COMPONENT 3

Abstract

This study focuses on the hyper-parameter tuning of the MNIST handwritten digit database using different regularization techniques, varying numbers of convolutional blocks, and learning rates. Different regularisation techniques which include L1, L2, and Batch Normalisation, were applied to the model after preprocessing the data with L2 and Batch Normalisation demonstrating the highest accuracy using Adam as the optimizer. The number of convolutional blocks was then varied, with the model using three convolutional blocks performing the best. Finally, the learning rate was adjusted, with a learning rate of 0.0005 showing the highest accuracy. Finally this report gives an insights into how hyper-parameters tuning can be used to optimize the performance of a convolutional neural (CNN).

Introduction

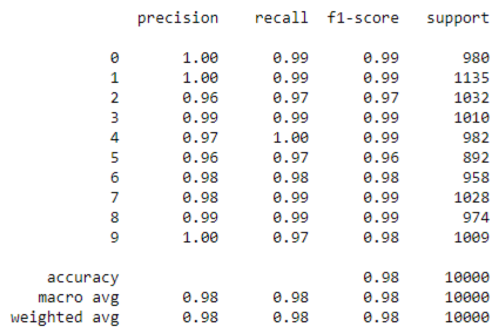
The MNIST database has become a cornerstone in the field of image processing and machine learning. This database consists of handwritten digits that have been extensively used for training and testing different kind of image processing systems. The database was developed by combining data samples from NIST's original datasets, as the creators believed that the original dataset was not suitable for machine learning experiments due to its source and image normalization. The MNIST database includes sixty thousand (60,000) training images and ten thousand (10,000) testing images. (*MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges*, no date). The main objective is to perform some hyper-parameter tuning by using different regularization, changing number of convolution blocks and varying learning rates on the dataset.

Methodology

The data was first loaded and the pixel values was normalize to a range between 0 and 1. The data was then reshaped to 4 dimensions so that it could be used with the Keras API. The transformation was then fitted to the training dataset. After the data preprocessing, the necessary libraries were imported for the model creation. The optimizer was chosen and Adam was selected for this study. Different numbers of convolutional blocks were used to determine their effect on the model's performance. The performance was evaluated using different evaluation metrics such as accuracy, loss, precision, recall, and F1 score. The results were recorded and analyzed to in order to determine the best number of convolutional blocks for this dataset.

Results and Discussion

Adam Optimizer

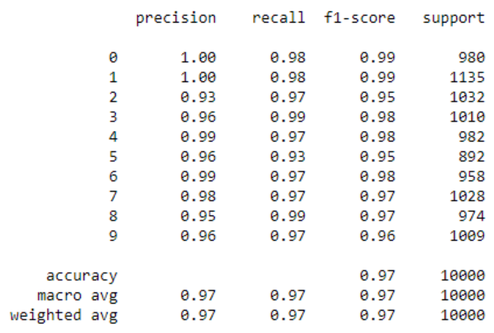


```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       980
           1       1.00      0.99      0.99      1135
           2       0.96      0.97      0.97      1032
           3       0.99      0.99      0.99      1010
           4       0.97      1.00      0.99       982
           5       0.96      0.97      0.96       892
           6       0.98      0.98      0.98       958
           7       0.98      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       1.00      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

SDG Optimizer

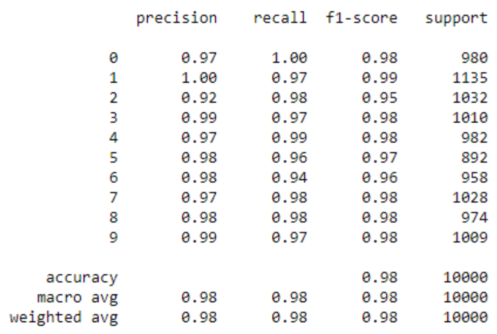


```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       980
           1       1.00      0.98      0.99      1135
           2       0.93      0.97      0.95      1032
           3       0.96      0.99      0.98      1010
           4       0.99      0.97      0.98       982
           5       0.96      0.93      0.95       892
           6       0.99      0.97      0.98       958
           7       0.98      0.97      0.97      1028
           8       0.95      0.99      0.97       974
           9       0.96      0.97      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

RMSprop Optimizer



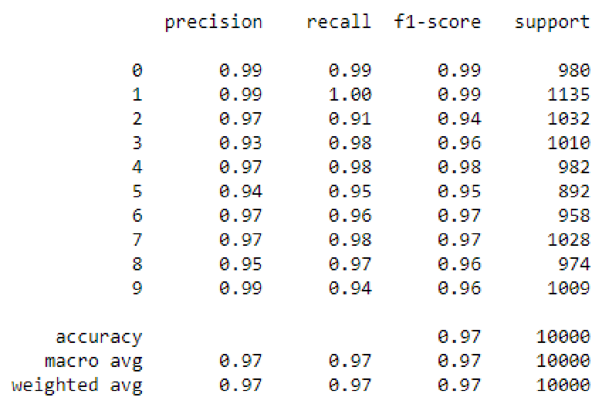
```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       980
           1       1.00      0.97      0.99      1135
           2       0.92      0.98      0.95      1032
           3       0.99      0.97      0.98      1010
           4       0.97      0.99      0.98       982
           5       0.98      0.96      0.97       892
           6       0.98      0.94      0.96       958
           7       0.97      0.98      0.98      1028
           8       0.98      0.98      0.98       974
           9       0.99      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Figure 1.0 – Classification report using ADAM, SDG and RMSprop as optimizer

Looking at the performance metrics (precision, recall, and f1-score) and accuracy from Figure 1.0, Adam optimizer performed the best compared to SGD and RMSprop optimizers. The precision, recall, and f1-score of Adam optimizer are consistently high across all classes and the overall accuracy of the model is 0.98. Therefore, Adam optimizer is recommended for this particular classification task.



```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       0.99      1.00      0.99      1135
           2       0.97      0.91      0.94      1032
           3       0.93      0.98      0.96      1010
           4       0.97      0.98      0.98       982
           5       0.94      0.95      0.95       892
           6       0.97      0.96      0.97       958
           7       0.97      0.98      0.97      1028
           8       0.95      0.97      0.96       974
           9       0.99      0.94      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

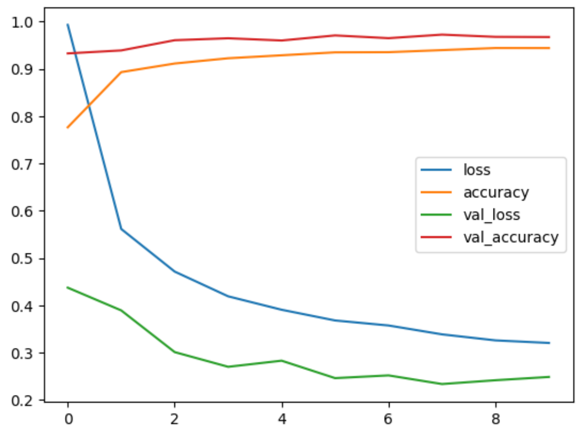
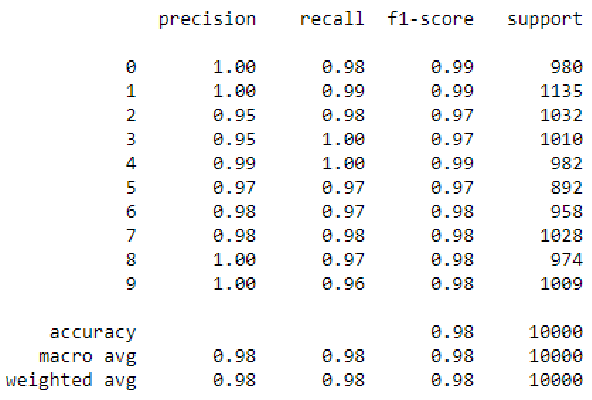


Figure 2.0 - Classification report and learning rate curve using L1 Regularizer



```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       980
           1       1.00      0.99      0.99      1135
           2       0.95      0.98      0.97      1032
           3       0.95      1.00      0.97      1010
           4       0.99      1.00      0.99       982
           5       0.97      0.97      0.97       892
           6       0.98      0.97      0.98       958
           7       0.98      0.98      0.98      1028
           8       1.00      0.97      0.98       974
           9       1.00      0.96      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

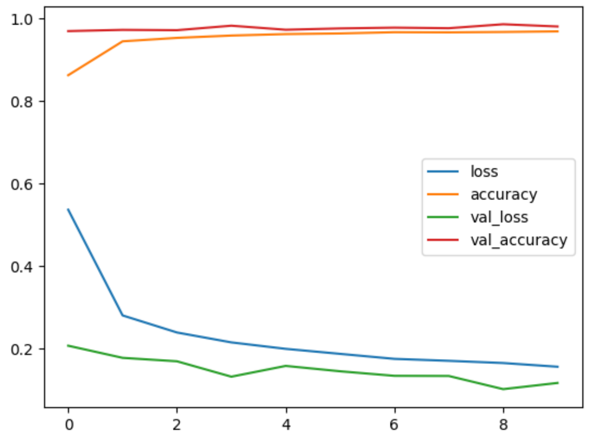
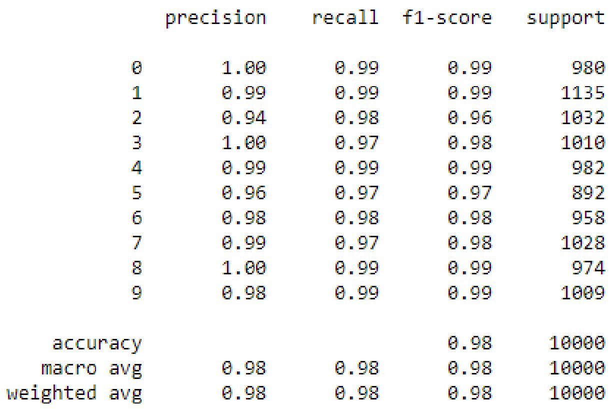


Figure 2.1 - Classification report and learning rate curve using L2 Regularizer



```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       980
           1       0.99      0.99      0.99      1135
           2       0.94      0.98      0.96      1032
           3       1.00      0.97      0.98      1010
           4       0.99      0.99      0.99       982
           5       0.96      0.97      0.97       892
           6       0.98      0.98      0.98       958
           7       0.99      0.97      0.98      1028
           8       1.00      0.99      0.99       974
           9       0.98      0.99      0.99      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

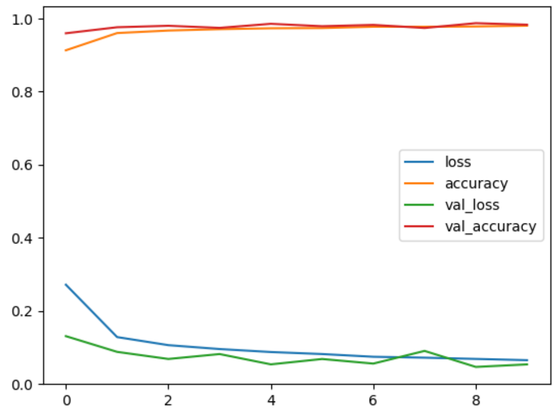
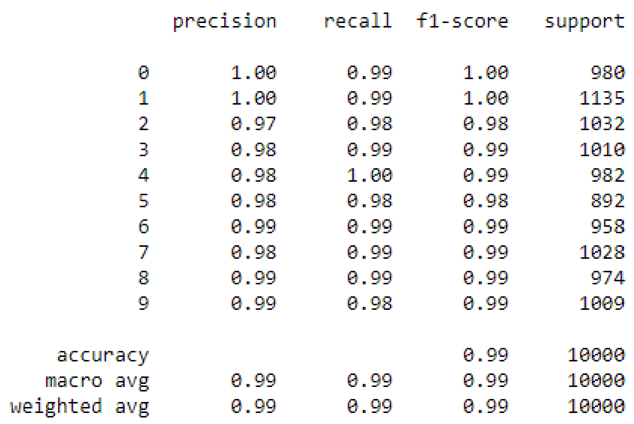


Figure 2.2 - Classification report and learning rate curve using Batch Normalization Regularizer

Figures 2.0, 2.1, and 2.2 show that the three regularisation methods have all performed well, and the model's accuracy is high. However, L2 regularization and Batch Normalization have achieved a slightly higher accuracy compared to L1 regularization. This might be because L2 regularization and Batch Normalization are more flexible and generalize better for complex models with many layers and parameters.

From the information provided in Figure 2.1 of the L2 regularizer report, we can observe that the model achieves excellent accuracy for all classes, resulting in an overall accuracy score of

98%.The precision, recall, and F1-scores are all high for each class, which indicates that model is performing well across all classes, regardless of their support. The high precision, recall, and F1-score values in the classification report suggest that the L2 regularizer is performing well on this particular dataset, providing a good balance between model complexity and performance. L1 regularization can be used for feature selection, L2 regularization for reducing the impact of outliers, and Batch Normalization for improving the training of deep neural network (Géron, 2019).



```
              precision    recall  f1-score   support

           0       1.00      0.99      1.00       980
           1       1.00      0.99      1.00      1135
           2       0.97      0.98      0.98      1032
           3       0.98      0.99      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.98      0.98      0.98       892
           6       0.99      0.99      0.99       958
           7       0.98      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.98      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

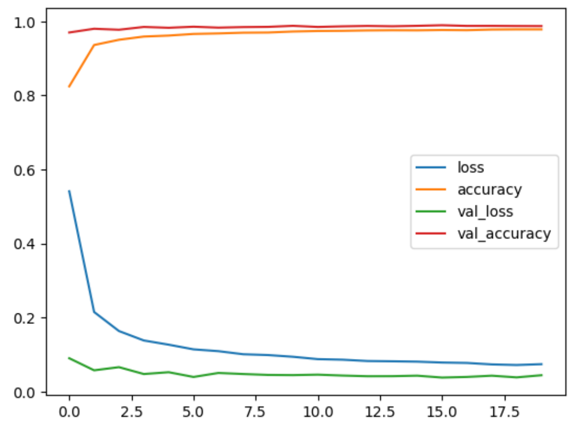
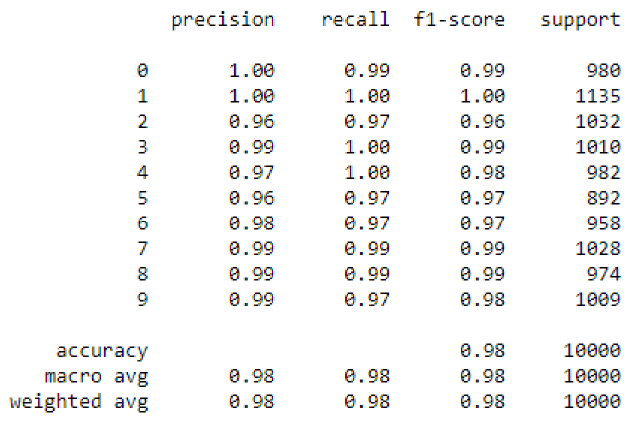


Figure 3.0 - Classification report and learning rate curve using 3 Convolutional block



```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       980
           1       1.00      1.00      1.00      1135
           2       0.96      0.97      0.96      1032
           3       0.99      1.00      0.99      1010
           4       0.97      1.00      0.98       982
           5       0.96      0.97      0.97       892
           6       0.98      0.97      0.97       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

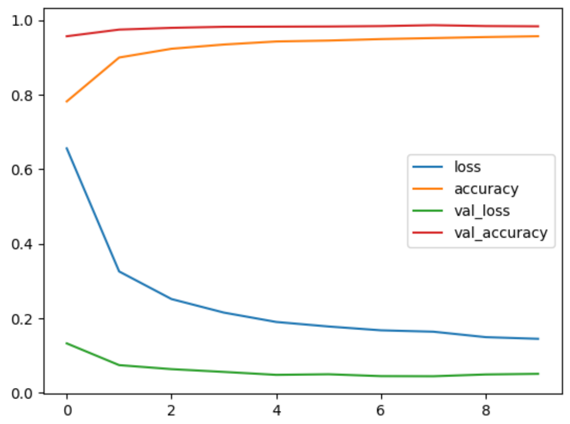
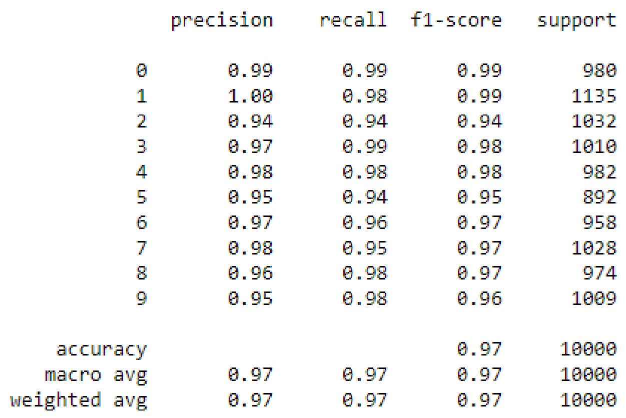


Figure 3.1 - Classification report and learning rate curve using 2 Convolutional block



```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       1.00      0.98      0.99      1135
           2       0.94      0.94      0.94      1032
           3       0.97      0.99      0.98      1010
           4       0.98      0.98      0.98       982
           5       0.95      0.94      0.95       892
           6       0.97      0.96      0.97       958
           7       0.98      0.95      0.97      1028
           8       0.96      0.98      0.97       974
           9       0.95      0.98      0.96      1009

    accuracy                           0.97     10000
   macro avg       0.97      0.97      0.97     10000
weighted avg       0.97      0.97      0.97     10000
```

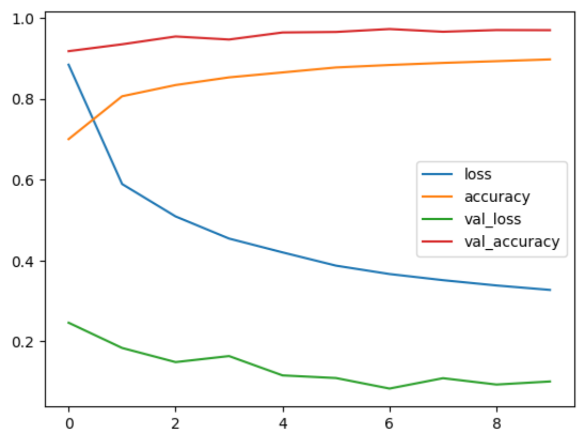
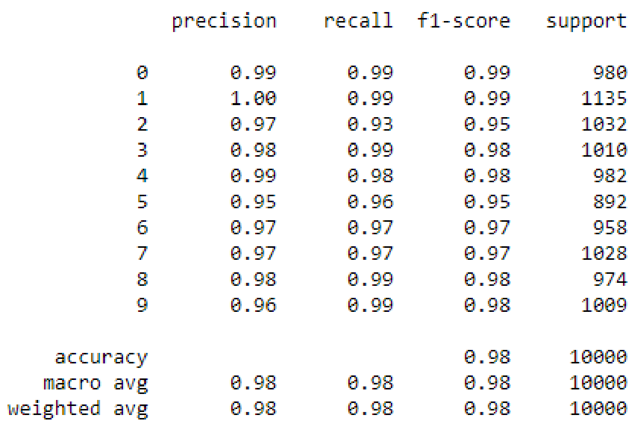


Figure 3.2 - Classification report and learning rate curve using 1 Convolutional block

As we can see from the classification reports in Figure 3.0, Figure 3.1 and Figure 3.2, the model with three convolutional blocks performs the best among the three models, with an overall accuracy of 0.99, and high precision, recall, and F1 scores for all classes. The model with 2 convolutional blocks also performs well, but has slightly lower scores compared to the model with 3 blocks, especially for class 2 and 5. The model with 1 convolutional block has the lowest overall accuracy, and lower precision and recall scores for some classes, especially for class 2. Increasing the number of convolutional blocks can increase its performance by allowing the model to learn more complicated features from the data. Increasing the number of convolution blocks can also contribute to the risk of overfitting, especially if the dataset is small or noisy (Chollet, 2021).



```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       980
           1       1.00      0.99      0.99      1135
           2       0.97      0.93      0.95      1032
           3       0.98      0.99      0.98      1010
           4       0.99      0.98      0.98       982
           5       0.95      0.96      0.95       892
           6       0.97      0.97      0.97       958
           7       0.97      0.97      0.97      1028
           8       0.98      0.99      0.98       974
           9       0.96      0.99      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

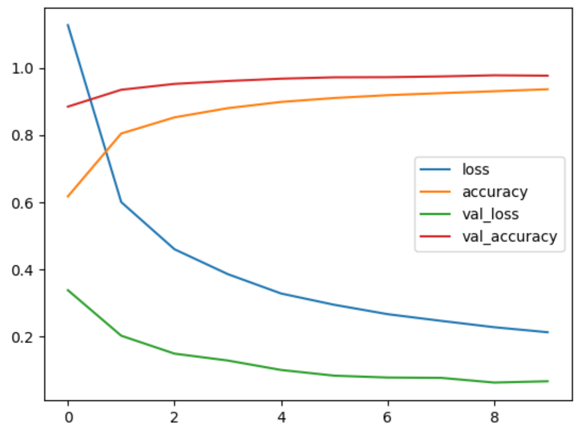
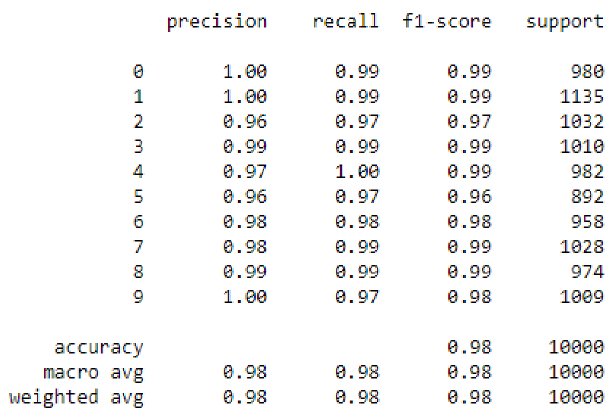


Figure 4.0 - Classification report and learning rate curve using 0.0001 Learning rate



```
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       980
           1       1.00      0.99      0.99      1135
           2       0.96      0.97      0.97      1032
           3       0.99      0.99      0.99      1010
           4       0.97      1.00      0.99       982
           5       0.96      0.97      0.96       892
           6       0.98      0.98      0.98       958
           7       0.98      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       1.00      0.97      0.98      1009

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

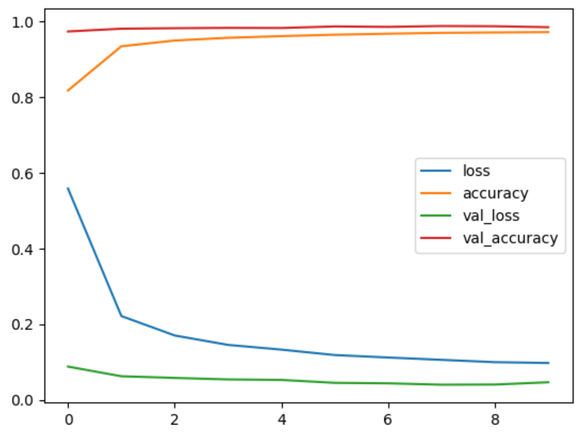
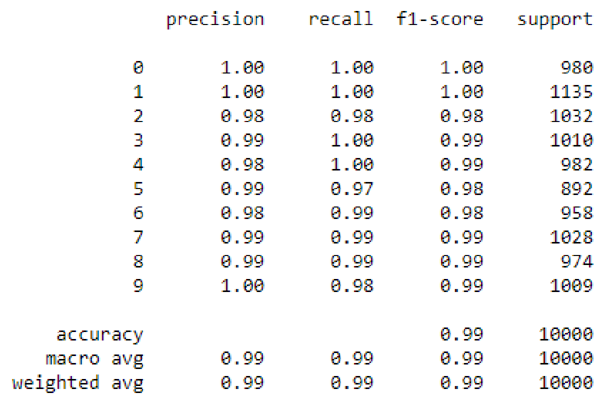


Figure 4.1 - Classification report and learning rate curve using 0.001 Learning rate



```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       980
           1       1.00      1.00      1.00      1135
           2       0.98      0.98      0.98      1032
           3       0.99      1.00      0.99      1010
           4       0.98      1.00      0.99       982
           5       0.99      0.97      0.98       892
           6       0.98      0.99      0.98       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       1.00      0.98      0.99      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

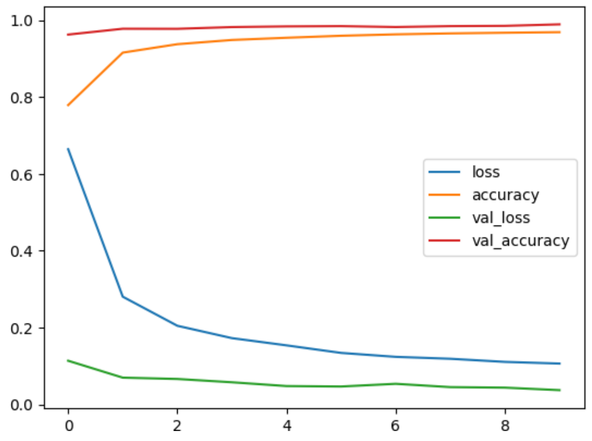


Figure 4.2 - Classification report and learning rate curve using 0.0005 Learning rate

The dataset is trained using the Adam as the optimizer with different learning rates of 0.0001, 0.0005, and 0.001 as shown in Figures 4.0, 4.1, and 4.2. The overall accuracy of the model increases from 0.98 to 0.99 when the learning rate goes from 0.0001 to 0.0005, with the precision, recall, and f1-score increasing for the majority of the classes. However, when the learning rate is increased further to 0.001, the accuracy drops back down to 0.98 and the precision, recall, and f1-score for some classes decrease as well. Faster convergence may result from a higher learning rate, but it also increases the likelihood that the model would exceed its ideal parameters and perform poorly. A lower learning rate, on the other hand, can help the model converge more slowly, perhaps leading to higher performance, but it may also result in slower training and longer training timeframes. (Goodfellow et al., 2016).

There is no overfitting at any point in model when comparing the training and validation curves. The model is generalizing well and we have a higher percentage on the accuracy of the model. If the training loss keeps decreasing while the validation loss increases, this implies that the model is overfitting to the training data and failing to perform well on new data. In such cases, specific techniques such as regularization (e.g., dropout) or early stopping could be used to prevent overfitting (Goodfellow et al., 2016).

Conclusion

The results show that the Adam optimizer performed the best among the three optimizers tested, and L2 regularization and Batch Normalization provided slightly better accuracy compared to L1 regularization. The model's performance was enhanced by increasing the number of convolutional blocks. The learning rate also had an effect on the model's performance, with 0.0005 being the greatest overall accuracy. Finally the report shows the importance of hyper-parameter tuning in achieving optimal performance for CNNs. We were able to determine the best settings for this dataset by testing with multiple hyper-parameters.

References

Chollet, F. (2021) *Deep Learning with Python, Second Edition*. Simon and Schuster.

Géron, A. (2019) *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, Inc.

Goodfellow, I., Bengio, Y. & Courville, A. (2016) *Deep Learning*. MIT Press.

MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges (no date). Available online: http://yann.lecun.com/exdb/mnist/ [Accessed 04/05/2023].