Rock-Paper-Scissors

Predicting gesture with CNN

Aleksandra Zografska

48924A

12.07.2025

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

INTRODUCTION

In this paper I am going to build **5** convolutional neural networks, using cross validation and neural network structures, as well as evaluate how data preprocessing affects the end result of the model. The processing of the data will be done in Python with the help of the keras library.

DATA

All my analysis and processing is based on the [Rock-Paper-Scissors dataset] from Kaggle. There are 726 images of 'Rock', 710 images of 'Paper' and 752 images of 'Scissors'. In order to perform cross-validation and then test predictions on the real data I divided the folder in three parts: train, validation and test, containing 70%, 20% and 10% of the data respectively with a python script named `restructure.py`. Before dividing, I performed a random shuffle on the data.

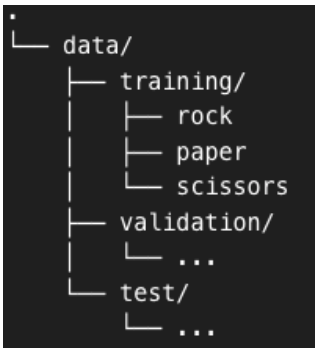


Figure 1. Data folder restructure

After this division, I loaded the data, which was already sized as in the shape of 300x200px. Initially I ran some models on this data shape, and after in order to improve my training time and overall model performance I resized the images to 150x150px. I performed data augmentation by introducing a random horizontal and vertical flip, as well as random crop, brightness, zoom and contrast. On the resulting images I performed image normalization.

After I visualised the initial and the augmented dataset to see the differences.

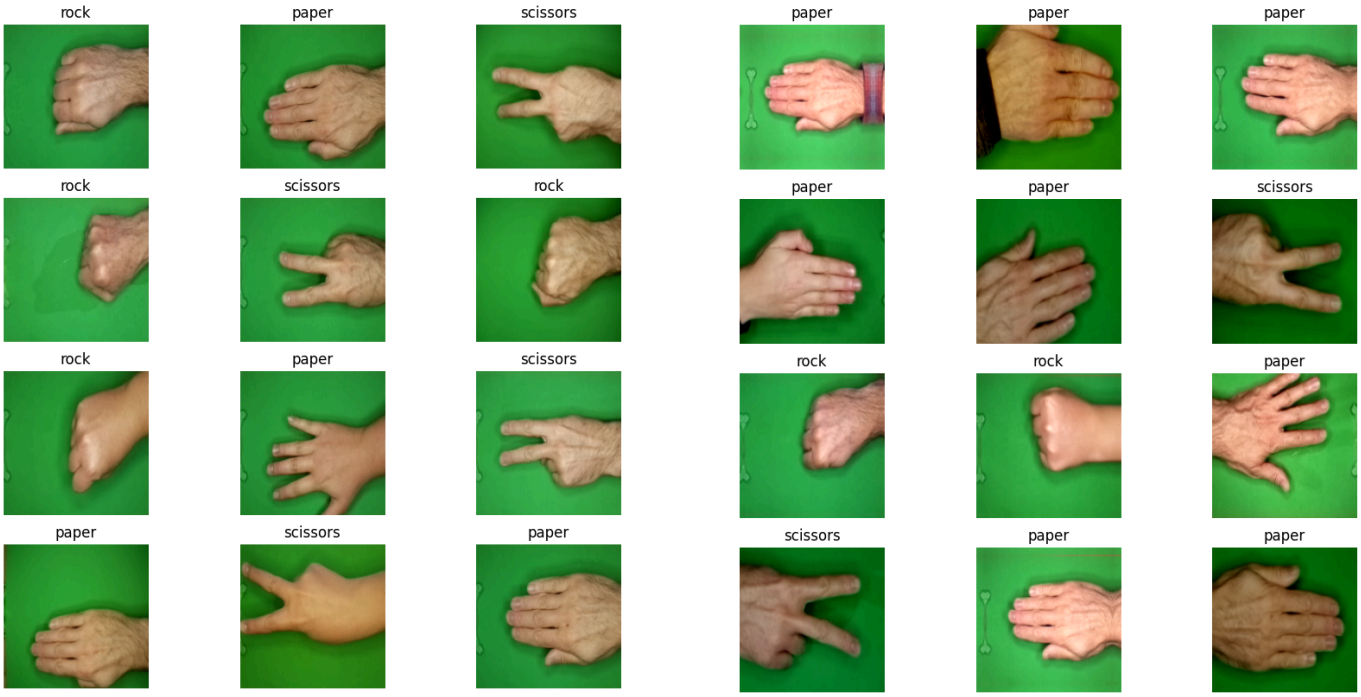


Figure 2. (left) pure dataset, Figure 3. (right) augmented dataset

MODELS AND TRAINING

SIMPLE MODEL

First, I created a simple model, with an input shape of 200x300, containing one convolutional layer, one hidden Dense layer and an output Dense layer with a softmax activation function. In a way, this simple model can be used as a benchmark for the following more complex models. In my model, I tuned the number of filters for the convolutional layer, the number of units in the Dense layer and the learning rate of the Adam optimizer. In all the networks I choose the Adam optimizer since it is better for CNN models, it converges more quickly, it's good for noisy data and complex models.

By using keras Hyperband tuner, I was able to more optimally tune the parameters and got the following results.

- Best # of convolutional filters: 64
- Best # of dense units: 512
- Best learning rate: 0.001

Since the hyperparameter tuning took a lot of time, I decided to check the best batch size by running the simple model along with the best hyperparameters through a loop of different batch sizes. The result was batch_size = 32.

In the Image below we can see the layers of the simple neural network, as well as the number of parameters they take. This model has 241,680,896 trainable parameters.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 298, 198, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 149, 99, 32)	0
flatten_3 (Flatten)	(None, 472032)	0
dense_6 (Dense)	(None, 512)	241,680,896
dense_7 (Dense)	(None, 3)	1,539

Figure 4. Simple Model Architecture

From this graph (Figure 5.) we can conclude that there are evident signs of overfitting, signaled from the high training accuracy, as well as the growth of validation loss over epochs, while training loss goes down.

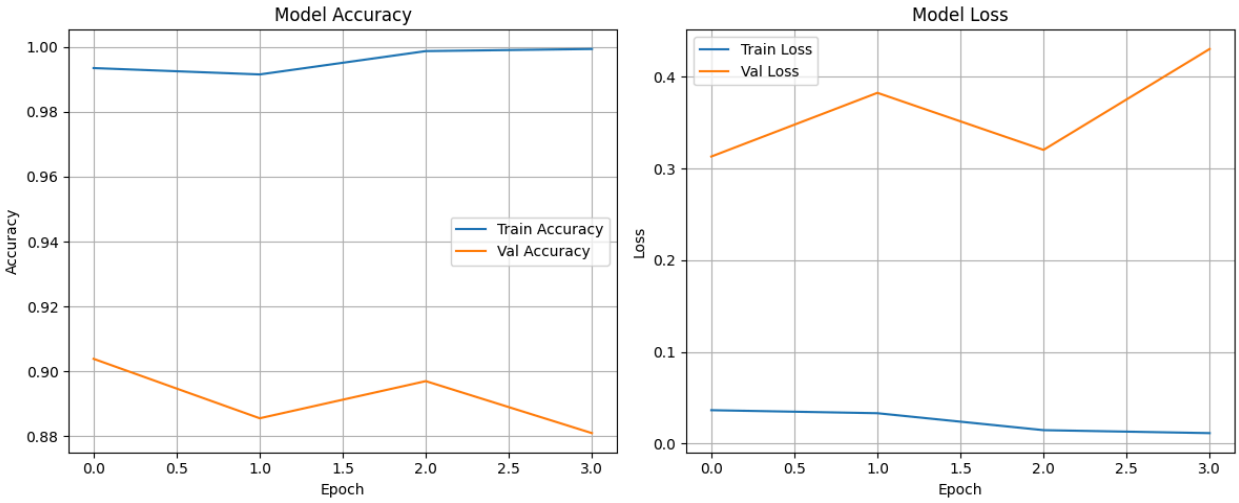


Figure 5. Simple Model Training History

IMPROVED SIMPLE MODEL

In order to improve this simple model, I decided to use the dropout layer after the Convolutional NN layer and the Dense layer, with dropout rates of 0.15 and 0.3

respectively. This type of layer randomly selects a fraction of the NN's outputs and sets it to zero, which in term helps reduce overfitting. The results from the improved model are below (Figure 6.).

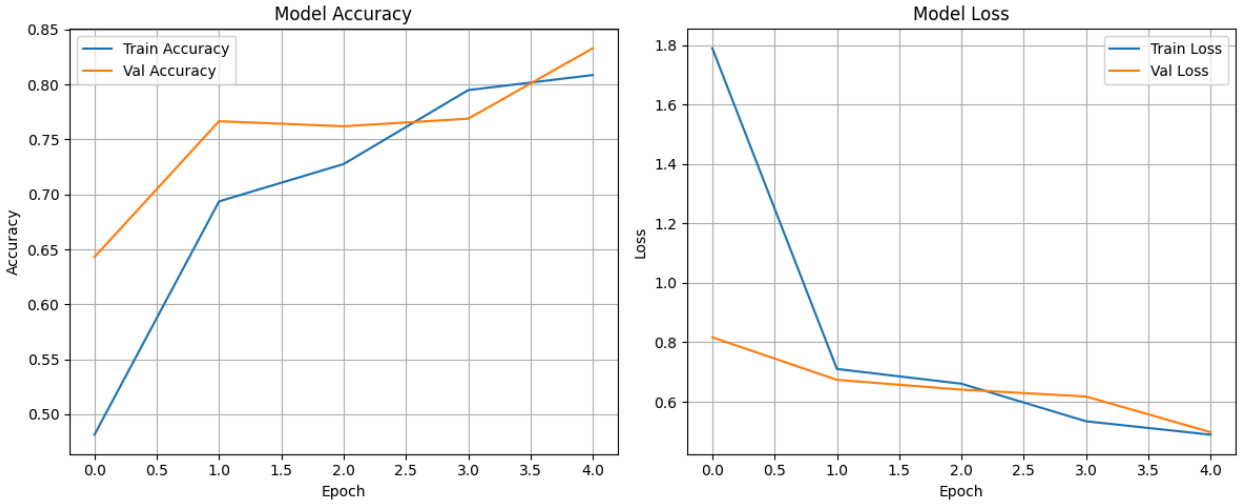


Figure 6. Improved Simple Model Training History

This new graph shows both training curves increase together, which gives some signs of improvement on the generalization of the model. For this model, the training time took around 15-20 minutes, training on max of 10 epochs, and having an early stopping callback, that stops training after 2 epochs of increasing validation loss.

SIMPLE MODEL with IMAGE RESIZING (150x150)

In order to make a further attempt in improving the accuracy and training time, I resized the image into smaller dimensions 150x150. It's worth noting that the number of parameters has significantly decreased to 174,621,184.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
conv2d_1 (Conv2D)	(None, 146, 146, 64)	18,496
max_pooling2d (MaxPooling2D)	(None, 73, 73, 64)	0
flatten (Flatten)	(None, 341056)	0
dense (Dense)	(None, 512)	174,621,184
dense_1 (Dense)	(None, 3)	1,539

Figure 6. Simple Model with Resized Images Architecture

In the graph below, we can see the new training results.

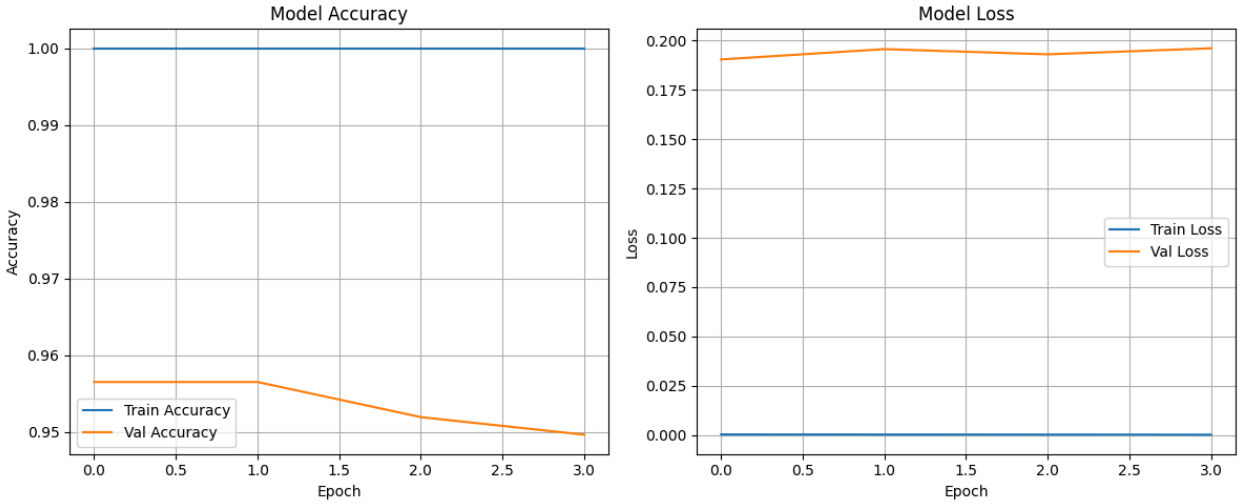


Figure 7. Simple Model with Resized Images Training History

Since the first epoch in this model we have a training accuracy of 1, which could be a sign of overfitting of the model, although validation accuracy is high, there is still a 4% difference between them, as well as a drop in validation accuracy in the later epochs. This overfit might occur because I lowered the overall number of parameters of the model by resizing the image. To fix it, I will again add a Dropout layer, with higher dropout rate because the overfit is bigger.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 74, 74, 32)	0
dropout_2 (Dropout)	(None, 74, 74, 32)	0
flatten_2 (Flatten)	(None, 175232)	0
dense_4 (Dense)	(None, 512)	89,719,296
dropout_3 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 3)	1,539

Figure 8. Improved Simple Model Architecture

Since the Dropout layer introduces randomness, the validation accuracy is expected to

go a bit up and down, until it stabilizes. Additionally, the model needs more epochs to converge, so I run the fit on 20 epochs, with an early stop tolerance of 5 epochs.

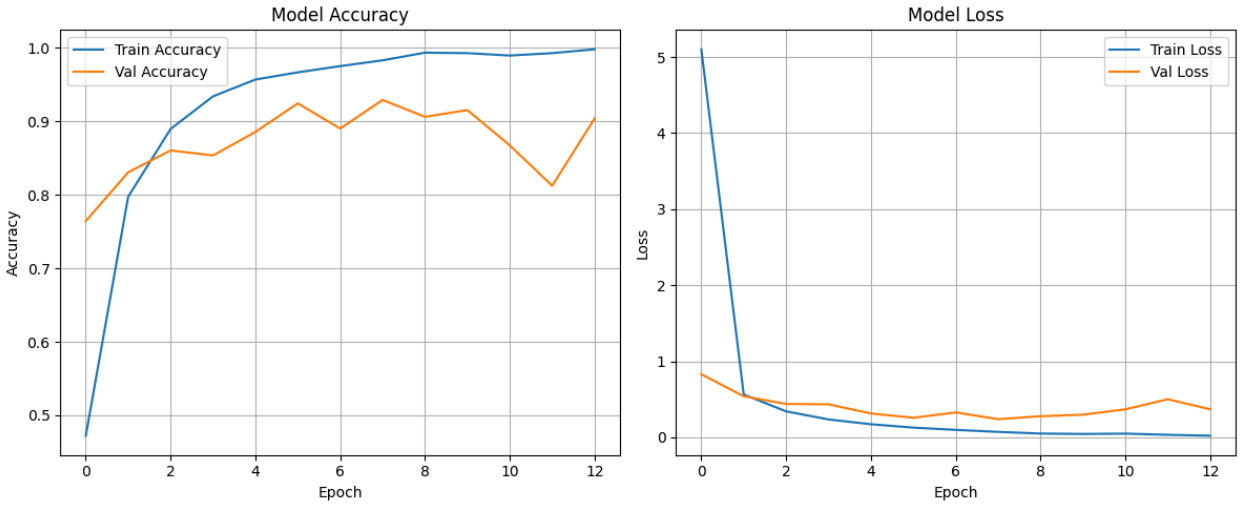


Figure 9. Improved Simple Model with Resized Images Training History

EXTENDED MODEL

The extended model is more complex and has 2 convolutional neural networks and 2 hidden layers. The convolutional layers have 32 and 64 filters respectively, since the initial layer captures smaller features, and the next layer captures more complex structures like patterns. I added an additional hidden layer, since it can help with generalization, which is something the previous model needed, since it scored less on unseen data. After every NN layer there is a Dropout layer with rate 0.1, to prevent overfitting on the data.

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_7 (MaxPooling2D)	(None, 74, 74, 32)	0
dropout_8 (Dropout)	(None, 74, 74, 32)	0
conv2d_9 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_8 (MaxPooling2D)	(None, 36, 36, 64)	0
dropout_9 (Dropout)	(None, 36, 36, 64)	0
flatten_5 (Flatten)	(None, 82944)	0
dense_11 (Dense)	(None, 128)	10,616,960
dropout_10 (Dropout)	(None, 128)	0
dense_12 (Dense)	(None, 512)	66,048
dropout_11 (Dropout)	(None, 512)	0
dense_13 (Dense)	(None, 3)	1,539

Figure 10. Extended Model Architecture

The result of the fitting is training and validation accuracy increasing together, as well as loss decreasing respectively. Both are good signs for the generalization improvement of the model.

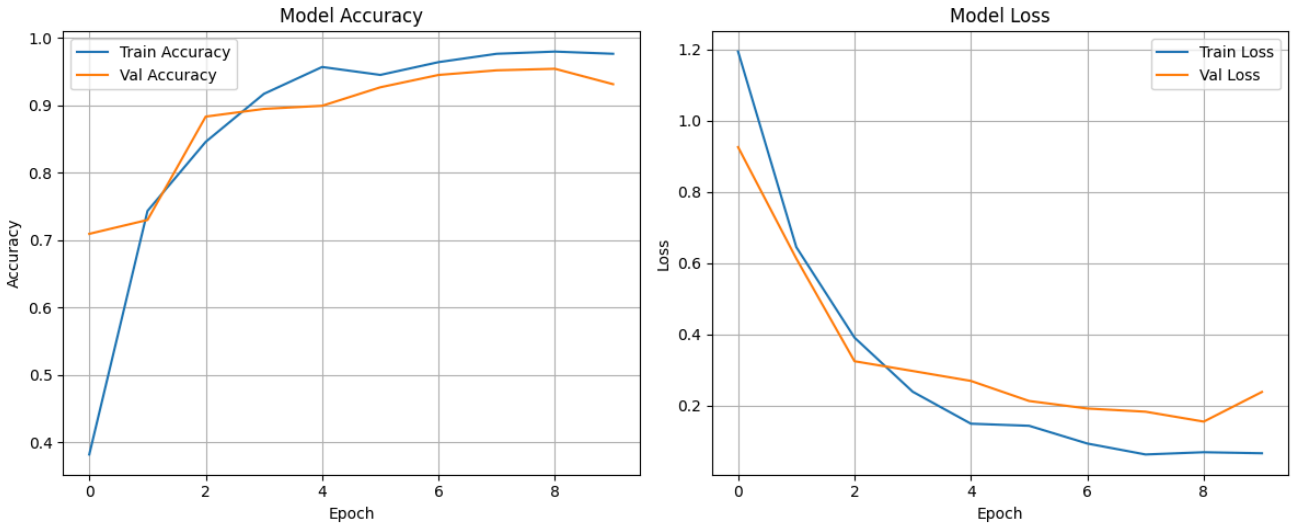
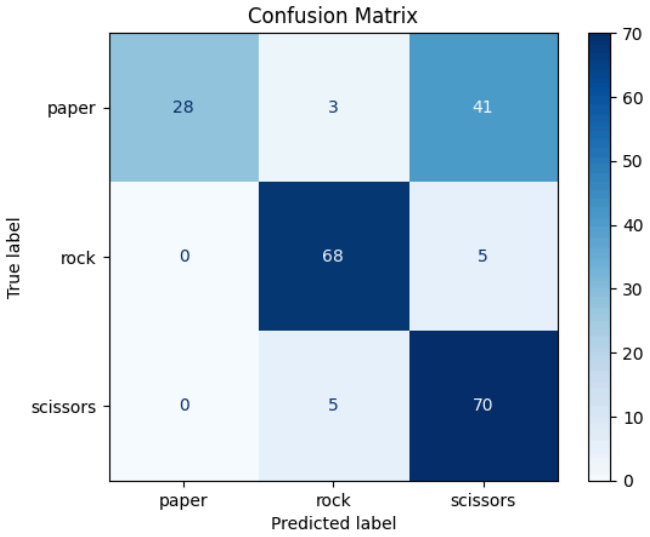


Figure 11. Extended Model Training History

RESULTS AND EVALUATION

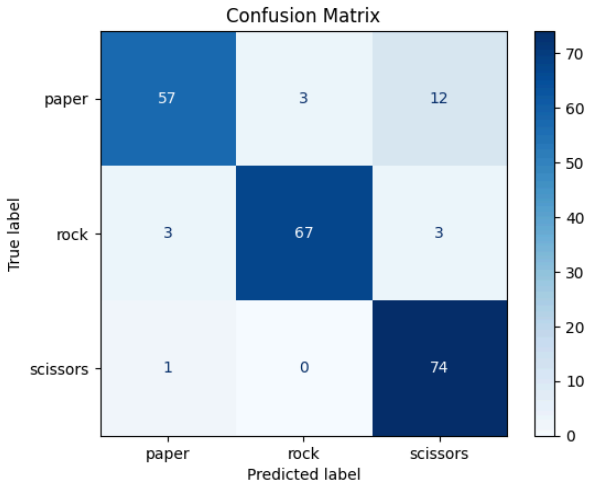
The final results from the models, correspond with what we see on the training and validation charts. The models are tested on unseen data, in order to assess the generalization capabilities.



Simple Model (Accuracy: 0.7545)

category	precision	recall	f1-score
paper	1.0	0.388	0.56
rock	0.894	0.931	0.912
scissors	0.603	0.933	0.732

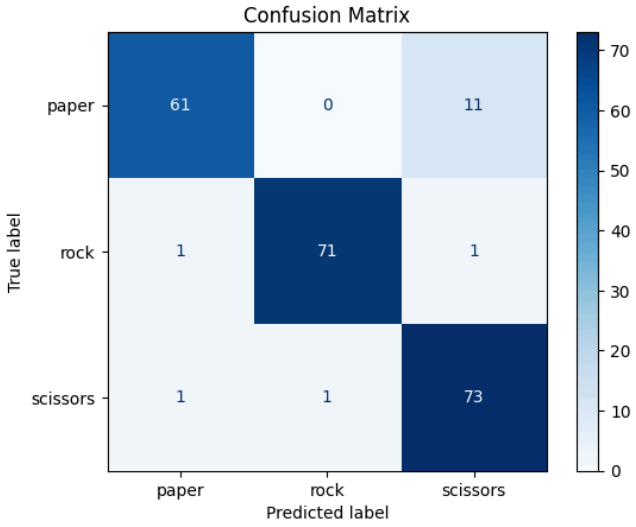
Precision means that every time the model predicts a class, it's always true. In the case of paper, it always predicts paper images as paper. While for scissors it has a lower precision. On the other hand, the recall for paper is really small. The recall tells us how much of the true paper classes were predicted as paper. The confusion matrix also lets us see more clearly that a lot of the true paper datapoints are predicted as scissors, which also accounts for the low scissors precision. F1-score indicates if the model has high precision and recall, since it is derived from those metrics. Here we see that for the class rock there is a good trade-off balance between paper and scissors, while for the other two classes it is lower, and indicates the opposite.



Improved Simple Model (Accuracy 0.883)

category	precision	recall	f1-score
paper	0.934	0.791	0.857
rock	0.957	0.917	0.937
scissors	0.831	0.986	0.902

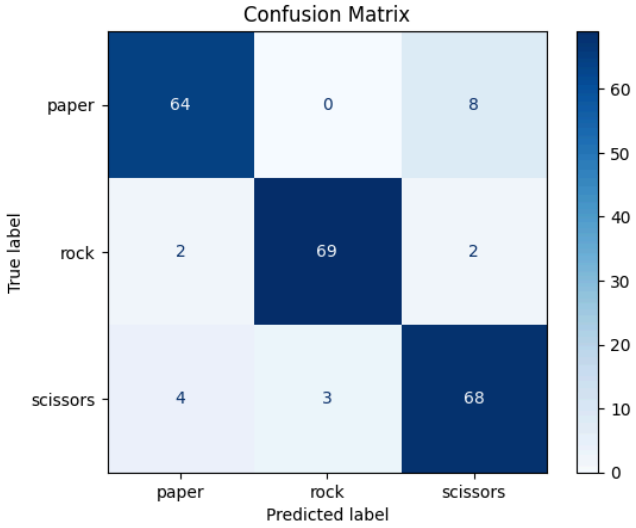
There is a significant improvement in the precision of predicting paper and scissors as well as the recall, then f1 score. It's also evident from the deeper colors along the diagonal line in the confusion matrix.



Simple Model Resized (Accuracy: 0.931)

category	precision	recall	f1-score
paper	0.996	0.847	0.903
rock	0.986	0.972	0.979
scissors	0.858	0.973	0.912

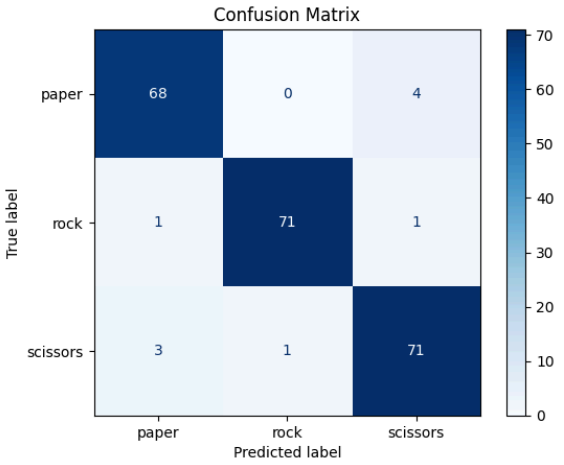
This model has a bigger improvement in the f1-scores compared to the non-resized image models.



Improved Simple Model Resized (Accuracy: 0.913)

category	precision	recall	f1-score
paper	0.914	0.888	0.901
rock	0.958	0.945	0.951
scissors	0.871	0.906	0.888

This model has further improvement on the f1-score, except for the one on scissors. And for now, has the smallest misclassified paper classes as scissors.



Extended Model (Accuracy: 0.954)

category	precision	recall	f1-score
paper	0.944	0.944	0.944
rock	0.986	0.972	0.979
scissors	0.934	0.946	0.940

This model has the largest improvement in accuracy where all the classes have an f1-score above 0.94. The big success rate is also evident from the confusion matrix, only having 10 misclassified samples, on a test size of 210. Still the most misclassified is paper, and it's why it has the smallest recall rate from all the classes.

GENERALIZATION TEST

I took three pictures per class of my hand, to see how well the model does on generalization. I took the pictures behind an olive cloth in direct sunlight. I predicted them on the Extended model.

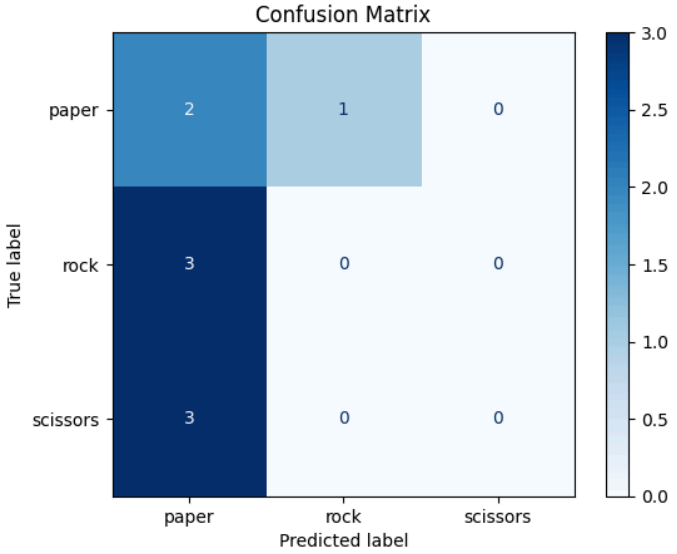


Figure 17. Confusion matrix on predictions on raw personal data

After getting the initial model, I got an idea on how to maybe improve the result. I removed the background of the images and replaced it with a green background, to resemble the background of the training data. This led to some more promising results, of accuracy 66%

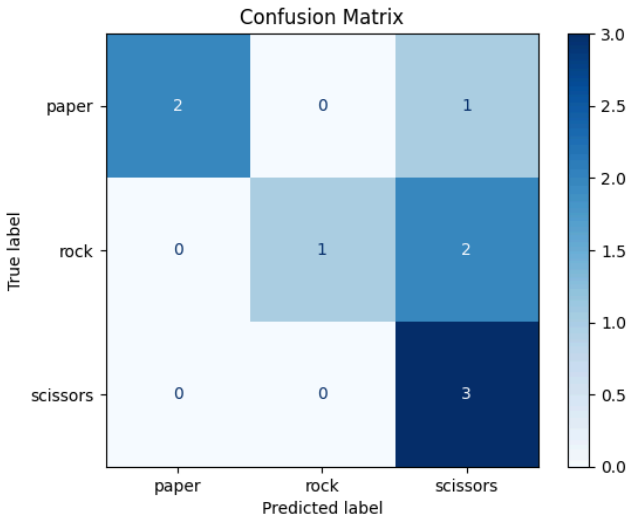


Figure 18. Confusion matrix on predictions on data with removed background

This accuracy improvement indicates that the model focuses on unimportant features of the image. Further data augmentation is needed in order to make the model indifferent to the background.

CONCLUSION

The models in this paper are simple and experimental. In order to do something more substantial, more time is needed to tune and train them. While there are incremental improvements in the models, they are still unusable for real-life tasks like a rock-paper-scissors online game.