https://github.com/BeckSchemenauer/Breast-Histopathology-Image-Classification

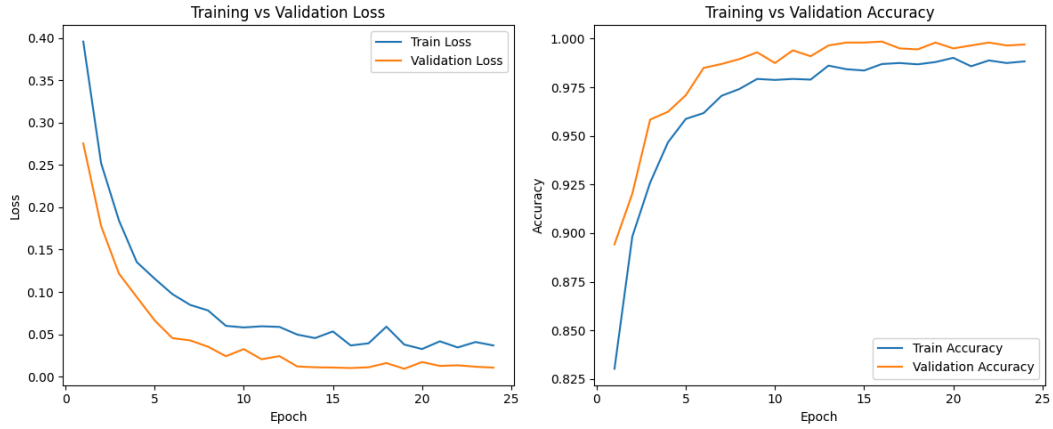
FNN Results

The FNN achieved 99.55% test accuracy, which was very suspicious, especially for image data. I believed that this likely was a result of data leakage since patient IDs were not considered. In other words, images from the same patient may have been seen during training and testing. I retrained using GroupShuffleSplit, and accuracy dropped to ~93%, suggesting the original result was a result of data leaking. There was overfitting due to non-independent data, therefore performance was artificially inflated.

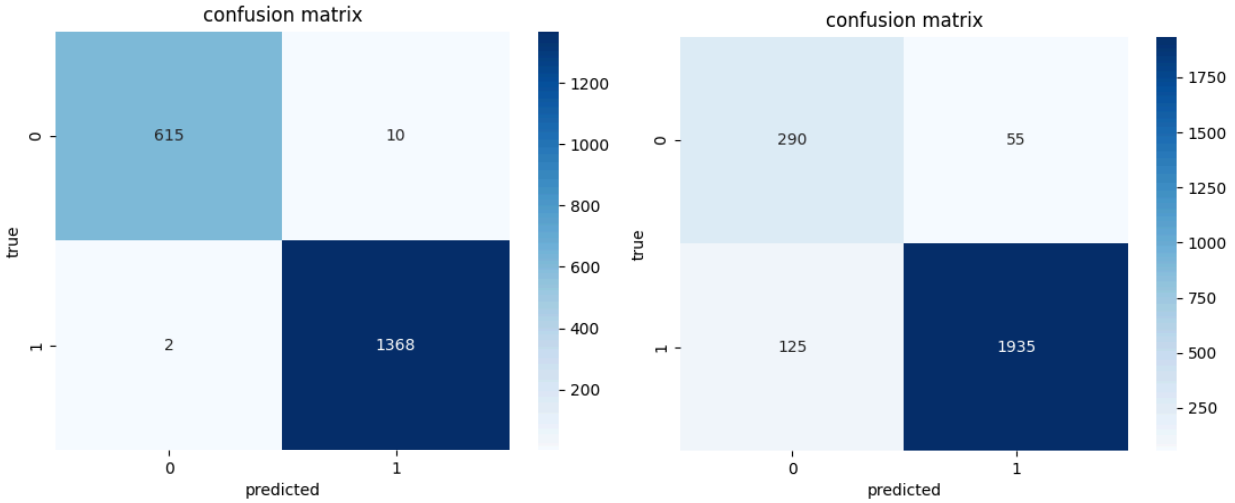
```
classification report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       625
           1       0.99      1.00      1.00      1370

    accuracy                           1.00      1995
   macro avg       1.00      0.99      0.99      1995
weighted avg       1.00      1.00      1.00      1995
```



Plots for the 99.55% accuracy model, which do not indicate overfitting, but do show evidence of data leaking, as the lines follow each other closer.



The confusion matrix on the left is the initial run, and the one on the right is after using GroupShuffleSplit

---

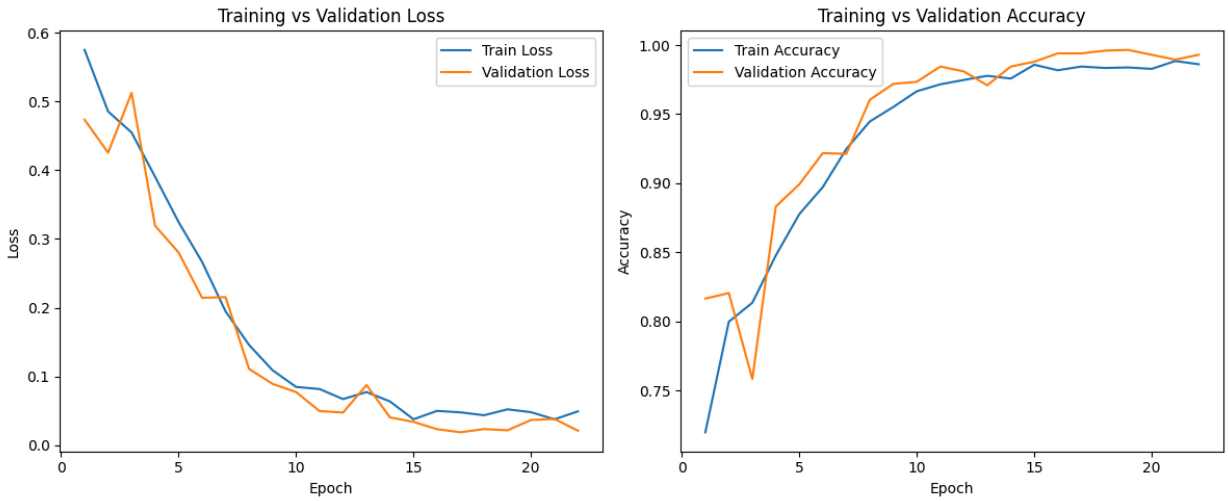
CNN Without Augmentation

The test accuracy was 99.25%, which is slightly lower than the FNN. It used the same split as the FNN (no GroupShuffleSplit), so it likely suffered from the same data leakage issue. Additionally, the training and validation curves showed smooth convergence, but this may not reflect true generalization. In short, there was high performance, it was not reliable due to potential overlap between patients in training and test sets.

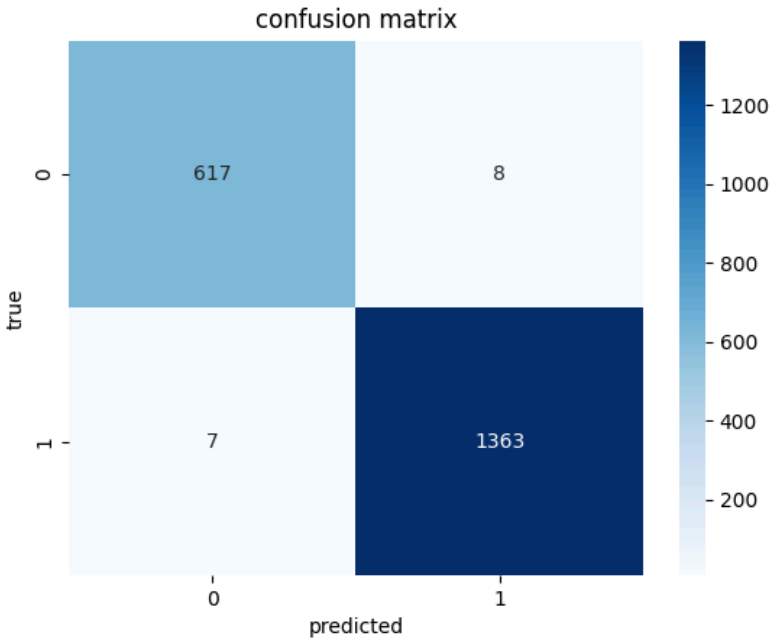
```
classification report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       625
           1       0.99      0.99      0.99      1370

    accuracy                           0.99      1995
   macro avg       0.99      0.99      0.99      1995
weighted avg       0.99      0.99      0.99      1995
```



Once again the loss and accuracy plots show evidence of data leakage as the



CNN With GroupShuffleSplit + Augmentation

This model got the best accuracy (ignoring the suspicious accuracies from before), as it used the most robust dataset which involved different magnification sizes and data augmentation techniques. All of these models took significant time to train on my computer, so when it came to hyperparameter tuning on this one, I could not get away with a gridsearch, and had to instead make a few careful changes to attempt to improve accuracy. Below is a table showing my different iterations and corresponding results.

Model #	Batch Size	Learning Rate	Accuracy	F1-Score (Benign)	F1-Score (Malignant)
1	2	.001	93.97%	0.82	0.96
2	8	.0001	92.72%	0.77	0.96
3	64	.001	94.59%	0.82	0.97
4	128	.0005	90.64%	0.75	0.94
5*	64	.001	95.01%	0.84	0.97

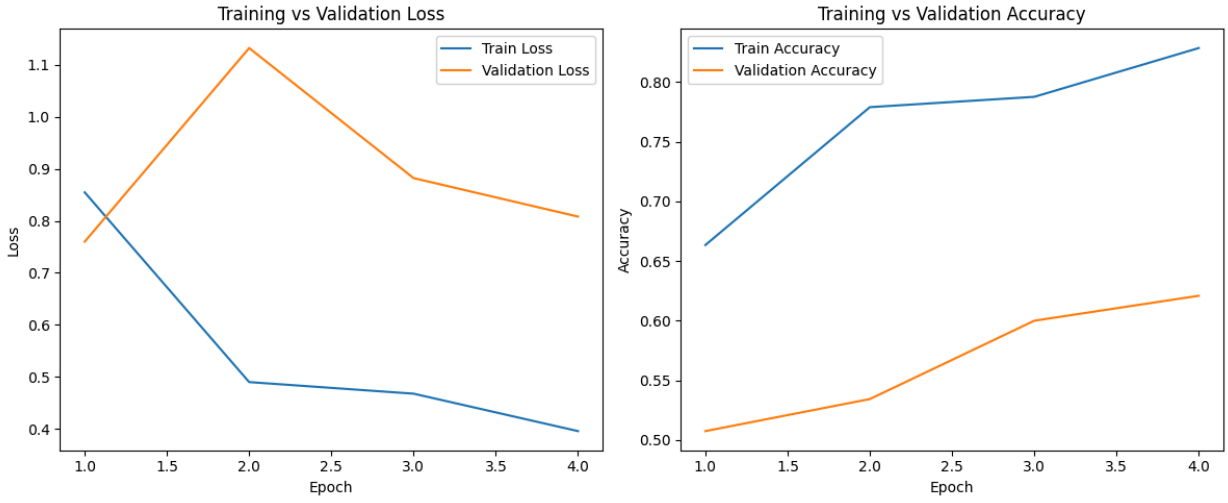
5* (changes for the best model)

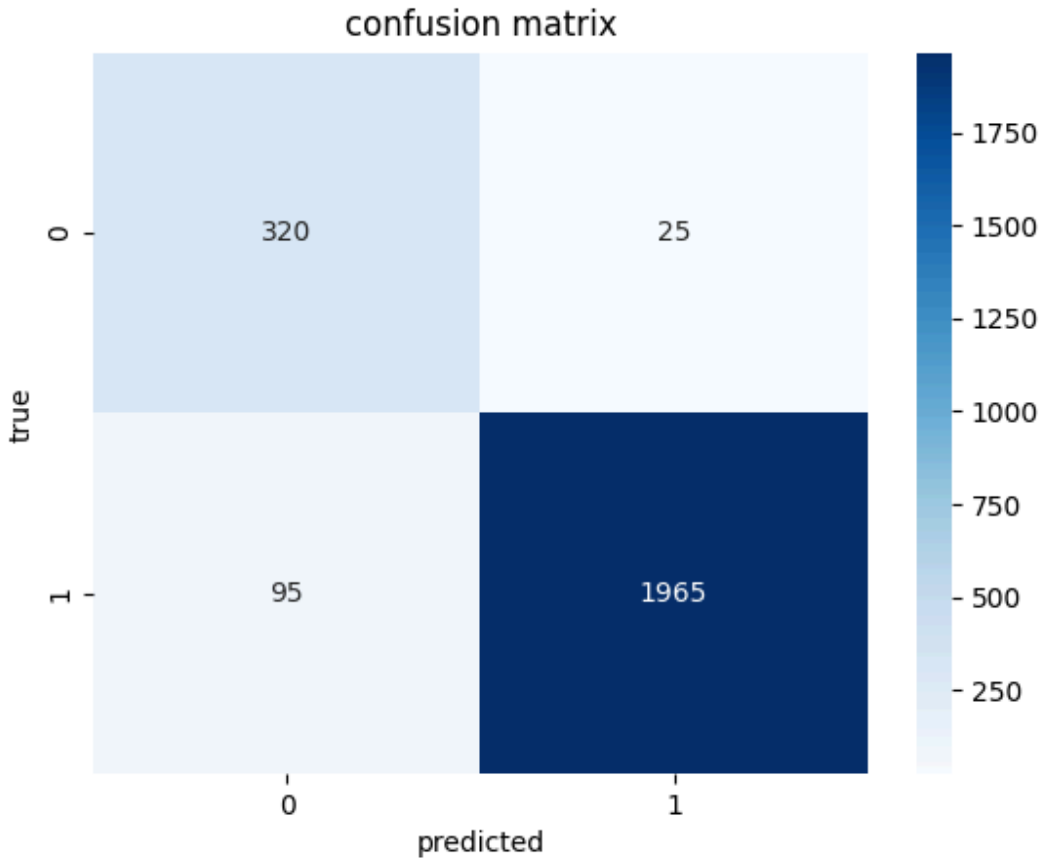
- Added a learning rate scheduler that reduces the learning rate by half if validation loss does not improve for 2 consecutive epochs
- Reduced dropout from 0.5 to 0.4 after the dense layer with 1024 neurons to slightly lower regularization and allow the model to retain more information

```
classification report:
              precision    recall  f1-score   support

           0       0.77      0.93      0.84       345
           1       0.99      0.95      0.97      2060

    accuracy                           0.95      2405
   macro avg       0.88      0.94      0.91      2405
weighted avg       0.96      0.95      0.95      2405
```

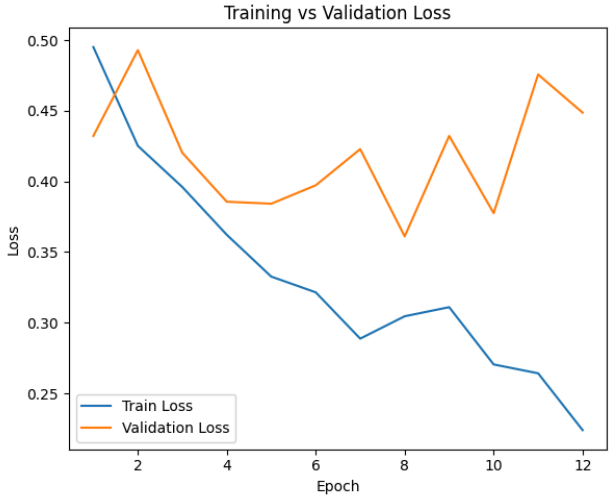


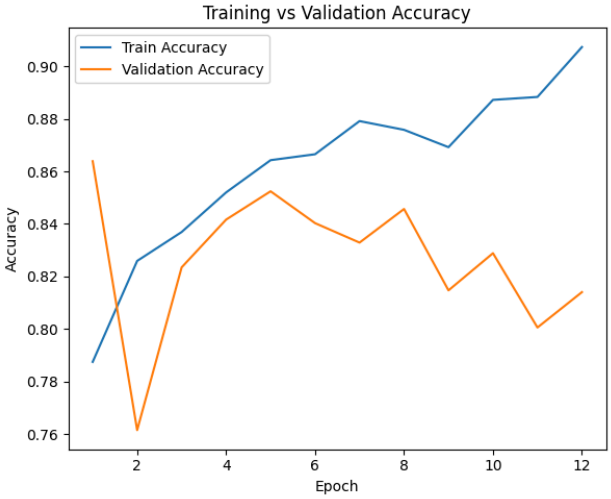


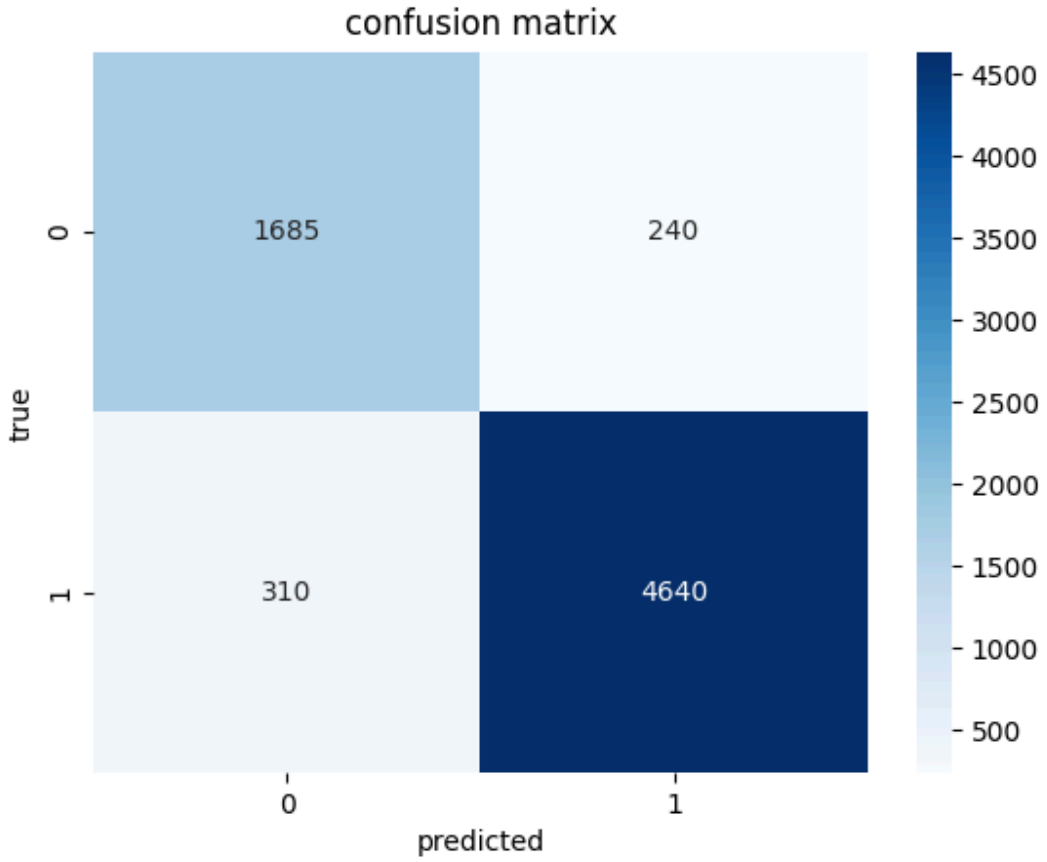
I discovered an error in that my second CNN was only using the 40x images, additionally the GroupShuffleSplit gave very inconsistent distributions among the training, validation and testing groups.

train class distribution: {0: 1955, 1: 3940}
validation class distribution: {0: 825, 1: 850}
test class distribution: {0: 345, 1: 2060}

Upon fixing the magnification error, and writing my own split method, I achieved 92.00% accuracy. The corresponding plots are seen below:


Training vs Validation Loss
Loss
Epoch
Train Loss
Validation Loss


Training vs Validation Accuracy
Accuracy
Epoch
Train Accuracy
Validation Accuracy


confusion matrix
1685
240
310
4640
true
predicted
0
1
4500
4000
3500
3000
2500
2000
1500
1000
500

cuda
loaded image paths and labels
Train set: 25250 samples — class distribution: {1: 17075, 0: 8175}
Validation set: 7420 samples — class distribution: {1: 5120, 0: 2300}
Test set: 6875 samples — class distribution: {1: 4950, 0: 1925}
split data
epoch 1, val loss: 0.4322, val acc: 0.8639
epoch 2, val loss: 0.4929, val acc: 0.7615
epoch 3, val loss: 0.4203, val acc: 0.8235
epoch 4, val loss: 0.3856, val acc: 0.8416
epoch 5, val loss: 0.3842, val acc: 0.8524
epoch 6, val loss: 0.3973, val acc: 0.8403
epoch 7, val loss: 0.4229, val acc: 0.8329
epoch 8, val loss: 0.3611, val acc: 0.8457
epoch 9, val loss: 0.4322, val acc: 0.8147
epoch 10, val loss: 0.3776, val acc: 0.8288
epoch 11, val loss: 0.4757, val acc: 0.8005
epoch 12, val loss: 0.4487, val acc: 0.8140
early stopping triggered.

```
classification report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86      1925
           1       0.95      0.94      0.94      4950

    accuracy                           0.92      6875
   macro avg       0.90      0.91      0.90      6875
weighted avg       0.92      0.92      0.92      6875

final test accuracy: 0.9200
```

Overall

The FNN appeared most accurate at first glance but suffered most from what appeared to be data leakage. This is my conclusion at the time, but I think I will need to investigate more to confidently find the true root of the problem. The second CNN could have potential but suffered from the same issues as seen in the FNN. The final CNN performed much better, especially taking into account the use of group shuffle split and a more fair training process.