Corinna Chang
CS 171, section 1
December 13, 2022

Final Report
Classifying Tensorflow Rock Paper Scissors Dataset

I. Dataset

The Rock Paper Scissors Dataset from tensorflow contains images of hands posed as rock, paper, or scissors. The images contain hands of people from different genders, races, and ages. The images were created by Laurence Moroney using CGI techniques. The images can be classified into 3 different classes: rock, paper, or scissors.The dataset contains a total of 2892 images, with 372 testing images and 2520 training images. Each image has a plain white background and is 300×300 pixels in dimension.

II. Feedforward Neural Network

A feedforward neural network is a simple artificial neural network where information only moves in one direction from the input layer, through the hidden layer(s), and to the output layer. There are no loops that cycle through the network. The only parameter I tuned for this model is the number of hidden layers. I used the Rectified Linear Unit (ReLU) activation function and fitted the model with 1, 2, or 3 hidden layers.

Number of Hidden Layers	Training Accuracy	Test Accuracy
1	0.9846	0.6156
2	0.9877	0.5699
3	0.9916	0.5000

Figure 2.1 shows the training accuracy and test accuracy resulting from setting the model with different numbers of hidden layers.

Using 1 hidden layer in the feedforward neural network resulted in the highest test accuracy of 61.56%. Despite the test accuracy being so low, the training accuracy was very high, at 98.46%. With the increase in the number of hidden layers, the training accuracy also increases, but the test accuracy decreases. This shows that there is a high likelihood of the model overfitting the training data. These results show that a feedforward neural network is not ideal for training this image dataset.

When fitting this model, I had expected that the results were not going to be optimal. However, I wanted to see the improvement of using CNN for image classification as compared to a simple neural network. Although a test accuracy of 61.56% is not ideal, it is a good basis for comparison with other models studied in this report.

III. Convolutional Neural Network (CNN)

CNN is a network architecture for deep learning that is used for image recognition. The model learns directly from the data by processing the pixel data and finding patterns in images. Since CNN is one of the most popular neural network models that are used for image classification, it is an ideal algorithm for classifying this dataset.

Although there are many important parameters that may affect the outcomes of the training, I focused on the number of convolutional layers in CNN. I tried running the model with the number of convolutional layers set to 2, 4, and 6. The results of each output can be seen below in fig 3.1.

Number of Convolutional Layers	Training Accuracy	Test Accuracy
2	0.9062	0.8360
4	1.0000	0.8495
6	1.0000	0.9220

Figure 3.1 shows the training accuracy and test accuracy resulting from setting the CNN model with different numbers of convolutional layers.

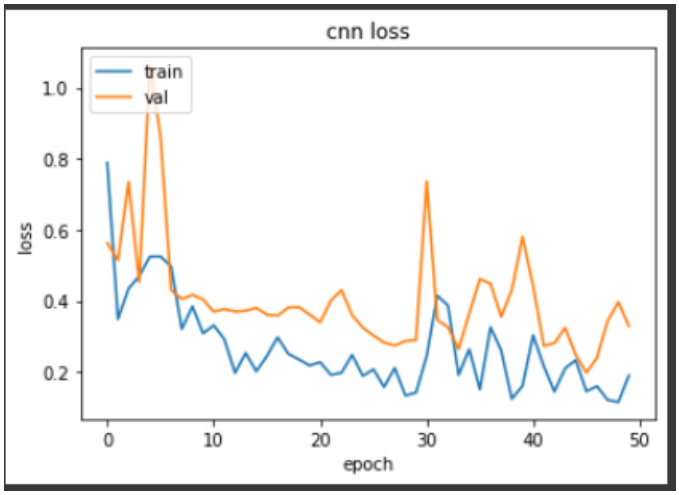

Figure 3.2 Validation loss compared to training loss in CNN with 2 convolutional layers. Since the validation loss stayed fairly consistent with training loss, it is unlikely that overfitting has occurred.

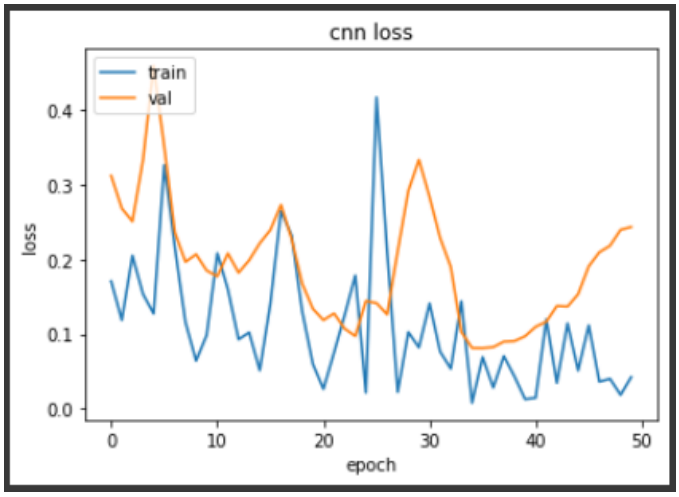

Figure 3.3 Validation loss compared to training loss in CNN with 4 convolutional layers. Since the validation loss started to increase around epoch 40 while training loss remained consistent, it is possible that overfitting has occurred.

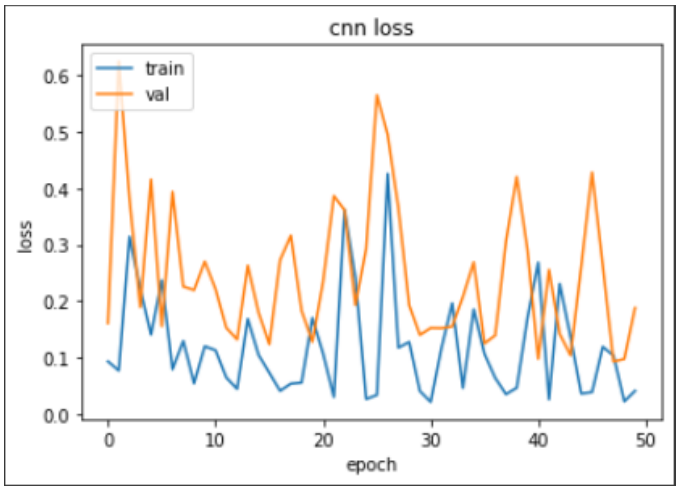

Figure 3.4 Validation loss compared to training loss in CNN with 6 convolutional layers. Since the validation loss stayed fairly consistent with training loss, it is unlikely that overfitting has occurred.

The best result is produced when the number of convolutional layers is set to 6. This returns a training accuracy of 100% and a test accuracy of 92.2%. As seen in fig. 3.4, the model is unlikely to be overfitted since validation loss stayed fairly consistent with the training loss. I had expected the accuracy to improve with the number of convolutional layers, but there is a big jump in test accuracy from 4 layers to 6 layers. Unfortunately, the program crashes when I try running more than 6 convolutional layers, so I was unable to continue testing. However, the testing accuracy of 92.2% is a satisfactory result.

IV. Support Vector Machine (SVM)

A SVM is a supervised machine learning algorithm that analyzes data for classification. The SVM algorithm finds a hyperplane in the N-dimensional space that distinctly separates data into classifications. Since SVM works best on small, complex datasets, it is an ideal algorithm for classifying the images, like the Rock Paper Scissors dataset.

The parameters I tuned are C (regularization parameter), gamma, kernel type, and degree. The degree parameter represents the degree of the polynomial kernel function, so it is ignored by all other kernels. The values used for each parameter can be seen in the table below (fig. 4.1).

C	Gamma	Kernel	Degree
0.1	0.01	linear	3
1	0.1	**poly**	5
10	1	-	**7**

Figure 4.1 Values used for parameters C, gamma, kernel, and degree in SVM. The bolded values produced the highest training accuracy.

```
Best Parameters: {'C': 0.1, 'degree': 7, 'gamma': 0.01, 'kernel': 'poly'}
Training Accuracy: 0.7190476190476189
Test Accuracy: 0.510752688172043
       C  degree  gamma  kernel  Accuracy
53  10.0       7   1.00    poly  0.719048
15   0.1       7   0.10    poly  0.719048
51  10.0       7   0.10    poly  0.719048
49  10.0       7   0.01    poly  0.719048
35   1.0       7   1.00    poly  0.719048
33   1.0       7   0.10    poly  0.719048
31   1.0       7   0.01    poly  0.719048
17   0.1       7   1.00    poly  0.719048
13   0.1       7   0.01    poly  0.719048
47  10.0       5   1.00    poly  0.685714
45  10.0       5   0.10    poly  0.685714
43  10.0       5   0.01    poly  0.685714
```

Figure 4.2 Top outputs of SVM model.

I used the GridSearchCV method imported from class sklearn.model_selection. This method allows the model to automatically perform an exhaustive search with the specified parameter values. Fig. 4.2 shows the results after fitting the training data using GridSearchCV.

The results of the tuning show that the polynomial kernel with degree 7 is the best option for classifying this image dataset. It returns a training accuracy of 71.90% and a test accuracy of 51.08%. Unfortunately, the training accuracy is still quite low, despite the tuning. In addition, the testing accuracy is much lower than the training accuracy. This shows that the model may be overfitted. The parameters C and gamma are ignored, since they did not have much effect on

the training accuracy. To conclude, although SVM can be a reliable classifying model, it is not the best way for this dataset to be classified.

V. K-Nearest Neighbor (KNN)

The KNN algorithm is a supervised machine learning algorithm that classifies data by the k nearest neighbors. The algorithm calculates the likelihood that a data point belongs to a certain classification by looking at the classification of data points closest to it. Although KNN is an intuitive and easy to implement algorithm, it can easily become computationally expensive when training large datasets. However, since this dataset is relatively small (with only 2892 samples), KNN can be a good model for training it.

The parameters I changed for this model is n_neightbors, which is the number of neighbors used for classifying a data point. Changing this parameter will have the biggest impact on the training accuracy. It is important to find the optimal number of neighbors when running KNN in order to achieve a higher accuracy for the training dataset.

Number of Neighbors	Training Accuracy
1	0.6476
8	0.5810
5	0.5286
12	0.5000
20	0.4381
25	0.4381
15	0.4048

Figure 5.1 Training Accuracy based on number of neighbors.

The parameter for 'n_neighbors' with the highest training accuracy of 64.76% is 1. It is slightly higher than the testing accuracy of 62.10%. Since both accuracy rates are fairly close, it is unlikely that the data is overfitting. However, considering how low the accuracy of this model is, KNN is not a good model for classifying the Rock Paper Scissors Dataset.

VI. Stacking Ensemble Learning

Stacking is an ensemble learning method that combines multiple models into a single model using a meta classifier. Individual models are first trained using different subsets of the dataset, and training accuracy from each of them are combined to make the final prediction. This allows the final model to have an improved performance as compared to the individual models. However, stacking works better for bigger datasets and is especially useful for dealing with imbalanced datasets to reduce variance. It is a more complex approach, and therefore,

more computationally expensive compared to other algorithms mentioned in this report. As a result, stacking may not be the best approach for this dataset due to its small size.

The stacking algorithm takes parameters estimators, a list of estimators that are stacked together, and a final_estimator, the classifier that is used to combine the base estimators. I decided to use the classifiers I have already experimented with, SVM and KNN. In addition, I decided to also include the random forest classifier and MLP classifier. I tried fitting the dataset by setting one of these classifiers as the final_estimator and the rest were put into a list as the base estimators.

final_estimator	Training Accuracy (%)	Test Accuracy (%)
MLP	100.00	62.90
Random Forest	100.00	48.12
KNN	90.62	47.31
SVM	100.00	33.33

Figure 6.1 Classifiers used for the stacking algorithm and their respective training accuracy and test accuracy.

As can be seen in fig. 6.1, when the final_estimator is set to MLP, it produces the highest testing accuracy of 62.9%. The training accuracy of this dataset is 100%, which is considerably higher than the testing accuracy, which suggests that the model is overfitting. In fact, the model seems to be overfitting no matter which final_estimator is used. This was surprising because I had expected the stacking algorithm to have an improved performance as compared to the individual models. However, it has a much lower test accuracy than KNN (64.76%), but it did perform better than SVM (51.08%). Because of the poor performance and overfitting, the stacking model is not ideal for classifying the Rock Paper Scissors Dataset.

VII. Conclusion

Classifier	Test Accuracy (%)
CNN	92.20
Stacking	62.90
KNN	62.10
Feedforward Neural Network	61.56
SVM	51.08

Figure 7.1 Highest test accuracy returned after tuning each classifier.

In the figure above, we can see that CNN returned the highest testing accuracy (92.20%) out of all the models, and thus is the most optimal model for fitting this dataset. Many of these models were overfitting, including the Feedforward Neural Network, SVM, and stacking algorithm. Although KNN did not overfit, it returned a very low test accuracy, which makes it not very effective for classifying this image dataset. A possible cause for the overfitting is that this dataset is too small. From Midterm 1, I had classified the Tensorflow MNIST Dataset using a simple neural network and obtained a test accuracy of 98.17%. The MNIST Dataset has a total of 70,000 examples, while the Rock Paper Scissors Dataset only has 2892.

Possible experiment I can conduct on this dataset in the future is fitting it with other image classification techniques, such as Naïve Bayes Algorithm and Random Forest Algorithm. In addition, I would like to figure out the exact cause of the issue with overfitting by tuning additional parameters from each model. Although I am content with the result of 92.20% accuracy I obtained using CNN, applying additional classification algorithms to the dataset may further improve the accuracy to a higher rate, such as 99.7%.

VIII. References

https://becominghuman.ai/simple-neural-network-on-mnist-handwritten-digit-dataset-61e47702ed25
https://www.kaggle.com/code/alroger/mnist-cnn-svm-knn
https://www.kaggle.com/code/duonggiakhanhb/rock-paper-scissors-for-ml/notebook#Rock-Paper-Scissors-for-Machine-Learning-mid-term
https://www.kaggle.com/code/julianowakowska/image-classifier-rock-paper-scissors/notebook?scriptVersionId=100497907
https://laurencemoroney.com/datasets.html
https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac
https://medium.com/analytics-vidhya/rock-paper-scissor-cnn-670f5ffec767
https://medium.com/geekculture/rock-paper-scissors-image-classification-using-cnn-eefe4569b415
https://stackoverflow.com/questions/34972142/sklearn-logistic-regression-valueerror-found-array-with-dim-3-estimator-expec
https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
https://www.tensorflow.org/tutorials/images/cnn#import_tensorflow
https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d