Rock-Paper-Scissors Classification Using CNNs:

1. Introduction

1.1 Problem Statement

The goal of this project is to create and implement an image classifier that can identify hand gestures from the game "rock, paper, scissors". Since the classifier was developed from scratch, it does not rely on pre-trained models or external tools for transfer learning, instead, every step of the process (data preprocessing, model architecture, and training) is done by hand. The main challenge is to create a convolutional neural network (CNN) that is both efficient and able to generalize well to new input images.

Limiting the computational complexity and training time was one of the project's constraints. Because of this, every model was purposefully kept lightweight, which inevitably reduces its maximum accuracy. Many of the design decisions made throughout the pipeline, such as the network's structure and preprocessing strategies, were influenced by this restriction.

For the evaluation process, the models were also tested in two practical applications: a real-time webcam interface that predicts the user's hand gesture and an interactive game created with Streamlit. Despite not being the project's primary goal, these applications serve as crucial real-world tests for the models and guided several choices about generalization, latency, and robustness.

1.2 Objectives

The objective of this project is to develop and evaluate three distinct convolutional neural network (CNN) classifiers with progressively higher levels of complexity that are intended to identify hand gestures from static images. In order to examine various trade-offs between generalization, simplicity and real-world applicability, each model was developed with a specific objective in mind.

The three models' progressive design facilitates a structured comparison and encourage a broader reflection on the ways in which preprocessing and model complexity affect generalization in real-world tasks.

1.3 Constraints and Limitations

The available computational time was the primary constraint. Because of this limitation, were designed lightweight neural networks that could be trained quickly while maintaining reasonable classification performance.

The variability of real-world input conditions was another relevant limitation. It was not possible to account for every imaginable background, lighting conditions or hand poses, even though data augmentation was used to increase robustness. As a result, performance in real-time scenarios will differ a lot from the one on the validation set. It would have been easier adding to the already existing dataset totally different tipe of pictures, on different backgrounds, positions, lights: the model tried to converge to this using data augmentation: but of course, it can't have the same accuracy. To ensure a practical and repeatable workflow, these restrictions were acknowledged as a necessary component of the experimental design.

1.4 Overview of Model A, B, and C

Three models of varying complexity are used in the project: Model A, B, and C. Each of them is intended to investigate distinct trade-offs between simplicity, generalization, and robustness.

- **Model A** provides a straightforward and sound baseline. It relies on minimal preprocessing and reflects a standard supervised classification setup. Its primary functions include establishing a starting point for performance assessment and verify the classification task's basic feasibility.
- **Model B** focuses on generalization. It is intended to simulate real-life situations where the hand gesture might occur under various circumstances by altering the background and adding a moderate amount of variability. Even if some precision in the predictions on the given dataset must be sacrificed, the objective is to maintain consistent performance across a wide range of contexts, like general pictures different from the one downloaded.
- **Model C** aims to push generalization further, particularly in settings with clean or light-colored backgrounds. It uses a more aggressive preprocessing approach to increase robustness against lighting changes and subtle variations, and it is purposefully optimized for semi-ideal real-world conditions. When incorporated into real-time applications, this is expected to perform better, even though it might result in more errors on the standard test set.

2. Dataset Preparation

2.1 Original Dataset Description

Approximately 750 images per class (rock, paper, scissors) are included in the dataset; even for rare ambiguous samples, no images have been manually removed. Originally taken against a green backdrop, each image contains a single hand making a gesture. Data augmentation was used increasingly across models to enhance generalization, but always within acceptable bounds. Minor rotations, horizontal flips, and light-to-strong contrast and brightness adjustments were among the techniques used. To maintain gesture integrity, more drastic changes like cropping or zooming were avoided.

2.2 Preprocessing Overview

A common preprocessing framework was used for all three versions to guarantee uniform evaluation and comparability between models. To enable the network to receive input of uniform shape, each image was first resized to a fixed resolution of 300×200 pixels. A stratified split of the dataset was then performed to create training and test sets, with equal representation for each class in both.

To avoid overfitting and cut down on unnecessary training time, all models were trained for a maximum of 20 epochs, with early stopping based on the validation loss. Unless otherwise specified, the optimization was performed using categorical cross-entropy as the loss function and a standard Adam optimizer with default learning rate settings.

To improve robustness and replicate the variability of hand gestures in the real world, data augmentation was gradually added to each model.

2.3 Model A, B, C Preprocessing Pipeline

Model A acts as the baseline and uses the original dataset without background removal. Every image was augmented through random horizontal flips, small rotations, and brightness/contrast adjustments. To replicate slight variability, augmentations were applied in a probabilistically while maintaining the green background. After transformation, the original images were overwritten, maintaining the dataset size.

Model B enhances generalization by substituting the green background with randomly generated colours. This strategy maintains the gesture itself while simulating a variety of real-life settings. Additionally, every image goes through a random transformation, such as horizontal flip, light rotation, or contrast/brightness variation. The aim is to make the model robust to various lighting conditions and hand orientations. No images were deleted or duplicated.

Model C builds upon the B version but focuses on a particular use case: performance in well-lit environments with clean, bright backgrounds. To ensure visual coherence, the green background is eliminated and uniformly substituted with white. Data augmentation is slightly stronger.

3. CNN Architectures

3.1 Introduction to CNNs

One class of deep learning model that works especially well for image recognition applications is the Convolutional Neural Network (CNN). They use convolutional, pooling, and fully connected layers to automatically learn the spatial hierarchies of features. They are perfect for image classification tasks like hand gesture recognition because of their capacity to extract local patterns combined with parameter sharing. The CNNs in these models are complete but relatively simple, in order to keep time of the training limited.

3.2 Model A Architecture

Model A serves as a minimal baseline that is intended to verify that the task with very little complexity is feasible. The network is made up of:

- 2 convolutional layers (16 and 32 filters) with 3×3 kernels
- 2 max pooling layers (2×2)
- 1 dense layer with 32 units (ReLU)
- Output layer: softmax with 3 units for classification

Regularization and dropout are not used. All that is done is normalization (rescaling between 0 and 1). This model is predicted to overfit rapidly and has the lowest capacity. However, it trains quickly and offers a benchmark for evaluating advancements in other versions.

3.3 Model B Architecture

In order to enhance generalization, Model B adds complexity. It includes:

- 3 convolutional layers (16, 32 and 64 filters)
- 3 max pooling layers
- 1 dense layer with 64 units
- Dropout (0.4) before the final layer to reduce overfitting
- Output layer: softmax (3 classes)

To prevent overfitting for the various epochs, a light online data augmentation (differently from the one did in the preprocessing, changing in every epoch) pipeline is added to the training set, which includes brightness variation and random horizontal flips.

3.4 Model C Architecture

Model C introduces three significant changes while maintaining the framework of Model B

- Prior to training, a hyperparameter tuning step is carried out, testing three batch sizes (16, 32, and 64) over a few epochs in order to determine the optimal configuration based on validation accuracy.
- The online augmentation is more powerful than in model B: it applies random contrast changes and minor rotations in addition to flips and brightness adjustments. This enhances generalization in various lighting and pose scenarios.

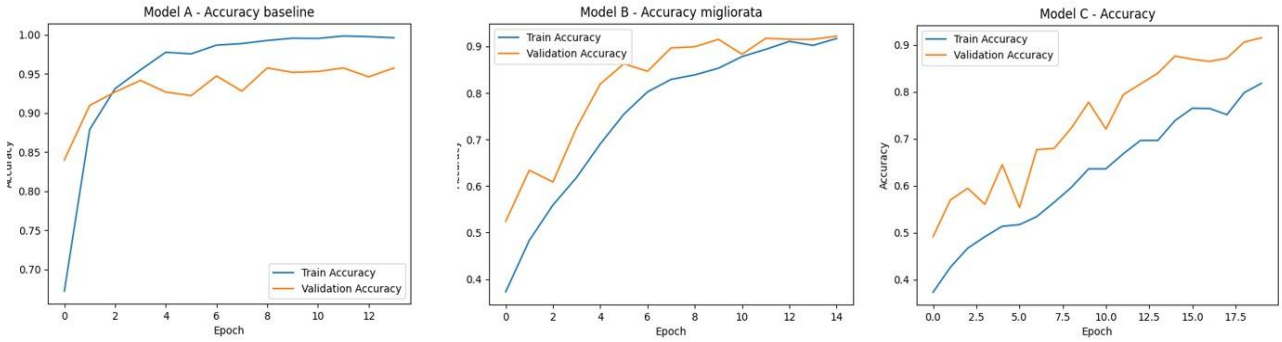
Although it is trained more meticulously and has stronger regularization and tuning, the architecture is the same as in Model B. We expect to have a weaker general accuracy due the strong modifies we made on some pictures, but to adapt better in real world condition: performing good on a light background with different type of lights.

Each model saves the errors made on the best epoch (with the lower loss on the test set), putting in a specific folder: the pictures wrongly predicted in the test set, and a csv that compare the right prediction of each picture with the given wrong one. This is useful for studying the limits of the predictors.

All models individually render a plot that describes the differences between train and validation during the different epochs, to study the overfitting/underfitting.

For each model a predictor is given as output: we can upload it in the real time predictor (realtime_predd.py) to study how well it predicts in real world.

4. Model Evaluation



4.1 Accuracy and Performance Metrics

Classification accuracy was the main metric used to assess all models on both training and validation sets. Because of the balanced dataset and clear-cut gesture classes, it is a useful metric in this situation even though it cannot fully capture all aspects of model performance. Model generalization is compared using validation accuracy, and early stopping makes sure that the results show the best-performing epoch.

4.2 Training and Validation Curves

By comparing the training and validation accuracy curves for Models A, B, and C, the above figure illustrates the learning dynamics and generalization ability of each approach:

- **Model A (left)** quickly reaches high training accuracy, almost reaching 99%, while validation accuracy plateaus around 93% with some oscillations. The model fits the training data very well, but it does not generalize correctly, this is a clear indication of overfitting. This effect is facilitated by the simpler architecture and the lack of online data augmentation.
- **Model B (center)** shows smoother and more consistent growth in both training and validation accuracy. Good generalization is suggested by the two curves' convergence at 90%. Dropout

regularization and moderate online data augmentation are used to help minimize overfitting and enable the model to learn significant features under various conditions.

- **Model C (left)** exhibits an unusual trend: the validation accuracy is consistently higher than the training accuracy. At first, this might seem positive but it might also be a sign of another problem: over-regularization brought on by excessively aggressive online data augmentation on the training set. While the validation set is still clean and simple to categorize, the model finds it more difficult to fit the data due to the high variability in the training inputs. As a result, validation accuracy increases more quickly while training accuracy increases more slowly. However, this discrepancy raises the possibility of a mismatch between the test and training conditions.

In summary, the importance of carefully tuning data augmentation strength to reflect realistic deployment conditions is highlighted by the fact that, although Model C avoids classic overfitting, it may experience **underfitting because of excessive augmentation noise**.

4.3 Error Analysis with Visual Examples

filename	true_label	predicted_label
wrong_001.jpg	scissors	paper
wrong_002.jpg	paper	scissors
wrong_003.jpg	paper	rock
wrong_004.jpg	rock	scissors
wrong_005.jpg	paper	scissors
wrong_006.jpg	scissors	paper
wrong_007.jpg	scissors	rock
wrong_008.jpg	paper	scissors
wrong_009.jpg	scissors	paper
wrong_010.jpg	paper	rock
wrong_011.jpg	paper	rock
wrong_012.jpg	scissors	paper
wrong_013.jpg	scissors	rock
wrong_014.jpg	paper	rock
wrong_015.jpg	scissors	paper
wrong_016.jpg	paper	rock
wrong_017.jpg	scissors	paper
wrong_018.jpg	scissors	paper
wrong_019.jpg	scissors	rock
wrong_020.jpg	paper	scissors
wrong_021.jpg	scissors	paper
wrong_022.jpg	rock	scissors

wrong_001.jpg	paper	scissors
wrong_002.jpg	rock	scissors
wrong_003.jpg	paper	rock
wrong_004.jpg	paper	rock
wrong_005.jpg	paper	scissors
wrong_006.jpg	paper	scissors
wrong_007.jpg	paper	scissors
wrong_008.jpg	scissors	rock
wrong_009.jpg	paper	rock
wrong_010.jpg	rock	paper
wrong_011.jpg	paper	scissors
wrong_012.jpg	scissors	paper
wrong_013.jpg	rock	scissors
wrong_014.jpg	paper	scissors
wrong_015.jpg	paper	scissors
wrong_016.jpg	rock	scissors
wrong_017.jpg	paper	scissors
wrong_018.jpg	rock	scissors
wrong_019.jpg	paper	scissors
wrong_020.jpg	scissors	paper
wrong_021.jpg	paper	scissors
wrong_022.jpg	scissors	paper

filename	true_label	predicted_label
wrong_001.jpg	paper	scissors
wrong_002.jpg	rock	scissors
wrong_003.jpg	scissors	rock
wrong_004.jpg	paper	rock
wrong_005.jpg	rock	scissors
wrong_006.jpg	rock	paper
wrong_007.jpg	paper	scissors
wrong_008.jpg	rock	paper
wrong_009.jpg	rock	scissors
wrong_010.jpg	paper	rock
wrong_011.jpg	rock	scissors
wrong_012.jpg	scissors	paper
wrong_013.jpg	paper	scissors
wrong_014.jpg	rock	paper
wrong_015.jpg	scissors	paper
wrong_016.jpg	rock	paper
wrong_017.jpg	paper	scissors
wrong_018.jpg	paper	scissors
wrong_019.jpg	paper	scissors
wrong_020.jpg	rock	scissors
wrong_021.jpg	scissors	rock
wrong_022.jpg	rock	paper

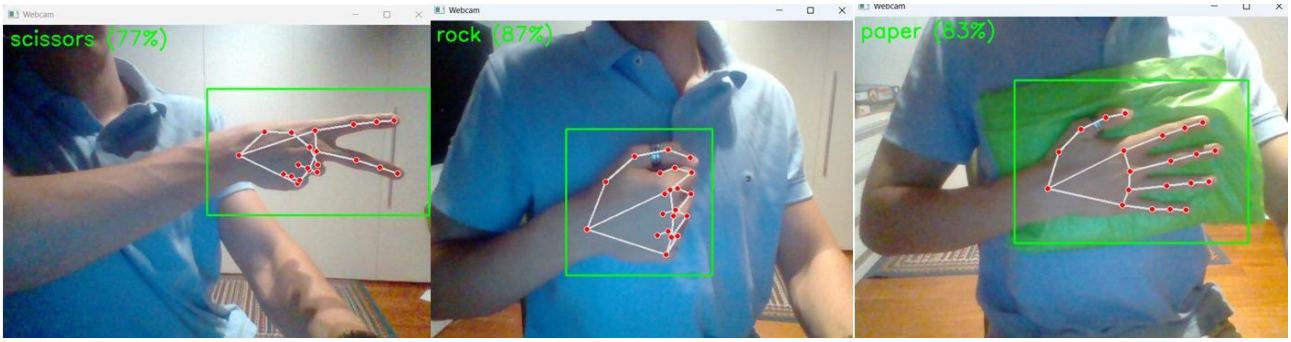
Examining the three models' validation misclassifications shows clear and recurring trends:

- Model A frequently misclassifies between scissors and paper, instead of exhibiting a dominant bias toward one class. This behaviour raises the possibility that the model is just too shallow to extract rich enough features, leading to confusion between classes that share visual similarities. Although this will not translate well to real-world data with greater variability, performance is high within the constraints of dataset, suggesting the model can handle clean, structured inputs.
- Model B does more mistakes than A: this is due to the data augmentations, but even if it perform a little bit worse on the test set than A (pictures of the validation are new for the model, but still very similar to the training one, and in optimal zoom and light conditions), it will surely perform better on new, "general conditions" pictures. Because of a deeper architecture and light data augmentation, the model seems to have better internalized class boundaries, and the misclassifications are more evenly distributed.
- Model C shows a balanced distribution of errors across all classes with no clear dominance of any label. All pairwise combinations (paper → scissors, scissors → rock, rock → paper) are misclassified. This implies that the model is more consistently challenged, which could be brought on by over-regularization or high data variability from augmentation. Although this can help in generalization, if excessive, it may also blur decision boundaries.

In general, models are working well: at least half of the incorrect images are actually difficult to predict.

5. Extra Evaluation

5.1 Real-Time Prediction

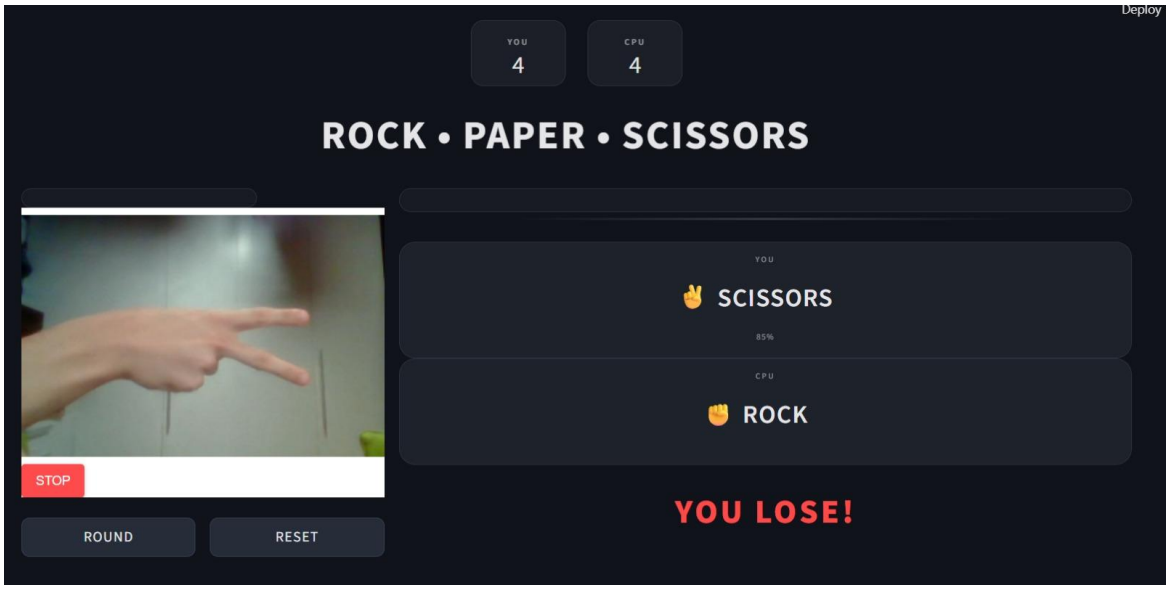


The trained models were integrated into a webcam-based prediction system to test real-world applicability. Model B performs marginally better in general scenarios because it balances data augmentation and background processing during training, even though both B and C are usable. Model A, as expected, due to high overfitting couldn't predict at all.

Both B and C are still extremely vulnerable to non-uniform backgrounds, shadows and lighting, though. Strong contrast or sudden changes in brightness can mislead the prediction but plain walls and well-lit spaces may produce accurate results.

The gestures made with the left and right hands differ slightly. The lack of systematic mirroring of each image (duplicate the whole dataset costs double the time in training) is probably the reason why the models do not fully generalize across orientations even when horizontal flips are used during training.

5.2 Streamlit Game



To turn the project into an interactive game, a straightforward **Streamlit interface** was created. Using their webcam, users compete against the model in a game of rock, paper, scissors. The user interface includes:

- A webcam feed that captures the user's gesture.
- A ROUND button that freezes the current frame and triggers the prediction.
- A CPU move randomly selected.
- Real-time display of results, prediction confidence and score tracking.

This design enhances prediction reliability because the user can modify lighting and hand position prior to classification. In this configuration, the models react with a relatively high degree of accuracy, which makes the game both functional and entertaining.

6. Conclusion

This project explored the development of a CNN-based classifier for rock-paper-scissors recognition, using three progressively complex models.

Model A demonstrated fast convergence: it quickly overfit the training data and showed poor generalization, making it unsuitable for real-world use.

Model C, although more advanced, suffered from overly strong data augmentation, which may have hindered learning by excessively altering training input, leading to underfitting.

The best balance was achieved with Model B: which maintaining a moderate augmentation strategy performed more consistently between the semi-realistic scenarios.

Despite overall good performance the simplicity of the models imposed clear limitations: In real-world applications, small changes in lighting, shadows or hand position can significantly impact accuracy.

To improve further, it would be necessary to augment the dataset with webcam-captured images under diverse conditions, tailored for this specific use case. However, within the constraints of the assignment, the models achieved respectable accuracy and generalization using only the original dataset.