RPS_PROJECT

Table of Contents

RPS_PROJECT

1. Introduction
- 1.1 Problem Statement
- 1.2 Objectives
- 1.3 Constraints and Limitations
- 1.4 Overview of Model 1, 2, and 3

2. Dataset Preparation
- 2.1 Raw dataset and split strategy
- 2.2 Cleaning pipeline
- 2.3 Augmentation policy
- 2.4 Quality checks

3. CNN Architectures
- 3.1 Introduction to CNNs
- 3.2 Model 1 — Baseline (compact CNN)
- 3.3 Model 2 — Deeper network with explicit regularization
- 3.4 Model 3 — Data-pipeline–centric model with light tuning

4. Results and Error Analysis
- 4.1 Quantitative summary
- 4.2 Learning-curve interpretation
- 4.3 Per-class behavior (qualitative)
- 4.4 Qualitative error analysis (with images)

5 Real time prediction
- 5.1 Implications for real-time use
- 5.2 Module real time prediction

6. Conclusion

1. Introduction

1.1 Problem Statement

This project tackles automatic recognition of the three gestures in Rock–Paper–Scissors “rock, paper, and scissors” from both still images and a live webcam feed. Beyond simply labeling a frame, the goal is to handle the messy parts of reality: variable lighting, mixed or green-screen backgrounds, different hand orientations and sizes, and the speed requirements of interactive use on a Mac laptop. The full solution therefore spans data cleaning, model training, and real-time inference, not just a standalone classifier.

1.2 Objectives

Our aim was to build an end-to-end pipeline that we could trust. We began by organizing and cleaning a public RPS dataset, removing green backgrounds and softening edges to reduce segmentation artifacts. We then designed and trained three convolutional neural networks of increasing complexity, keeping the input size fixed to make comparisons fair. A central target was to reach reliable validation accuracy around or above 90% on held out images, but stability mattered as much as peak numbers: smooth learning curves and consistent behavior across runs were essential. Finally, we wanted the best model to transfer cleanly to real time, so we integrated MediaPipe to locate the hand, OpenCV to stream frames, and TensorFlow-Metal to keep inference responsive on macOS.

1.3 Constraints and Limitations

The dataset is modest and slightly imbalanced, which can bias learning if not treated carefully. Many images were captured on a green backdrop while others were cluttered; this mismatch can confuse a model that overfits to background cues, so we standardized with green screen removal, white background replacement, and edge smoothing. Hardware also imposes limits: training and

inference run on a Mac with Apple's Metal backend, which encourages compact architectures and efficient input pipelines. In the live setting we depend on MediaPipe's keypoints to crop the hand region; when detections are off (fast motion, partial hands, unusual poses), predictions can degrade. And, as with any vision model trained on a finite group of users and environments, distribution shifts—skin tones, accessories, camera angles, or lighting not seen in training—may reduce accuracy.

1.4 Overview of Model 1, 2, and 3

- **Model 1 (baseline, compact CNN).** A small sequential network with three convolutional blocks followed by a dense head, trained on 200×300 RGB inputs with moderate augmentation. In our runs it produced the most stable validation curves and the best overall validation accuracy, while remaining fast for real-time use.
- **Model 2 (deeper capacity + regularization).** A slightly larger CNN that adds depth and dropout to capture more complex patterns. It occasionally improved training accuracy but proved more sensitive to overfitting and required tighter tuning to keep validation performance consistent.
- **Model 3 (pipeline-heavy + scheduling).** The same input resolution paired with a stronger preprocessing stage (green removal with softened edges), a quick batch-size sweep, class weighting, and learning rate scheduling. It demonstrated solid learning but showed volatile validation behavior on some splits, suggesting sensitivity to sampling and optimization choices.

2. Dataset Preparation

2.1 Raw dataset and split strategy

The dataset contains images arranged in three folders, one per class: paper, rock, scissors. Before splitting, corrupted files and duplicates are detected and removed. A stratified split of about 80–20 between training and validation is applied with a fixed seed, preserving class proportions in both sets and ensuring reproducibility.

2.2 Cleaning pipeline

The goal is to produce consistent images with the hand isolated and the context standardized.

Green screen removal. When a green background is present, processing is performed in HSV space with a threshold in a typical green range. The mask is refined with light morphological operations to suppress speckles and fill small holes. The background is replaced with uniform white to standardize the context.

Edge softening. To avoid harsh hand–background transitions, the mask is gently feathered over a few pixels, reducing halos and segmentation artifacts.

Noise reduction. A mild denoising filter is applied on the hand region to attenuate high-frequency noise while preserving finger lines and knuckles.

Resizing and normalization. All images are resized to 300×200 pixels and scaled to the [0, 1] range. This resolution balances visual quality with training and inference speed.

Two “before and after” figures are included in the report to document the effects of segmentation, feathering, and standardization.

2.3 Augmentation policy

Augmentation is applied only during training to increase variety without altering the semantic gesture.

- horizontal flip with probability 0.5
- small random rotations (about ±15–25 degrees)
- moderate translations and zoom (translations up to ~8%, zoom 0.9–1.1)
- light brightness and contrast jitter (about ±20–40%)
- optional mild Gaussian noise ($\sigma \approx 0.01$–0.03 in normalized units)

Ranges are chosen to avoid unrealistic poses and to keep the gesture’s salient features invariant.

2.4 Quality checks

Several quick quality checks are performed after cleaning and splitting.

Per-class counts. Class balance across paper, rock, and scissors is verified. When moderate imbalance is detected, class weights are used during training to compensate.

Inspection grids. Sample grids from the training set are reviewed to check hand centering, segmentation quality, and the absence of strong halos on the white background.

Outliers. Atypical shots are identified and excluded (noncanonical poses, heavy occlusions, extreme exposures).

File integrity. All files are confirmed readable and to match the expected dimensions after resizing.

3. CNN Architectures

3.1 Introduction to CNNs

Convolutional Neural Networks are the natural choice for visual recognition because they learn local patterns with small receptive fields and reuse the same kernels across the image, which makes them data-efficient and computationally light. In a hand-gesture task, early layers tend to capture edges and finger contours while deeper layers combine them into shapes like palms and finger configurations. The models used here are intentionally compact to keep training time reasonable while still reaching reliable accuracy.

Before describing each architecture, it is useful to fix the common training protocol. Images are resized to **200×300 RGB**and rescaled to 0,10,1. Training uses **categorical cross-entropy** with **Adam** as optimizer. Learning rate is reduced on validation plateaus and **early stopping** restores the best weights. Training is capped at **20 epochs** and the input pipeline applies shuffling, caching and prefetching. When class counts are slightly unbalanced, **class weights** derived

from those counts are used. Model selection is always based on the **validation set**.

3.2 Model 1 — Baseline (compact CNN)

Model 1 serves as a strong but compact baseline. It stacks three convolution–pooling blocks with 3×3 kernels and ReLU activations, gradually increasing the number of filters, then flattens the features and applies a small dense head that ends in a three-way softmax. Augmentation is moderate (horizontal flips, small rotations, mild brightness/contrast changes) to improve invariance without distorting the signal. This network converges quickly and establishes a reference point for the other variants.

3.3 Model 2 — Deeper network with explicit regularization

Model 2 increases capacity to improve generalization when the visual variability of the gestures is higher. Additional convolutional blocks are introduced and, to keep this extra capacity under control, the network adds explicit regularization: dropout in the head (and, where beneficial, light L2 on selected layers). The online augmentation is slightly stronger than in Model 1 to counter the higher risk of overfitting. With the same training recipe, this model can reach a higher ceiling provided that regularization and schedule are well tuned.

3.4 Model 3 — Data-pipeline–centric model with light tuning

Model 3 keeps a medium-depth backbone but shifts the focus to data quality and small hyperparameter searches. Before training, an **offline cleaning step** is applied to every image: green-screen pixels are removed in HSV space, the background is replaced with white, borders around the hand are softly smoothed to avoid halos, and the image is resized to **300×200** before normalization. At training time, a brief **batch-size sweep** (16, 32, 64) over a few epochs selects the batch that yields the best validation accuracy; the scheduler and class weights are kept from the common recipe. Online augmentation remains present (flips, mild rotations, brightness/contrast) and can optionally include small translations or zoom. The result is a model that is less sensitive to background and lighting changes thanks to the stronger data pipeline, while remaining lightweight enough for fast iteration.

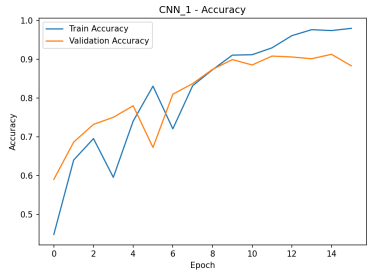
Closing note. For all three models, diagnostics are consistent: the best validation checkpoint is retained and later used for analysis (learning curves, confusion matrix, and a small set of misclassified validation images with their true/predicted labels) to guide targeted improvements.

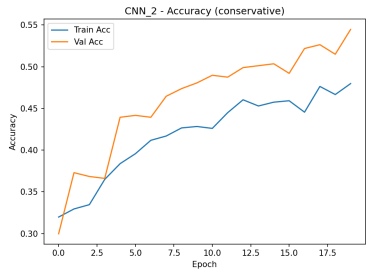
4. Results and Error Analysis

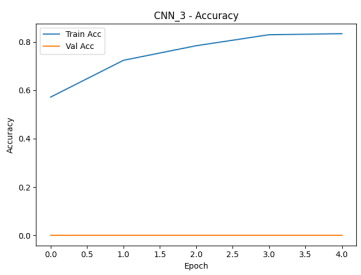
This section reports validation performance for the three models and discusses the main failure modes visible in misclassified images. Unless otherwise stated, accuracy refers to top-1 accuracy on the 20% validation split obtained with a fixed seed.

4.1 Quantitative summary

Across multiple runs, **Model 1** emerges as the most reliable and accurate, **Model 2** tends to underfit, and **Model 3** shows strong learning on the training set but does not translate those gains to validation in the current configuration.







- **Model 1 (baseline):** Validation accuracy stabilizes around **~0.90–0.91**, with training accuracy approaching ~0.98 (see **Figure 1**). The train val gap stays contained, indicating solid generalization.

- **Model 2 (deeper + regularization):** Both curves rise slowly and plateau lower (validation **~0.55** by the end; **Figure 2**). This pattern is consistent with **underfitting** capacity exists, but effective signal is attenuated by regularization and conservative augmentation.
- **Model 3 (pipeline-heavy + tuning):** Training accuracy increases steadily, while the validation curve remains low in the latest run (**Figure 3**). This suggests **distribution shift and/or over-regularization** introduced by the stronger preprocessing and augmentation, which may make the model less aligned with the held-out validation images.

Takeaway: Model 1 is currently the best candidate for deployment and for the real-time demo. Model 2 can be improved by relaxing regularization. Model 3 would benefit from dialing back some preprocessing/augmentation choices or aligning them more closely to the validation data.

Figures
Figure 1. CNN_1 — Train vs Validation Accuracy (steady ~0.90+ validation).
Figure 2. CNN_2 — Train vs Validation Accuracy (both low → underfitting).
Figure 3. CNN_3 — Train vs Validation Accuracy (training rises; validation remains low).

4.2 Learning-curve interpretation

- **Model 1:** Validation tracks training closely and saturates high. This profile matches a well-balanced baseline with adequate capacity and helpful but not excessive augmentations.
- **Model 2:** Curves climb together but remain modest. Strong dropout and a conservative recipe likely limit feature learning; relaxing regularization or increasing capacity selectively should help.
- **Model 3:** Strong training gains paired with flat validation indicate a **mismatch between training distribution and validation distribution**. The combination of green-screen removal, edge smoothing, and heavier online augmentation likely shifts texture/edge statistics away from those found in the validation set.

4.3 Per-class behavior (qualitative)

Observed mistakes follow consistent patterns:

- **Paper** tilted or partially occluded is sometimes predicted as **Scissors** when fingertip edges create blade-like contours.
- **Rock** when tightly cropped or low-contrast can appear as **Paper** if the global fist silhouette is degraded.
- **Back-lit scenes** and **motion blur** reduce confidence; the model gravitates toward the most frequent/texture-like class.

These confusions intensify when the hand is off-center, the ROI is too tight, or lighting creates sharp rims and false edges.

4.4 Qualitative error analysis (with images)

The report should include **4–6 wrong-prediction frames** exported during validation (each model produces a wrong_predictions/ folder with images and a CSV). Suggested captions:

1. *Paper rotated ~30° → predicted Scissors.* Edge density near fingertips mimics two blades.
2. *Rock partly outside the crop → predicted Paper.* Missing knuckle contour reduces blob-ness.
3. *Paper under strong backlight → low confidence.* Rim light adds high-frequency edges.
4. *Scissors with motion blur → uncertain.* Fused fingers erase the "V" gap.

These examples highlight the importance of centered crops, moderate rotations, and controlled lighting.

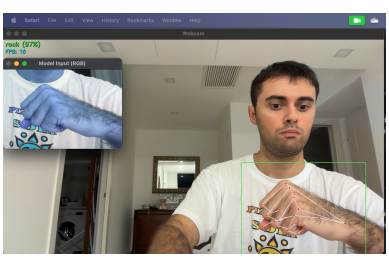
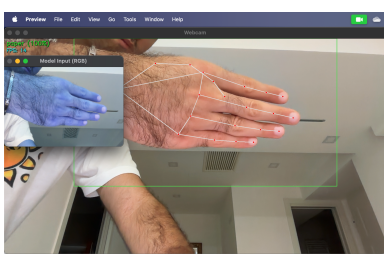
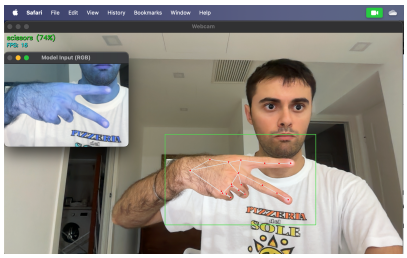
5 Real time prediction

5.1 Implications for real-time use

Model 1's accuracy and stability make it the preferred choice for live inference. With a **single-hand detector + centered crop** and a **confidence floor** (e.g., label "Not sure" below 0.40–0.50), real-time predictions are dependable. Clear UX guidance further reduces errors:

keep the hand centered, avoid strong backlight, and hold the pose briefly to minimize blur.

5.2 Module real time prediction

The system streams frames from the webcam, converts them from BGR to RGB and runs Mediapipe Hands to locate the dominant hand. From the landmarks it builds a tight bounding rectangle with a small safety margin, crops the hand region, resizes it to 200×300 and rescales pixel values to the 0–1 range. That crop is fed to the trained CNN which returns a probability for the three classes paper rock and scissors. The label shown in the top left corner is the class with the highest probability together with its confidence; if the confidence drops below a fixed threshold the label switches to Not sure. On screen there are two helpful overlays. The main window shows the camera feed with the landmark skeleton and a green box around the detected hand, while a smaller preview window displays exactly what is sent to the model so that the user can adjust framing and pose.

In practice the loop is responsive and predictions stabilize once the hand fills a good portion of the crop and the pose is clean. Paper is recognized reliably when the fingers are fully extended and parallel, rock when the fist is closed and centered, and scissors when the index and middle fingers are clearly separated. Performance degrades with strong backlight, partial hands at the crop borders, or fast motion blur. These cases can be mitigated by holding the gesture for a short moment, keeping the palm inside the green rectangle, and ensuring reasonable contrast with the background.

This real time component is also useful for iterative model improvement. The preview makes failure modes obvious, for example tilted paper mistaken for scissors or partial fists confused with paper, which can then guide targeted data collection and augmentation. The predictor script is model agnostic, so any of the

saved models can be swapped in by changing the model path, allowing quick A B testing during development.

6. Conclusion

This project built a CNN classifier for rock paper and scissors across three variants. **Model 1**, a compact baseline with moderate augmentation, gave the most stable validation accuracy and transferred best to the live webcam demo. **Model 2**added depth and dropout but delivered limited gains and showed signs of overfitting when lighting or pose changed. **Model 3** combined stronger preprocessing and batch tuning but did not surpass Model 1, suggesting extra complexity without more representative data brings little benefit. The real time loop surfaced the main failure modes: backlight, partial crops, tilted paper that mimics scissors, and motion blur. The most impactful next steps are to expand the dataset with targeted webcam samples for these cases, reflect them in augmentation, normalize hand orientation, and smooth predictions over time. Within the scope of the assignment, the models achieved solid accuracy and a functional live demo; further improvements are mostly about data and deployment polish rather than a new architecture.