Convolutional Neural Network

For Face Classification

Garrett Gibo **Siqi Huang** **Zekria Sidiqi**

Abstract

Deep learning has many applications with facial classification being one of the more prominent in recent times. Building and training a deep CNN from scratch can be impractical; however, discovering how to manipulate network architecture to get tangible results was a key point of this problem. Starting with a baseline CNN that has only a few convolutions, we built off this network by adding additional layers and modifying the convolutions themselves. In addition, applying different activation functions were experimented on in order to increase performance.

1. Introduction

The primary goal of was to build a deep CNN from scratch to classify the identities of various celebrities. There were 200 identities in the dataset, thus 200 classes to classify from. The initial baseline model that we tested against was created with randomized weight; however, we used Xavier weight initialization instead. The main idea of Xavier weight initialization is that the weights are chosen from a distribution with zero mean, and a specific variance. This ensures that the neurons' activations will not become saturated or die too quickly. In addition, we implemented batch normalization in the network. Batch normalization reduces the covariance shift of the hidden units by normalizing the output of the previous activations.

In our project, we use four different models. For the baseline model we had a test accuracy of 52.09% and for the transfer learning model we had an accuracy of 53.35%. We obtain the highest test accuracy of 64.31% from architecture 1. We obtain test accuracy of 62.32% from architecture 2, the ZekeNet.

2. Related Work

While experimenting with network architecture, we took cues from other established networks such as AlexNet (1) and ResNet (2). Specifically, we were inspired by the structure of individual convolutions and their respective change input and output channels.

3. Methods

Baseline: In order to understand our experiments, it is important to have a general understanding of the original baseline architecture. The baseline network was constructed with a series of Conv2d, ReLU, and BatchNorm2d layers. The activation function that we used for the last layer of the network was a softmax, and the loss criterion that we optimized was Cross Entropy loss.

Experimentation:

Architecture 1 was inspired by Alexnet and our discussion. We added a max pooling layer after the first convolutional layer, one in the middle of all layers, and one before entering the fully-connected layer. We have tried with large depth value as in Alexnet but it does not work well on the dataset and thus, we try increasing and decreasing the number of depth. However, the number of low depth works better. We also keep the activation function, number of stride, number of padding and number of filter in Alexnet for each convolutional layer. We choose to output our network with sigmoid function after the fully connected layer. Also, we choose the balanced class error as our loss function. To avoid overfitting, we also use early stopping when the loss increases twice continuously. For architecture 1 we performed the suggested 2-fold cross validation. This meant that we split dataset into 2 groups/folds of equal size (50-50 split). We use one group for validation and the other group for training. We repeat this process again but this time swapping the validation and training sets used.

Architecture 2 (ZekeNet) was inspired by ResNet plus our own contributions. Like ResNet we started with a small output channel of 11. We then increased this for every layer going forward. In between layers we rotated using stride lengths of 1 and 2 and ReLU and SELU activation functions. At the end of our last convolutional layer our number of parameters reached around 8000. We need added an additional layering of max pool with 5x5 filter size and stride of 5. This effectively dropped the training time substantially without losing too much information. We also experimented with adding additional convolutional layers but every attempt resulted in lower performance. So we stuck with just the max pool layer. For ZekeNet we performed the suggested 2-fold cross validation. This meant that we split dataset into 2 groups/folds of equal size (50-50 split). We use one group for validation and the other group for training. We repeat this process again but this time swapping the validation and training sets used.

Addressing the class imbalance is important due to the nature of the dataset. Since the classes have a mean of 167 images with a standard deviation of 64 images, it is likely that some classes have more than double the amount of images than others. This imbalance leads to the model being trained in a way that could favor or perform better on certain images, simply due to the fact that there was more data to train on. In order to fix this problem, we experimented on the transfer learning network by resampling the data to have batches of equally weighted classes.

Baseline CNN	
Layer	**Description of Layer**
Convolutional Layer 1	Input Channels: 3 Output Channels: 21 Kernel Size: 3 Padding: 1 Stride: 2 Activation Function: ReLU
Convolutional Layer 2	Input Channels: 21 Output Channels: 20 Kernel Size: 3 Padding: 1 Stride: 2 Activation Function: ReLU With Batch Normalization
Convolutional Layer 3	Input Channels: 20 Output Channels: 15 Kernel Size: 3 Zero-Padding: 1 Stride: 2

	Activation Function: ReLU With Batch Normalization
Convolution Layer 4	Input Channels: 15 Output Channels: 7 Kernel Size: 5 Padding: 1 Stride: 2 Activation Function: ReLU With Batch Normalization
Fully Connected Layer 1	In-feature : 1575 Out-feature: 599 Activation Function: ReLU

Fully Connected Layer 2	In-feature = 599
	Out-feature = 201
	Activation Function: Softmax

Transfer Learning	
Layer	**Description of Layer**
ResNet	See layer architecture from ResNet (2): final out-channels to go to fully connected layers: 512
Linear	in-channels: 512, out-channels: 420
ReLU	activation function
Linear	in-channels: 420, out-channels: 201

Architecture 1	
Layer	**Description of Layer**
Convolutional Layer 1	Input Channels: 3 Output Channels: 15 Kernel Size: 5 Zero-Padding: 2 Stride: 4 Activation Function: ReLU No Batch Normalization
Max-Pooling Layer 1	Kernel Size = 3 Stride = 2
Convolutional Layer 2	Input Channels: 15 Output Channels: 35

	Kernel Size: 5 Zero-Padding: 2 Stride: 1 Activation Function: ReLU No Batch Normalization
Max-Pooling Layer 2	Kernel Size: 3 Stride: 2
Convolutional Layer 3	Input Channels: 35 Output Channels: 50 Kernel Size: 3 Zero-Padding: 1 Stride: 1 Activation Function: ReLU No Batch Normalization
Convolution Layer 4	Input Channels: 50 Output Channels: 50 Kernel Size: 3 Zero-Padding: 1 Stride: 1 Activation Function: ReLU No Batch Normalization
Max-Pooling Layer 3	Kernel Size: 3 Stride: 2
Fully Connected Layer 1	In-feature : 2450 Out-feature: 1000 Activation Function: ReLU No Batch Normalization

Fully Connected Layer 2	In-feature = 1000
	Out-feature = 201
	Activation Function: Sigmoid
	No Batch Normalization

Architecture 2: ZekeNet	
Layer	**Description of Layer**
Convolutional Layer 1	Input Channels: 3 Output Channels = 11 Kernel Size = 3 Padding = 1 Stride = 1 Activation Function = ReLU No Batch Normalization
Convolutional Layer 2	Input Channels: 11

	Output Channels = 22 Kernel Size = 3 Padding = 1 Stride = 2 Activation Function = SeLU With Batch Normalization
Convolutional Layer 3	Input Channels: 22 Output Channels = 33 Kernel Size = 3 Padding = 1 Stride = 1 Activation Function = ReLU With Batch Normalization
Convolution Layer 4	Input Channels: 33 Output Channels = 44 Kernel Size = 3 Padding = 1 Stride = 2 Activation Function = SeLU With Batch Normalization
Max-Pooling Layer	Kernel Size: 5 Stride: 5

Fully Connected Layer 1	In-feature = 1584
	Out-feature = 503
	Activation Function = ReLU

Fully Connected Layer 2	In-feature = 503
	Out-feature = 201

4. Results

4.1 Baseline

For the baseline model, we obtain a test accuracy of 52.09%. Figure 5.1.1 shows the losses for the training and validation dataset. Figure 5.1.2 shows the accuracy for the training and validation dataset. In our baseline model, we stop training after the validation losses increase two times continuously. Table 5.1.1 shows the accuracy, precision, recall and balanced classification rate for the five classes with highest accuracy and Table 5.1.2 shows accuracy, precision, recall and balanced classification rate for the five classes with the lowest accuracy.

4.2 Transfer Learning

For the transfer learning model, we obtained an overall test accuracy of 53.35% test accuracy. Figure 5.2.1 shows the losses for both the training and validation dataset. Similarly to the baseline model, we stopped training after two consecutive increases in validation loss. Table 5.2.1 shows the accuracy, precision, recall and bcr of the top five performing classes in terms of accuracy. Table 5.2.2 shows the same metrics but for the bottom five classes. In our transfer learning model, precision, recall, and bcr were far higher for the classes with the top five accuracies when compared to the bottom five. Transfer learning ended up performing very similar in terms of accuracy, precision and bcr; however, the transfer learning model had much better recall compared to the baseline model for the top five classes.

4.3 Architecture 1

For architecture 1, we obtain a test accuracy of 64.%. Figure 5.1.1 shows the losses for the training and validation dataset. Figure 5.1.2 shows the accuracy for the training and validation dataset. In our baseline model, we stop training after the validation losses increase two times continuously. Table 5.1.1 shows the accuracy, precision, recall and balanced classification rate for the five classes with highest accuracy, Table 5.1.2 shows accuracy, precision, recall and balanced classification rate for the five classes with the lowest accuracy and Table 5.1.3 shows the average accuracy, precision, recall and balanced classification rate for the all classes.

In the averaged performance metric for this architecture, we can see that all metrics including accuracy, precision, recall and balanced classification rate are all higher than those of the baseline model. The values of the both top and bottom five accuracy classes also have higher metrics value than those of the baseline CNN. Thus, we can see that overall architecture 1 is performing better than the baseline model does. It also performs better in the prediction of each class.

4.4 Architecture 2 ZekeNet

For the first iteration of the 2-fold in the training vs validation losses plot we saw the validation loss going up around epoch 9 which is also the epoch training stopped during early stopping with 5-fold cross validation. But since we couldn't enable early stopping for 2-fold cross validation we see the validation loss increasing until the max 15 epochs. Following a similar trend, this was also the case for the second iteration of the 2-fold cross validation. Although the validation curve was a bit smoother it still went up around epoch 9 until the max 15 epochs (Fig 5.4.2).

Comparing this to the original 80-20 training-validation split we see the loss graph (Fig 5.4.1) looks smooth as expected. We implemented the original architecture with early stopping at 2 consecutive epoch increases in validation loss. This ended at epoch 9 out of the max 15.

For the baseline 2-fold cross validation we see an identical trend like ZekeNet architecture. We see for 2-fold iterations the loss of validation goes up around the 9th epoch without early stopping implemented. This trend continued until the max epoch of 15.

4.5 Face Detection/Classification

In order to do face detection, we choose to do that for each labels, we will find the five highest probabilities in the prediction array and add to the matrix with each label corresponding to each label, which is a matrix of 201 and 201. So according to our algorithm, we found out that the faces with class 1 and class 109 have the highest similariy.

5. Discussion

Out of all the architectures that we experimented on, Architecture 1 performed the best and ZekeNet was second best. While the other architectures only modified specific parameters of the model, architecture 1 combined the different modifications into a single network, thus optimizing it's performance the most. Surprisingly, the transfer learning architecture did not perform as well as expected. This could be due to the fact that the features that were being learned from ResNet were too different from the facial features of our data. Applying class balancing to the dataset appeared to decrease overall performance. This is most likely due to the fact of some classes that have more images biasing the overall metrics, thus balancing the data revealed a better representation of what the performance should be for the network.

For Zekenet, we found that two fold cross-validation showed a significant drop in performance. We believe that this is because having two fold cross-validation reduces that amount of training data for each time which decreases overall performance. All performed worse on classes that had less images. This makes intuitive sense due to the fact that there was simply less data for that class that was being trained on.

ZekeNet two fold predictions for accuracies of top five and bottom five were similar to those of baseline and transfer learning, but it has more average precision and recall and bcr for top and bottom 5 classes, which means there were not as large gaps as transfer learning and baseline model. To conclude ZekeNet performed the best overall, because the average precision, recall, and bcr are more averaged.

We found that class one consistently performed poorly across all experiments and models that we performed. It was found in all of the bottom five accuracies regardless of model. Our first prediction for why this might have occured was due to the fact of class imbalance, or rather class one simply having less data to train on. Surprisingly, this class had nearly 290 images, which was two standard deviations above the mean number of images per class, thus disproving our theory.

When looking at the performance metrics on a class basis, accuracy was essentially meaningless. Due to the high number of true negatives, accuracy was very high for all classes. Precision and recall were a far better way of understanding the overall performance. High precision would indicate that the classes that we predicted were indeed the correct class, while high recall indicates the overall success in predicting all items in a specific class.

6. Author's Contributions and References

6.1 Author's Contributions

All three members of the group contributed to the completion of the project. All of the three members discussed how the convolutional neural network works and implemented different parts of the projects. Siqi mainly works on the implementation for the base model and the base file for training. Zekria works on developing another architecture and the cross validation of the base model. Garrett works on implementation of transfer learning and putting up the whole report.

6.2 References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (USA), pp. 1097–1105, Curran Associates Inc., 2012.
[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

7. Figures and Tables

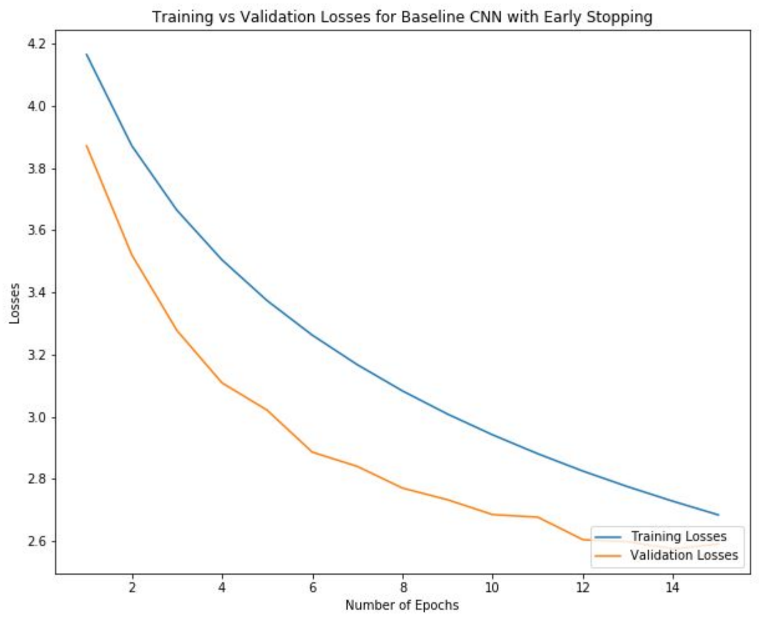


Figure 5.1.1 The above is the training versus the validation losses curves for Baseline CNN.

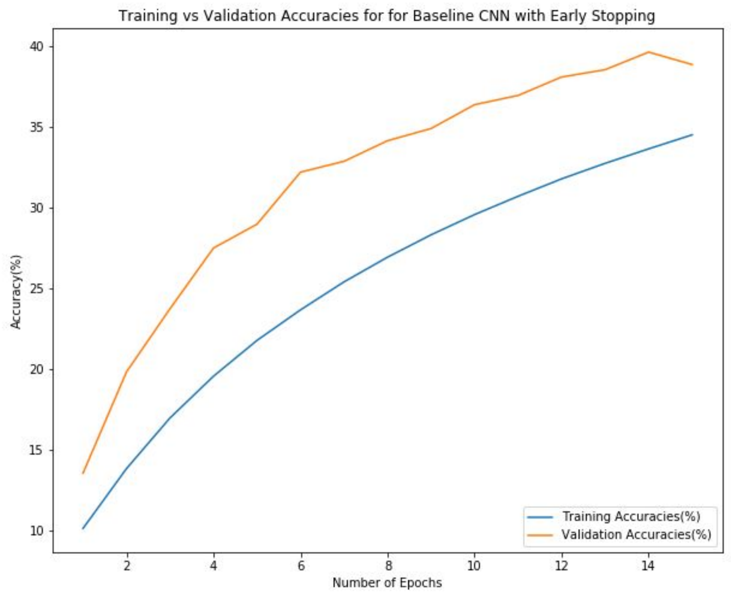


Figure 5.1.2 The above is the training versus the validation accuracy curves for Baseline CNN.

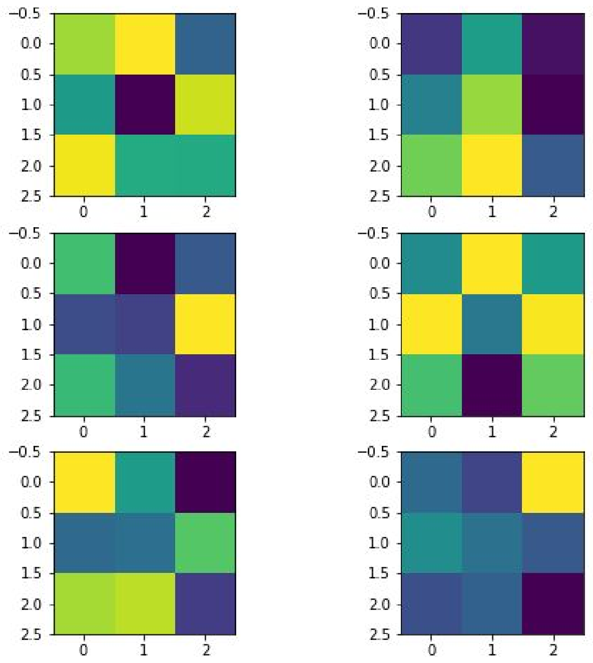
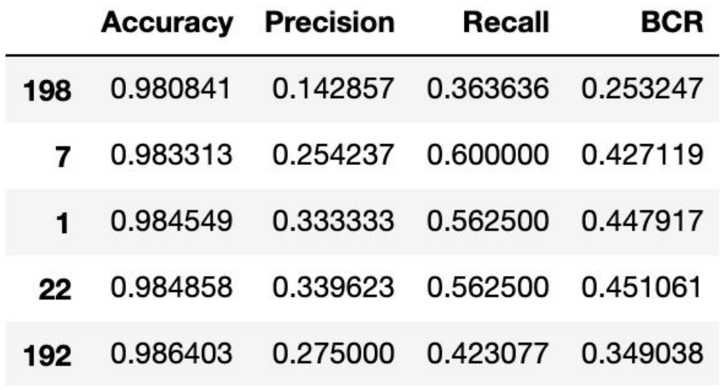


Figure 5.1.3 The above is learned filter maps of the Baseline CNN. The top 2 are for the early filter, the middle 2 are for the middle filters and the last 2 are for the late filters.

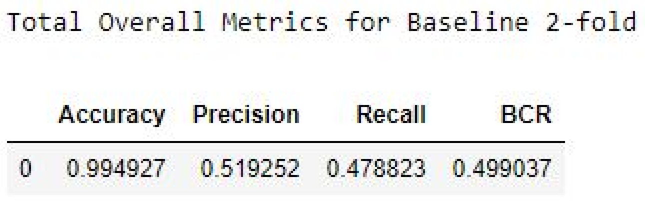
	Accuracy	Precision	Recall	BCR
199	1.000000	1.000000	1.000000	1.000000
196	0.999382	1.000000	0.750000	0.875000
91	0.999382	1.000000	0.714286	0.857143
55	0.999073	1.000000	0.571429	0.785714
47	0.999073	0.888889	0.800000	0.844444

Table 5.1.1 The above is the performances of the five classes with highest accuracy for Baseline CNN.



	Accuracy	Precision	Recall	BCR
198	0.980841	0.142857	0.363636	0.253247
7	0.983313	0.254237	0.600000	0.427119
1	0.984549	0.333333	0.562500	0.447917
22	0.984858	0.339623	0.562500	0.451061
192	0.986403	0.275000	0.423077	0.349038

Table 5.1.2 The above is the performances for the five classes with the lowest accuracy for the Baseline CNN model.



```
Total Overall Metrics for Baseline 2-fold
```

	Accuracy	Precision	Recall	BCR
0	0.994927	0.519252	0.478823	0.499037

Table 5.1.3 The above is the performances for the average accuracy, precision, recall and balanced classification rate for every class in Baseline 2-fold CNN model.

Transfer Learning:

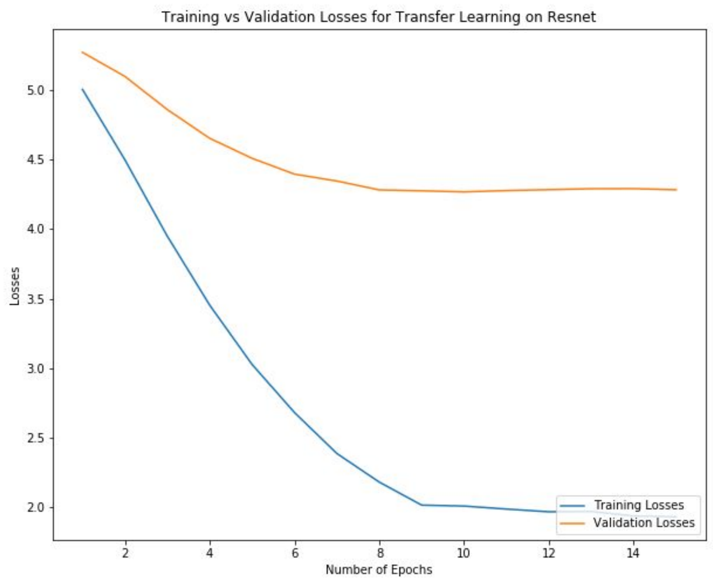


Figure 5.2.1 The above is the training versus the validation losses curve for Transfer Learning.

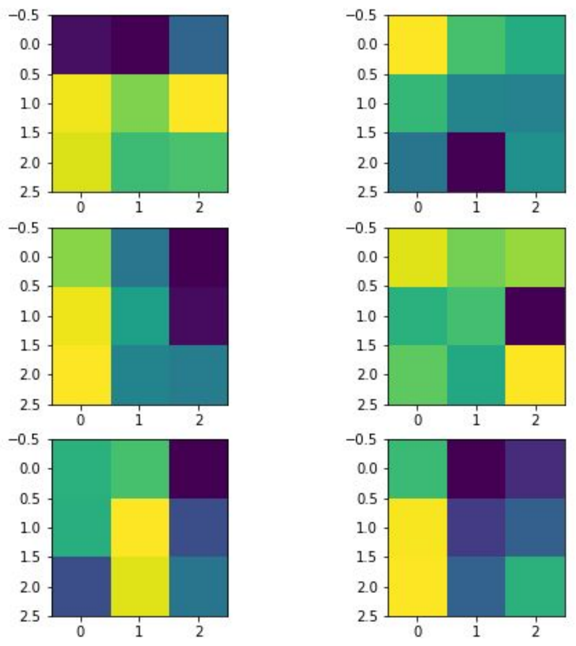
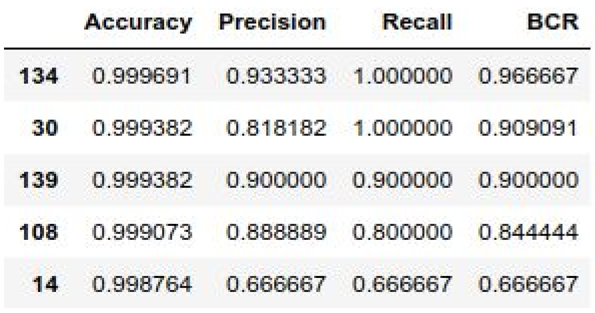
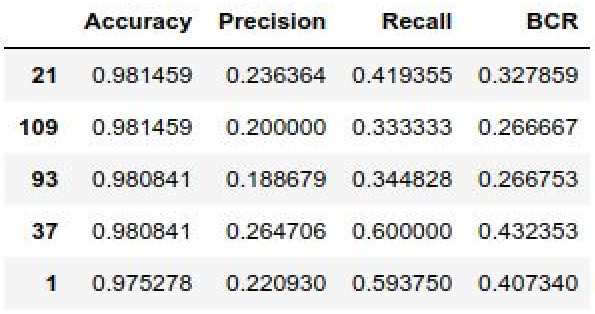


Figure 5.2.2 The above is learned filter maps of the Transfer Learning. The top 2 are for the early filter, the middle 2 are for the middle filters and the last 2 are for the late filters.



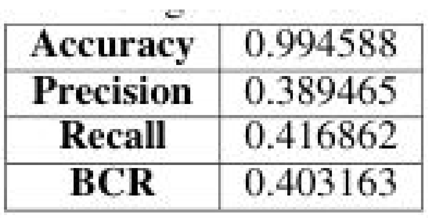
	Accuracy	Precision	Recall	BCR
134	0.999691	0.933333	1.000000	0.966667
30	0.999382	0.818182	1.000000	0.909091
139	0.999382	0.900000	0.900000	0.900000
108	0.999073	0.888889	0.800000	0.844444
14	0.998764	0.666667	0.666667	0.666667

Table 5.2.1 The above is the performances of the five classes with highest accuracy for Transfer Learning.



	Accuracy	Precision	Recall	BCR
21	0.981459	0.236364	0.419355	0.327859
109	0.981459	0.200000	0.333333	0.266667
93	0.980841	0.188679	0.344828	0.266753
37	0.980841	0.264706	0.600000	0.432353
1	0.975278	0.220930	0.593750	0.407340

Table 5.2.2 The above is the performances for the five classes with the lowest accuracy for the Transfer Learning.



Accuracy	0.994588
Precision	0.389465
Recall	0.416862
BCR	0.403163

Table 5.2.3 The above is the performances for the average accuracy, precision, recall and balanced classification rate for every class in transfer learning.

Architecture 1

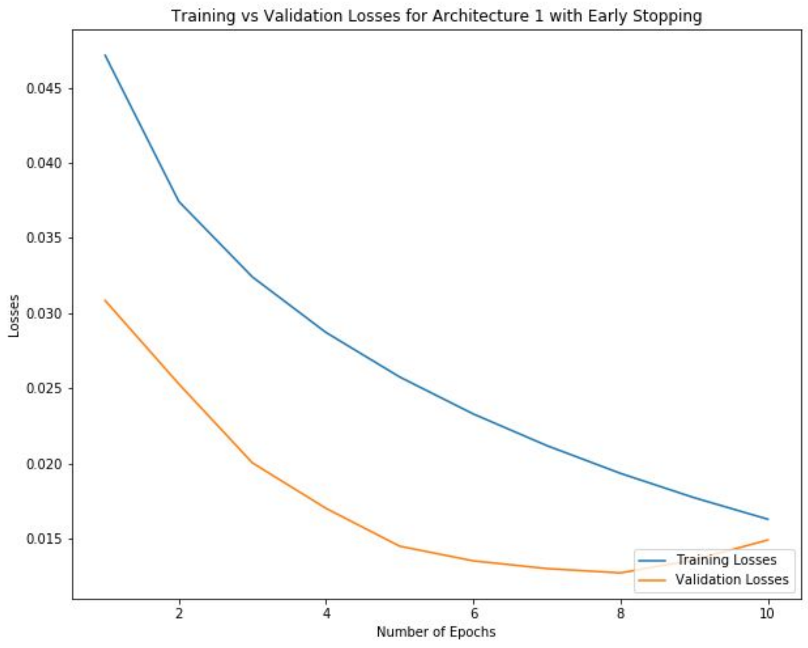


Figure 5.3.1 The above is the training versus the validation losses curve for Architecture 1.

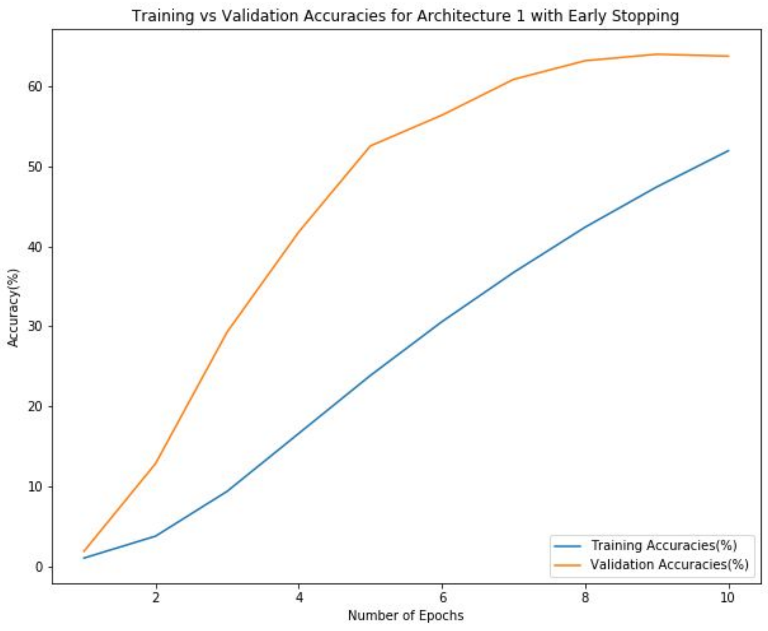


Figure 5.3.2 The above is the training versus the validation accuracy curves for Architecture 1.

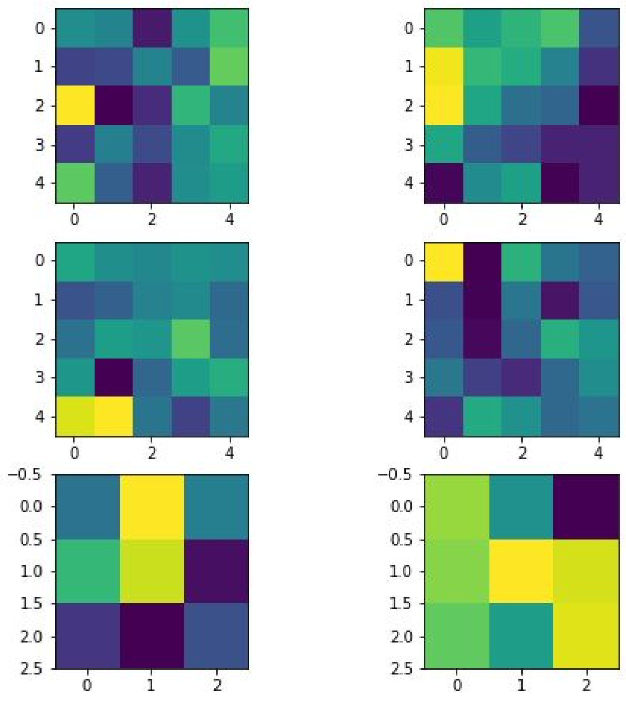
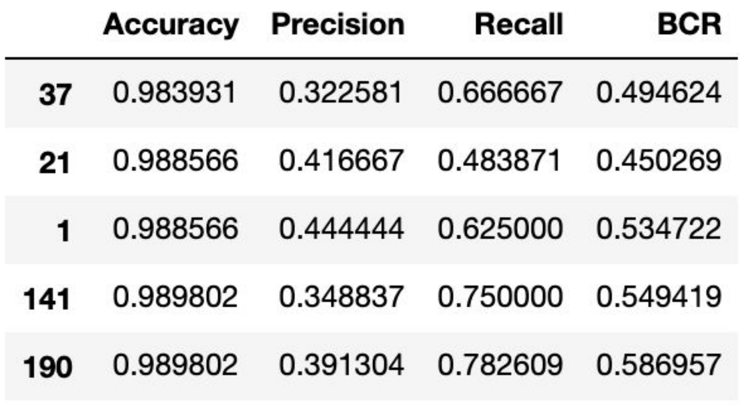
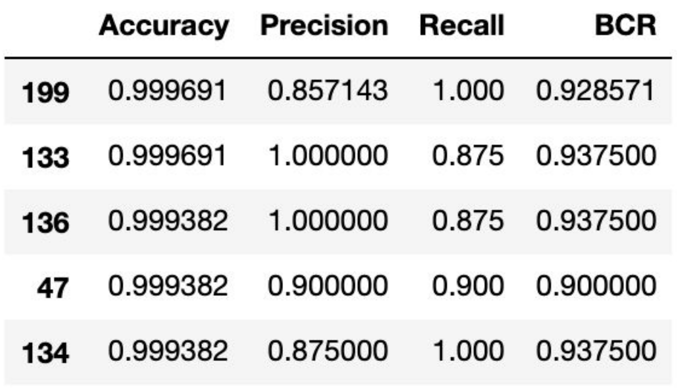


Figure 5.3.3 The above is learned filter maps of Architecture 1. The top 2 are for the early filter, the middle 2 are for the middle filters and the last 2 are for the late filters.



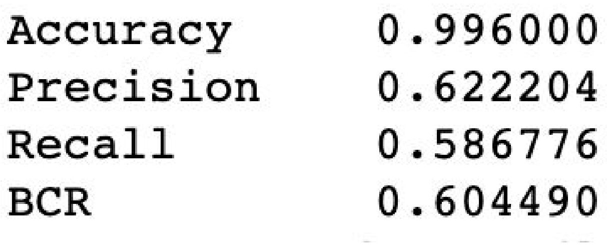
	Accuracy	Precision	Recall	BCR
37	0.983931	0.322581	0.666667	0.494624
21	0.988566	0.416667	0.483871	0.450269
1	0.988566	0.444444	0.625000	0.534722
141	0.989802	0.348837	0.750000	0.549419
190	0.989802	0.391304	0.782609	0.586957

Table 5.3.1 The above is the performances of the five classes with lowest accuracy for Architecture 1.



	Accuracy	Precision	Recall	BCR
199	0.999691	0.857143	1.000	0.928571
133	0.999691	1.000000	0.875	0.937500
136	0.999382	1.000000	0.875	0.937500
47	0.999382	0.900000	0.900	0.900000
134	0.999382	0.875000	1.000	0.937500

Table 5.3.2 The above is the performances for the five classes with the highest accuracy for Architecture 1.



```
Accuracy      0.996000
Precision     0.622204
Recall        0.586776
BCR           0.604490
```

Table 5.3.3 The above is the performances for the average accuracy, precision, recall and balanced classification rate for every class in Architecture 1.

Architecture 2 ZekeNet

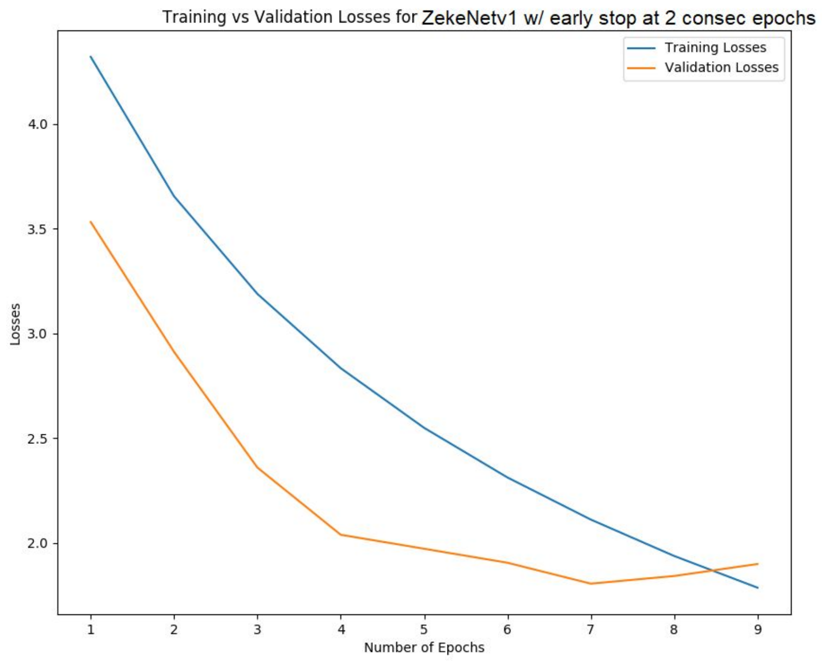


Figure 5.4.1 The above is the training versus the validation losses curve for Architecture 2 for with an 80-20 (a single 5-fold iteration) split and early stopping.

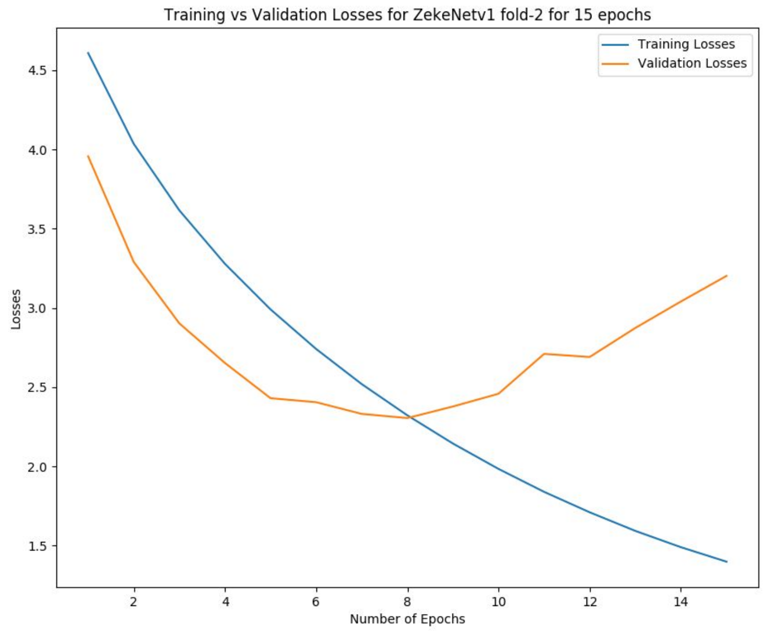


Figure 5.4.2 The above is the training versus the validation losses curve for Architecture 2 with 2-fold and no early stopping.

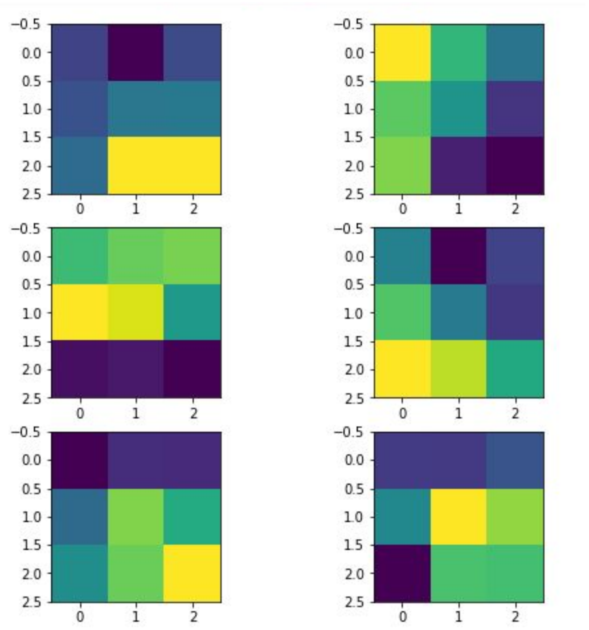
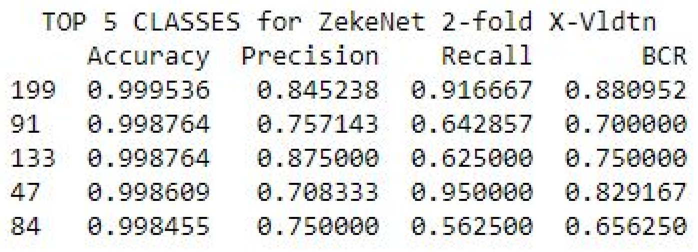
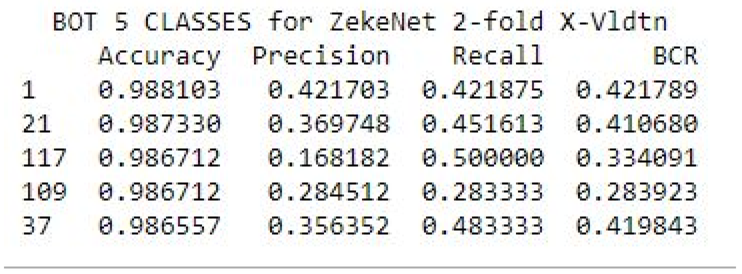


Figure 5.4.3 The above is learned filter maps of Architecture 2. The top 2 are for the early filter, the middle 2 are for the middle filters and the last 2 are for the late filters.



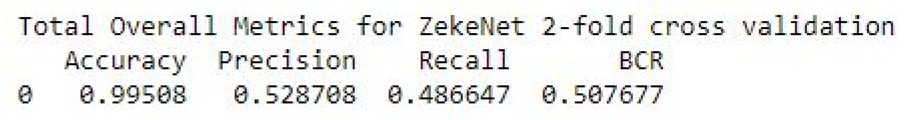
```
  TOP 5 CLASSES for ZekeNet 2-fold X-Vldtn
     Accuracy  Precision    Recall       BCR
199  0.999536   0.845238  0.916667  0.880952
91   0.998764   0.757143  0.642857  0.700000
133  0.998764   0.875000  0.625000  0.750000
47   0.998609   0.708333  0.950000  0.829167
84   0.998455   0.750000  0.562500  0.656250
```

Table 5.4.1 The above is the performances of the five classes with highest accuracy for Architecture 2.



```
  BOT 5 CLASSES for ZekeNet 2-fold X-Vldtn
     Accuracy  Precision    Recall       BCR
1    0.988103   0.421703  0.421875  0.421789
21   0.987330   0.369748  0.451613  0.410680
117  0.986712   0.168182  0.500000  0.334091
109  0.986712   0.284512  0.283333  0.283923
37   0.986557   0.356352  0.483333  0.419843
```

Table 5.4.2 The above is the performances for the five classes with the lowest accuracy for Architecture 2.



```
Total Overall Metrics for ZekeNet 2-fold cross validation
   Accuracy  Precision    Recall       BCR
0   0.99508   0.528708  0.486647  0.507677
```

Table 5.4.3 The above is the performances for the average accuracy, precision, recall and balanced classification rate for every class in Architecture 2.