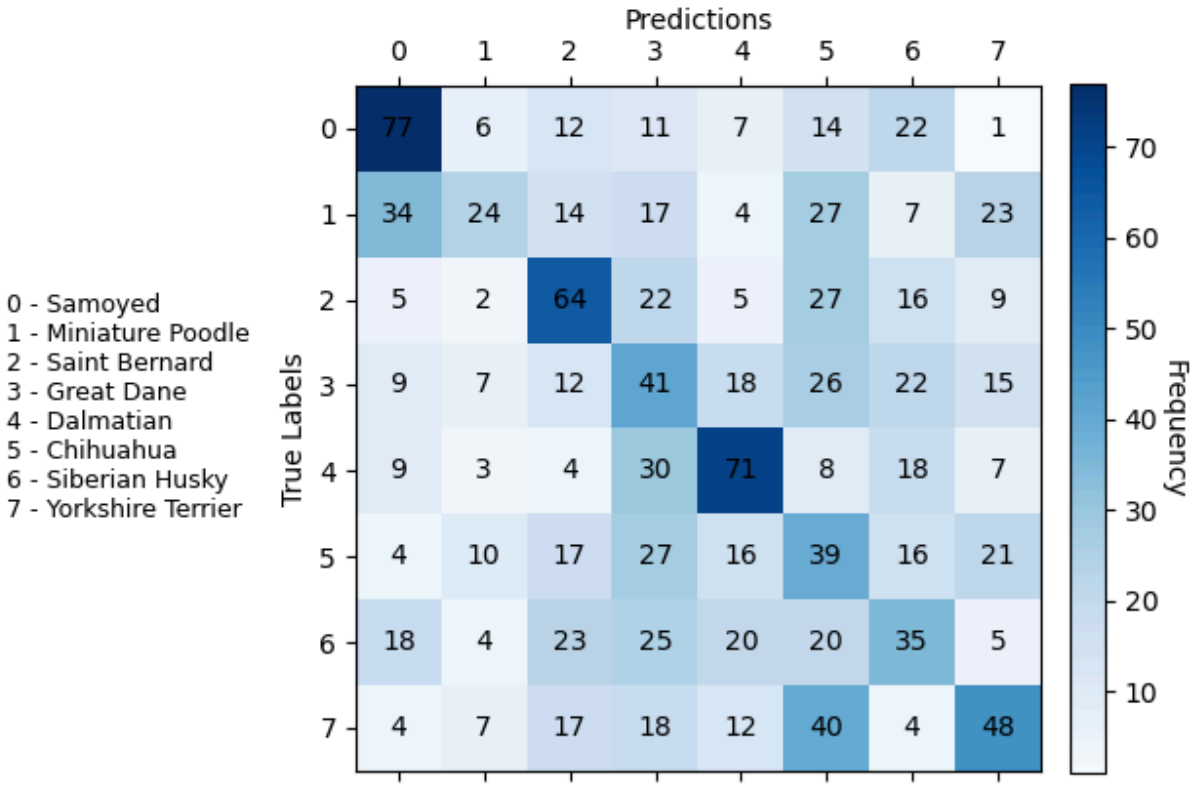
Epoch 11 corresponds to the model with the lowest validation loss.

```
(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason
_wu$ python train_source.py
loading train...
loading val...
loading test...
Number of float-valued parameters: 39952
Loading source...
Which epoch to load from? Choose in range [0, 2]. Enter 0 to train f
rom scratch.
>> 0
Checkpoint not loaded
Checkpoint successfully removed
Epoch 0
        Validation Accuracy:0.1325
        Validation Loss:2.1531
        Validation AUROC:0.5084
        Train Accuracy:0.144
        Train Loss:2.148
        Train AUROC:0.5143
Epoch 1
        Validation Accuracy:0.1667
        Validation Loss:2.0443
        Validation AUROC:0.5963
        Train Accuracy:0.194
        Train Loss:2.03
        Train AUROC:0.6248
Epoch 2
        Validation Accuracy:0.2275
        Validation Loss:1.9745
        Validation AUROC:0.6643
        Train Accuracy:0.2415
        Train Loss:1.947
        Train AUROC:0.6912
Epoch 3
        Validation Accuracy:0.2642
        Validation Loss:1.9265
        Validation AUROC:0.6834
        Train Accuracy:0.302
        Train Loss:1.8707
        Train AUROC:0.7203
Epoch 4
        Validation Accuracy:0.3008
        Validation Loss:1.883
        Validation AUROC:0.7042
        Train Accuracy:0.3195
        Train Loss:1.7997
        Train AUROC:0.7523
Epoch 5
        Validation Accuracy:0.3033
        Validation Loss:1.8497
        Validation AUROC:0.7134
        Train Accuracy:0.3585
        Train Loss:1.7382
        Train AUROC:0.7724
Epoch 6
        Validation Accuracy:0.3075
        Validation Loss:1.8414
        Validation AUROC:0.7235
        Train Accuracy:0.3505
        Train Loss:1.7102
        Train AUROC:0.7919
Epoch 7
        Validation Accuracy:0.2933
        Validation Loss:1.831
        Validation AUROC:0.718
        Train Accuracy:0.4085
        Train Loss:1.6403
```

```
Epoch 8
        Validation Accuracy:0.3192
        Validation Loss:1.8111
        Validation AUROC:0.7224
        Train Accuracy:0.4185
        Train Loss:1.5771
        Train AUROC:0.8141
Epoch 9
        Validation Accuracy:0.3117
        Validation Loss:1.8214
        Validation AUROC:0.7287
        Train Accuracy:0.43
        Train Loss:1.5566
        Train AUROC:0.8268
Epoch 10
        Validation Accuracy:0.3125
        Validation Loss:1.8133
        Validation AUROC:0.73
        Train Accuracy:0.4565
        Train Loss:1.5082
        Train AUROC:0.8386
Epoch 11
        Validation Accuracy:0.3325
        Validation Loss:1.7831
        Validation AUROC:0.7373
        Train Accuracy:0.483
        Train Loss:1.4339
        Train AUROC:0.854
Epoch 12
        Validation Accuracy:0.3233
        Validation Loss:1.8486
        Validation AUROC:0.7306
        Train Accuracy:0.473
        Train Loss:1.4112
        Train AUROC:0.8637
Epoch 13
        Validation Accuracy:0.32
        Validation Loss:1.872
        Validation AUROC:0.725
        Train Accuracy:0.5095
        Train Loss:1.3652
        Train AUROC:0.8723
Epoch 14
        Validation Accuracy:0.3283
        Validation Loss:1.8206
        Validation AUROC:0.7346
        Train Accuracy:0.5565
        Train Loss:1.2577
        Train AUROC:0.8936
Epoch 15
        Validation Accuracy:0.3233
        Validation Loss:1.8203
        Validation AUROC:0.7342
        Train Accuracy:0.5995
        Train Loss:1.191
        Train AUROC:0.9087
Epoch 16
        Validation Accuracy:0.3092
        Validation Loss:1.9325
        Validation AUROC:0.7272
        Train Accuracy:0.574
        Train Loss:1.2004
        Train AUROC:0.9114
Epoch 17
        Validation Accuracy:0.3242
        Validation Loss:1.9675
        Validation AUROC:0.7234
```

```
        Train Accuracy:0.574
        Train Loss:1.2004
        Train AUROC:0.9114
Epoch 17
        Validation Accuracy:0.3242
        Validation Loss:1.9675
        Validation AUROC:0.7234
        Train Accuracy:0.6155
        Train Loss:1.1295
        Train AUROC:0.9237
Epoch 18
        Validation Accuracy:0.3258
        Validation Loss:1.9133
        Validation AUROC:0.7253
        Train Accuracy:0.6675
        Train Loss:1.0131
        Train AUROC:0.9379
Epoch 19
        Validation Accuracy:0.3075
        Validation Loss:1.9432
        Validation AUROC:0.722
        Train Accuracy:0.702
        Train Loss:0.9299
        Train AUROC:0.9503
Epoch 20
        Validation Accuracy:0.2975
        Validation Loss:2.1283
        Validation AUROC:0.7174
        Train Accuracy:0.695
        Train Loss:0.9516
        Train AUROC:0.9505
Epoch 21
        Validation Accuracy:0.3108
        Validation Loss:2.1175
        Validation AUROC:0.7209
        Train Accuracy:0.7275
        Train Loss:0.8737
        Train AUROC:0.9602
Finished Training
```

3(d)

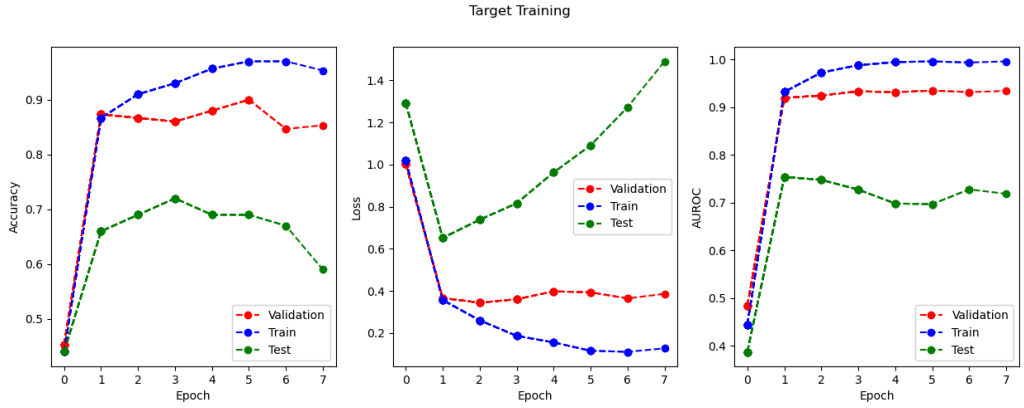


Dalmatian(4), Saint Bernard(2), and Samoyed(0) are the most accurate.
Siberian Husky(7), Chihuahua(6), and Great Dane(3) are the least accurate.
Appearance similarity: Certain dog breeds have great similarity in appearance (e.g., coat color, body type, hair type, etc.), and models may be difficult to distinguish. For breeds such as Chihuahua and Samoyed, models can be confused due to their small size and white hair.
Data Imbalance: If there is insufficient sample size for certain breeds in the training data set, the model may perform poorly in these categories. Chihuahua has a higher number of misclassifications, possibly due to a smaller sample of the breed or a less distinct feature.
Feature complexity: Some dog breeds may have unique features that make it easier for models to learn. For example, Dalmatian spots are very distinct visual features, and models may perform better at recognizing them.
3(e)
3(f)

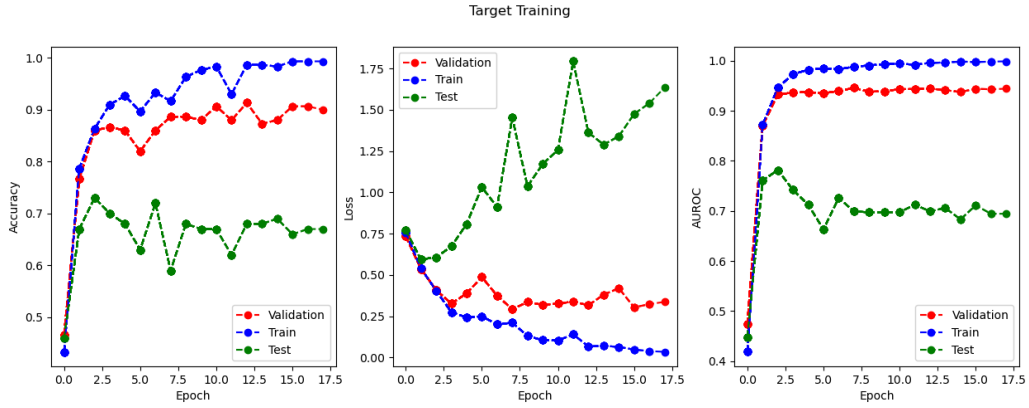
	AUROC		
	TRAIN	**VAL**	**TEST**
Freeze all CONV layers (Fine-tune FC layer)	0.8993	0.896	0.8204
Freeze first two CONV layers (Fine-tune last CONV and FC layers)	0.986	0.9081	0.768

Freeze first CONV layer (Fine-tune last 2 conv. and fc layers)	0.9964	0.9232	0.7196
Freeze no layers (Fine-tune all layers)	0.9872	0.9204	0.7292
No Pretraining or Transfer Learning (Section 2 performance)	0.9963	0.9406	0.6836

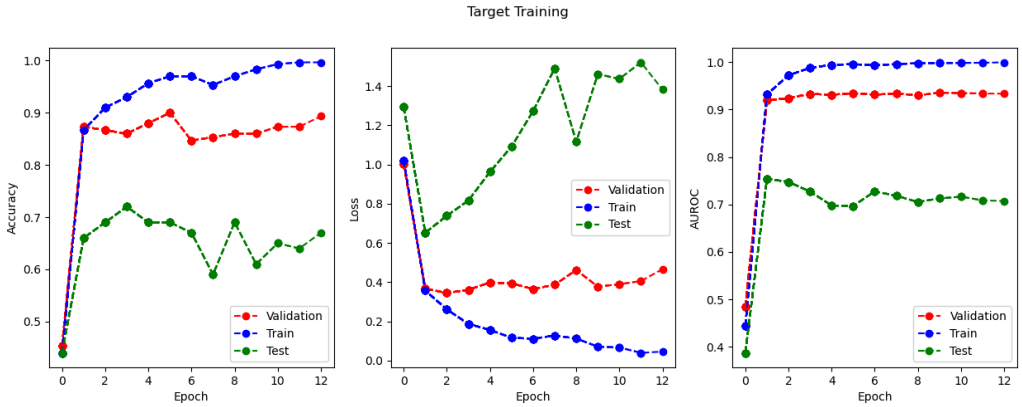
```
Epoch 2
        Validation Accuracy:0.8667
        Validation Loss:0.3447
        Validation AUROC:0.9244
        Train Accuracy:0.91
        Train Loss:0.2608
        Train AUROC:0.9724
        Test Accuracy:0.69
        Test Loss:0.7385
        Test AUROC:0.7476
```




```
Epoch 3
        Validation Accuracy:0.8667
        Validation Loss:0.3267
        Validation AUROC:0.9364
        Train Accuracy:0.91
        Train Loss:0.2744
        Train AUROC:0.9732
        Test Accuracy:0.7
        Test Loss:0.673
        Test AUROC:0.742
```



```
Epoch 2
        Validation Accuracy:0.8667
        Validation Loss:0.3447
        Validation AUROC:0.9244
        Train Accuracy:0.91
        Train Loss:0.2608
        Train AUROC:0.9724
        Test Accuracy:0.69
        Test Loss:0.7385
        Test AUROC:0.7476
```



4(a)

In the ViT architecture, an additional learnable [class] embed is a special marker used to represent the final output of an image classification task. In ViT, this [class] embed is fed into the Transformer Encoder and takes part in the attention calculation along with the other embeddings of the image patch. After being processed by the Transformer Encoder, the final [class] embed contains comprehensive information about the entire image and is used for the final classification task.
In the absence of this learnable [class] embed, the embed vectors output by the Transformer Encoder can be aggregated in other ways for classification by using average pooling or maximum pooling of all patches to get a global feature vector and use it for the classification task.

Advantages of using [class] embedding:
Focus on classification tasks: The learnable [class] embedded in the attention mechanism gradually accumulates information about the entire image and is specifically used for classification tasks, avoiding the loss of averaging information from different patches.
More flexible feature aggregation: Compared to average pooling, Transformer Encoder can adaptively provide richer context information for [class] embeddings through the attention mechanism to improve classification accuracy.
Simplified post-processing: The classification output is obtained directly through [class] embedding, and no additional aggregation operations are required for other patch embedding.

4(b)

Positional Embedding is used to provide positional information for the input patch, enabling the Transformer Encoder to understand the relative positions of different patches. In the original NLP Transformer architecture, the input was sequential data, and the order was important, so positional

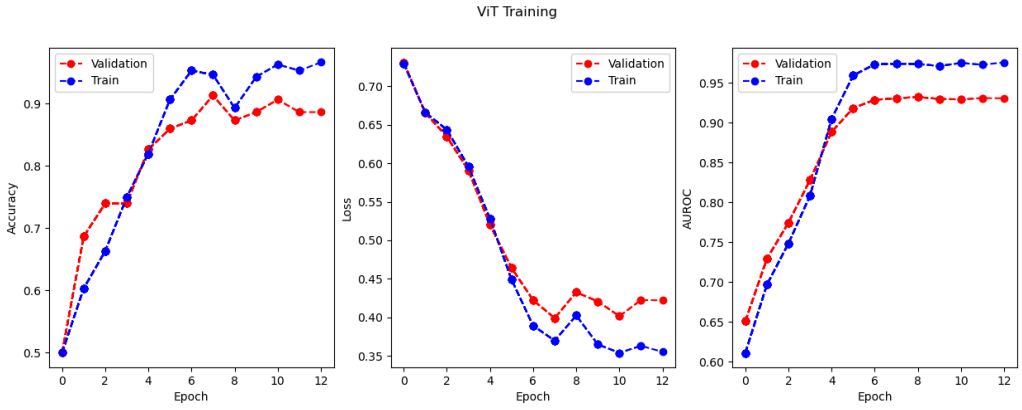
embedding was added. In ViT, since the image is a two-dimensional structure, the spatial position of the patch is equally critical.
If positional embedding is omitted, the model loses the location information of the patches and cannot distinguish which patches are in which position in the image. This can have the following effects:
Inability to recognize spatial structure: The spatial information of the image (such as shape, edge, position of the object, etc.) is very important for recognizing the object. Without positional embedding, the model cannot effectively distinguish the relative positions of different patches, resulting in degraded classification performance.
Model confusion: Without location information, the model may assume that patches at different locations are interchangeable, which interferes with the classification accuracy of the model.
Therefore, location embedding is essential for ViT models to correctly capture spatial relationships of images in image classification tasks.
4(c)
4(d)
4(e)
i) cls token is a learnable vector with an embedded dimension of d=16. Therefore, the number of parameters for the cls token is: 16.
ii) We split the image into 16 patches, each with dimensions of 3×64×64. After each patch is flattened, it is projected through a linear layer into the embedded dimension d=16. The number of parameters of the Patch Linear Projection is (3x64x64)x16=196608. Where 16 is the weight parameter of the linear layer, so 16 is also the bias parameter. Also, each patch has a positional embedding vector with dimension d = 16. We have num_patches = 16 patches. Therefore, the number of parameters of the Position Embedding is: 16×16=256. In total, we would have 196608+16+256=196880 parameters.
iii) MLP Head is a single-layer fully connected layer that maps the embedded dimension d = 16 to the class number C. Let's say we have C categories. The number of parameters for the MLP Head is: 16×C+C. In this case, for C=8, we have 16×8+8=136. (I actually don't know what this question's background exactly is, if it is still based on 1 that C=2, we have 16×2+2=34.)
iv)
Each attentional head has a linear projection of Query, Key, and Value that maps the embedded dimension d to query_size. Here query_size = d/num_heads = 16/2 = 8. For each attention head, we have: (d×8+8)×3=(16×8+8)×3=408 There are a total of 2 Attention heads, so the number of arguments for Multi-Head attention is: 408x2=916.
The output of multi-head attention is spliced and mapped back to the d dimension through a linear layer. The number of parameters of this linear layer is: d×d+d=16×16+16=272.
LayerNorm usually has two parameters: weight and bias, each with d parameters. There are two Layernorms in a Transformer Block, so the total number of parameters is: 2×d×2=2×16×2=64.
The MLP consists of two fully connected layers. The first layer maps d to 4d and the second layer maps 4d back to d. The number of parameters in the first layer is: d×4d×4d=16×64+64=1088. The number of layer 2 parameters is: 4d×d×d=64×16×+6=1040. Therefore, the total number of parameters for MLP is: 1088+1040=2128.
In total, we'll have 916+272+64+2128=3380.
v) The entire ViT model contains 2 Transformer blocks, so the total number of parameters in Transformer blocks is: 2x3380=6760.
4(f)
4(g)

The train/validation AUROC performance of the model from the epoch demonstrating the lowest validation loss is epoch 7.

	Training	Validation
AUROC	0.9742	0.9308

```
(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason_wu$ python train_vit.py
loading train...
loading val...
loading test...
Number of float-valued parameters: 7394
Loading ViT...
No saved model parameters found
Epoch 0
        Validation Accuracy:0.5
        Validation Loss:0.7311
        Validation AUROC:0.6508
        Train Accuracy:0.5
        Train Loss:0.7298
        Train AUROC:0.6111
Epoch 1
        Validation Accuracy:0.6867
        Validation Loss:0.6656
        Validation AUROC:0.7292
        Train Accuracy:0.6033
        Train Loss:0.6658
        Train AUROC:0.6966
Epoch 2
        Validation Accuracy:0.74
        Validation Loss:0.6346
        Validation AUROC:0.7746
        Train Accuracy:0.6633
        Train Loss:0.6436
        Train AUROC:0.7486
Epoch 3
        Validation Accuracy:0.74
        Validation Loss:0.5909
        Validation AUROC:0.8283
        Train Accuracy:0.75
        Train Loss:0.5957
        Train AUROC:0.8086
Epoch 4
        Validation Accuracy:0.8267
        Validation Loss:0.5201
        Validation AUROC:0.8885
        Train Accuracy:0.82
        Train Loss:0.5282
        Train AUROC:0.905
Epoch 5
        Validation Accuracy:0.86
        Validation Loss:0.4642
        Validation AUROC:0.9184
        Train Accuracy:0.9067
        Train Loss:0.4489
        Train AUROC:0.9595
Epoch 6
        Validation Accuracy:0.8733
        Validation Loss:0.4224
        Validation AUROC:0.9289
        Train Accuracy:0.9533
        Train Loss:0.3891
        Train AUROC:0.9734
Epoch 7
        Validation Accuracy:0.9133
        Validation Loss:0.399
        Validation AUROC:0.9308
        Train Accuracy:0.9467
        Train Loss:0.3697
        Train AUROC:0.9742
Epoch 8
        Validation Accuracy:0.8733
        Validation Loss:0.4326
```

```
        Validation AUROC:0.9323
        Train Accuracy:0.8933
        Train Loss:0.4024
        Train AUROC:0.9738
Epoch 9
        Validation Accuracy:0.8867
        Validation Loss:0.4206
        Validation AUROC:0.93
        Train Accuracy:0.9433
        Train Loss:0.3648
        Train AUROC:0.9713
Epoch 10
        Validation Accuracy:0.9067
        Validation Loss:0.4015
        Validation AUROC:0.9293
        Train Accuracy:0.9633
        Train Loss:0.3539
        Train AUROC:0.9751
Epoch 11
        Validation Accuracy:0.8867
        Validation Loss:0.4225
        Validation AUROC:0.931
        Train Accuracy:0.9533
        Train Loss:0.363
        Train AUROC:0.9728
Epoch 12
        Validation Accuracy:0.8867
        Validation Loss:0.4221
        Validation AUROC:0.9307
        Train Accuracy:0.9667
        Train Loss:0.3552
        Train AUROC:0.9759
Finished Training
```



4(h) & 4(i)

```
[(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason]
_wu$ python test_vit.py
loading train...
loading val...
loading test...
Loading Vit...
Which epoch to load from? Choose in range [0, 12]. Enter 0 to train
from scratch.
>> 7
Loading from checkpoint ./checkpoints/vit/epoch=7.checkpoint.pth.tar
?
=> Successfully restored checkpoint (trained for 7 epochs)
Epoch 7
        Validation Accuracy:0.9133
        Validation Loss:0.399
        Validation AUROC:0.9308
        Train Accuracy:0.9467
        Train Loss:0.3643
        Train AUROC:0.9742
        Test Accuracy:0.59
        Test Loss:0.7232
        Test AUROC:0.5616
```

AUROC performance:

	Training	Validation	Testing
ViT	0.9742	0.9308	0.5616
CNN	0.9963	0.9406	0.6836

Training (training set) : ViT: 0.9742 CNN: 0.9963
Analysis: On the training set, CNN performed slightly better than ViT. The higher training accuracy of CNN may indicate that it is stronger at fitting training data, or it may mean that it is more complex than ViT and suitable for working with this particular data set.
Validation (validation set) : ViT: 0.9308 CNN: 0.9406
Analysis: On the validation set, CNN performs slightly better than ViT, but not by much. This suggests that CNN may have better adaptability to the characteristics of this data set, especially the generalization performance on the validation set is slightly higher than that of ViT.
Testing (test set) : ViT: 0.5616 CNN: 0.6836
Analysis: On the test set, CNN performs significantly better than ViT, and the test accuracy of CNN is about 12% higher than that of ViT. This shows that the generalization performance on the test set is stronger on CNN than on ViT.
This gap may be related to the following reasons:
Training data size and data requirements of Vits: ViT models typically require larger data sets to adequately train because it does not have the convolutional structure of CNN and lacks the ability to extract local features on small data sets.
Advantages of CNN for visual tasks: The CNN model is naturally suitable for processing image data and can efficiently extract spatial local features through convolution kernel. However, ViT mainly relies on self-attention mechanisms to capture long-distance dependencies and may not be fully utilized on smaller data sets.
Feature extraction mechanism of ViT: ViT's self-attention mechanism relies on global information when the amount of data is small or the local features of the image are very small.

5

Here is the deep convolutional neural network I made of classifying Collies and Golden Retrievers. Below is a detailed discussion of the different design decisions I made:

1. Model Architecture

I used a deep convolutional neural network (CNN) with residual connections, with an architecture inspired by ResNet.

Initial Convolutional Layers: The first convolutional layer maps the input RGB image to 64 feature channels with batch normalization and ReLU activation.

Residual Blocks: 3 layers of residual modules are used in the network. Each module contains 2 residual blocks, with a progressively increasing number of channels (64 → 128 → 256 → 512), while downsampling is achieved by stride=2. The residual blocks mitigate the gradient vanishing problem in deep networks and ensure effective learning of deep features.

Global average pooling: after the last residual block, the output feature map is reduced to 8x8 using adaptive_avg_pool2d. This global pooling approach allows for better adaptation to the input image size.

Fully Connected Layers: Two fully connected layers are used with Dropout for regularization, and the final output is the prediction of the 2 categories.

2. Regularization

In order to prevent the model from overfitting, I use various regularization methods:

Dropout: A 50% dropout is added to the first fully connected layer, which helps reduce overfitting and improves the model's performance on the validation set.

Batch Normalization: Batch normalization is added after each convolutional layer. Batch normalization can speed up the training process, reduce the problem of gradient disappearance, and play a certain role in regularization.

3. Data Enhancement

I apply a variety of data enhancement techniques aimed at increasing the generalization ability of the model, including:

Random cropping and scaling: adding image segments of different sizes and positions through RandomResizedCrop to simulate different imaging conditions.

Horizontal Flip: Randomly flips the image horizontally to prevent the model from being biased against image features in a particular orientation.

Color Perturbation: Adjusts the brightness, contrast, saturation, and hue of an image using ColorJitter to increase the robustness of the model to changes in lighting.

Random Rotation: Randomly rotate the image to help the model better adapt to different orientations of the target.

4. Hyperparameter Selection

During model training, I adjusted some key hyperparameters to optimize the model performance:

Learning rate: the initial learning rate is set to 1e-4. Through experiments, I found that this learning rate can converge in a reasonable time while avoiding large oscillations.

Learning Rate Scheduler: The StepLR scheduler is used, which reduces the learning rate by 50% every 10 epochs to help the model continue to improve its accuracy in the later stages of convergence.

Batch Size: 32 was chosen as the batch size. Among the different sizes I tried, 32 balanced the memory usage and model training speed.

5. Model Evaluation Criteria

In the model evaluation process, I use the following metrics to judge the strengths and weaknesses of the model:

Accuracy: This is the main evaluation metric of the model, which directly reflects the classification accuracy of the model on the test set.

Loss: I use the loss value of the validation set to evaluate the convergence of the model and use it as a criterion for early stopping.

AUROC (Area Under ROC Curve): AUROC is a better measure of model performance on unbalanced datasets.

With these metrics, I was able to determine which model had better generalization performance.

6. Early Stopping and Patience settings

In order to prevent the model from overfitting during training, I introduced the Early Stopping mechanism:

Patience: Allow up to 5 epochs (patience = 5) before the validation loss decreases, and stop training if this number is exceeded.

Implementation: By calculating the validation loss after each epoch and comparing it with the previous epoch, and increasing the patience counter if there is no significant improvement, triggering an Early Stopping when patience reaches a threshold.

7. Future Improvement Directions

More residual blocks: the network can be further deepened by increasing the number of residual block layers.

Richer data enhancement: Try other data enhancement methods, such as Cutout or Mixup, to further improve the robustness of the model.

Optimize the regularization strategy: In future experiments, Dropout can be introduced in different layers, and even DropBlock can be used in the convolutional layer.

Extra Output:

```
Which epoch to load from? Choose in range [0, 12]. Enter 0 to train
from scratch.
>> 7
Loading from checkpoint ./checkpoints/target/epoch=7.checkpoint.pth.
tar?
=> Successfully restored checkpoint (trained for 7 epochs)
Epoch 7
        Validation Accuracy:0.8867
        Validation Loss:0.3308
        Validation AUROC:0.9406
        Train Accuracy:0.9667
        Train Loss:0.1044
        Train AUROC:0.9963
Epoch 8
        Validation Accuracy:0.86
        Validation Loss:0.4406
        Validation AUROC:0.9335
        Train Accuracy:0.9267
        Train Loss:0.1699
        Train AUROC:0.9896
Epoch 9
        Validation Accuracy:0.8667
        Validation Loss:0.343
        Validation AUROC:0.9383
        Train Accuracy:0.9533
        Train Loss:0.1172
        Train AUROC:0.9961
Epoch 10
        Validation Accuracy:0.8667
        Validation Loss:0.4436
        Validation AUROC:0.9346
        Train Accuracy:0.9767
        Train Loss:0.0873
        Train AUROC:0.9976
Epoch 11
        Validation Accuracy:0.88
        Validation Loss:0.4407
        Validation AUROC:0.936
        Train Accuracy:0.9867
        Train Loss:0.0654
        Train AUROC:0.9975
Epoch 12
        Validation Accuracy:0.88
        Validation Loss:0.3815
        Validation AUROC:0.9387
        Train Accuracy:0.99
        Train Loss:0.0475
        Train AUROC:0.9998
Epoch 13
        Validation Accuracy:0.8667
        Validation Loss:0.4437
        Validation AUROC:0.9404
        Train Accuracy:0.9733
        Train Loss:0.0904
        Train AUROC:0.9996
Epoch 14
        Validation Accuracy:0.8733
        Validation Loss:0.456
        Validation AUROC:0.941
        Train Accuracy:0.9933
        Train Loss:0.0425
        Train AUROC:1.0
Finished Training
```

```
[(445_p2) mason-wudeMacBook-Probuzhidaogaigaishamingzi:project2 mason]
_wu$ python test_cnn.py
loading train...
loading val...
loading test...
Loading cnn...
Which epoch to load from? Choose in range [0, 12]. Enter 0 to train
from scratch.
>> 11
Loading from checkpoint ./checkpoints/target/epoch=11.checkpoint.pth
.tar?
=> Successfully restored checkpoint (trained for 11 epochs)
Epoch 11
        Validation Accuracy:0.8733
        Validation Loss:0.4511
        Validation AUROC:0.936
        Train Accuracy:0.9933
        Train Loss:0.0438
        Train AUROC:0.9996
        Test Accuracy:0.64
        Test Loss:1.4767
        Test AUROC:0.6728
```

	AUROC		
	TRAIN	**VAL**	**TEST**
Freeze all CONV layers (Fine-tune FC layer)	0.9724	0.9244	0.7476
Freeze first two CONV layers (Fine-tune last CONV and FC layers)	0.9732	0.9367	0.742
Freeze first CONV layer (Fine-tune last 2 conv. and fc layers)	0.9724	0.9244	0.7476
Freeze no layers (Fine-tune all layers)			
No Pretraining or Transfer Learning (Section 2 performance)	0.9963	0.9406	0.6836