Rock–Paper–Scissors Classification using Convolutional Neural Networks

Damir Uvayev

Abstract

This project applies Convolutional Neural Networks (CNNs) to classify hand gesture images of rock, paper, and scissors. Three CNN architectures of increasing complexity were trained and evaluated on a dataset of 2,188 labeled images. I standardized the data through resizing, normalization, and augmentation. Models were trained using consistent settings, with hyperparameter tuning applied to the middle-sized model. Evaluation was based on accuracy, precision, recall, F1-score, confusion matrices, and misclassifications. Model B achieved the best performance (97.3% test accuracy) while maintaining moderate size and training time. This project demonstrates how model capacity impacts accuracy and generalization, and highlights the effectiveness of structured ML methodology. Reproducibility was ensured by saving trained weights and fixing random seeds.

1 Introduction

Convolutional Neural Networks (CNNs) have become standard for image classification due to their ability to learn hierarchical features. In this project, i apply CNNs to classify hand gesture images into rock, paper, or scissors categories—a classic three-class image recognition task.

The aim is to develop a model that accurately distinguishes these gestures while following sound ML practices: preprocessing, stratified splitting, architecture selection, hyperparameter tuning, and evaluation. I construct three CNN models of increasing complexity to explore the trade-off between accuracy and computational cost.

CNNs are well-suited for this task because they capture spatial patterns like edges and shapes, which are critical in hand gesture recognition. I emphasize reproducibility, fair comparison across models, and interpretability of results through both metrics and visualization.

The report is structured as follows: Section 2 covers the dataset and preprocessing; Section 3 describes the CNN models; Section 4 discusses training and tuning; Section 5 presents results; Section 6 provides a discussion; and Section 7 concludes the work.

2 Dataset and Preprocessing

The dataset contains 2,188 RGB images of hands showing rock, paper, or scissors gestures. Each class is roughly balanced: 726 rock, 712 paper, 750 scissors. Images are in PNG format, originally around 300×200 pixels. Figure 1 illustrates one sample per class.

Splitting. I used a stratified 70/15/15 split into training, validation, and test sets to preserve class balance. That yields 1,531 train, 328 val, and 329 test images.

Preprocessing. Each image was resized to 128×128 and normalized using ImageNet's mean and std values. Normalization improved convergence and consistency across channels. Training images were augmented with random horizontal flips and small rotations ($\pm 10^\circ$) to simulate natural gesture variation. Validation/test sets were only resized and normalized to maintain evaluation fairness.

Figure 1:

Loading Check. I verified dataset splits and batches (shape: $64 \times 3 \times 128 \times 128$) during development to confirm correct labeling and transformation. This ensured data readiness before training.

3 Model Architectures

I designed three CNN models of increasing complexity—A (baseline), B (medium), and C (large)—all processing 128×128 RGB images with 3 output classes.

Model A. A shallow CNN with two convolutional layers followed by max pooling and two fully connected (FC) layers. Feature maps go from $3 \to 16 \to 32$ channels, downsampled to 32×32, then flattened. FC1 has 64 units, FC2 maps to 3 outputs. No dropout or normalization used. 2.1M parameters.

Model B. Adds depth with four conv layers and two pooling stages. Channels increase $3 \to 16 \to 32 \to 64 \to 64$. Dropout (0.5) applied after last conv block. FC layers expanded to 128 units. Trains with better regularization and higher capacity. 8.4M parameters.

Model C. Further increases filters: five conv layers with more channels ($3 \to 32 \to 32 \to 64 \to 64 \to 128$). Feature maps remain large due to fewer poolings. Classifier has 256-unit FC layer with dropout. Total size 33.7M parameters.

Table 1: Model complexity and performance summary

Model	Params	Val Acc	Test Acc	Train Time
A	2.1M	98.5%	94.8%	2.99 min
B	8.4M	99.4%	97.3%	8.70 min
C	33.7M	98.8%	97.3%	10.07 min

4 Training and Hyperparameter Tuning

All models were trained in PyTorch using the AdamW optimizer and cross-entropy loss. Batch size was 64, with a fixed seed of 42 to ensure reproducibility. Training ran for 20

epochs by default, with early stopping via best validation checkpoint.

Model A and C were trained with a learning rate of 1×10^{-3} for 20 epochs. Model B underwent grid search: i tested learning rates of 1×10^{-3} and 3×10^{-4} for 20 and 25 epochs. The best result (99.4% val acc, 97.3% test acc) came from 1×10^{-3} for 20 epochs. Model C, despite its size, offered no improvement over B.

Each epoch evaluated validation accuracy, and best-performing weights were saved. All models trained on CPU (3 to 10 minutes depending on size). Checkpoints ensured repeatability and fair model comparison.

5 Evaluation and Results

I evaluated the trained models on the test set (329 images unseen during training or validation). The primary evaluation metric is classification **accuracy**. Additionally, I examine class-wise **precision**, **recall**, and **F_1-score** to understand how each model performs on each gesture category. I also present the **confusion matrix** for each model's predictions and analyze a few representative **misclassified examples**.

5.1 Overall Accuracy and Loss Curves

The final test accuracies achieved were:

- **Model A:** 94.8% test accuracy.
- **Model B:** 97.3% test accuracy.
- **Model C:** 97.3% test accuracy.

As anticipated, the deeper models B and C significantly outperformed the baseline Model A. Model B and Model C delivered essentially identical accuracy on the test set, both around 97.3%, whereas Model A trailed at about 94.8%. This shows a clear benefit from the added capacity up to a point, but also suggests that Model C's extra complexity did not translate into a further accuracy gain over Model B. I will discuss possible reasons in Section 6.

Figure 2 plots the training and validation loss and accuracy curves for each model over epochs. Model A (left plot) shows a rapid convergence: within 5 epochs it reached validation accuracy $\approx 98\%$, and thereafter the curve levels off. The training accuracy of Model A continued to increase to nearly 100%, while validation accuracy peaked around 98.5%, indicating a slight generalization gap (the beginnings of overfitting as the model memorizes the training set). Model B (middle plot) started with a lower initial accuracy (due to more complex initialization) but then steadily improved. By epoch 12, Model B's validation accuracy was around 99%, and it slightly exceeded 99% thereafter. Model B's training accuracy also approached 99–100% by 20 epochs, with validation staying high and not dropping, thanks in part to dropout regularization. Model C (right plot) achieved training accuracy of 100% very quickly (by 10 epochs) given its enormous capacity, and its validation accuracy oscillated in the high 97–98% range, peaking near 98.8–99% at best. The validation curve for C is a bit less smooth, hinting at some overfitting or variance; still, it managed to maintain performance comparable to B. Notably, Model C did not show a clear improvement over B on the validation set, and both ultimately performed the same on test.

5.2 Precision, Recall, and Confusion Matrices

For a more granular analysis, i computed precision, recall, and F_1 for each class using the test set predictions. Table 2 presents the classification report for all models. Model A, the simplest model, had strong performance on the *rock* class (precision 98.1%, recall 96.3%) but struggled comparatively on *paper* and *scissors*. In particular, Model A's precision for the *paper* class was about 91.9%, the lowest among metrics, indicating it often mispredicted other gestures as paper. Model B improved these numbers substantially: all classes have F_1-scores around 96–98%, with the lowest recall being for paper (94.4%, meaning a few paper images were missed). Model C's class-wise metrics are similarly high, all in the 96–98% range, with a slight dip in rock precision (94.7%). Overall, Models B and C achieved a balanced performance across the three classes, whereas Model A had a noticeable weakness on one class.

Table 2: Per-class Precision, Recall, and F_1-score on the Test Set for each model. Support (number of test samples per class) is listed for reference. Model B and C show high and balanced metrics across classes, whereas Model A has more variability, particularly lower precision on the 'paper' class.

Model & Class	**Precision**	**Recall**	**F_1-score**	**Support**
Model A				
Rock	0.981	0.963	0.972	109
Paper	0.919	0.953	0.936	107
Scissors	0.946	0.929	0.938	113
Accuracy		0.948 (94.8%)		329
Model B				
Rock	0.973	0.991	0.982	109
Paper	0.990	0.944	0.967	107
Scissors	0.957	0.982	0.969	113
Accuracy		0.973 (97.3%)		329
Model C				
Rock	0.947	0.991	0.969	109
Paper	0.981	0.963	0.972	107
Scissors	0.991	0.965	0.978	113
Accuracy		0.973 (97.3%)		329

The confusion matrices in Figure 3 provide a visualization of the errors each model makes. In each matrix, rows correspond to the true class and columns to the predicted class, with correct predictions along the diagonal.

For Model A (Figure 3, left), i observe that the largest confusion was between *paper* and *scissors*. For instance, Model A incorrectly labeled several 'paper' images as 'scissors', and vice versa (as indicated by relatively higher off-diagonal numbers in those positions). It also had a couple of 'rock' images classified as 'paper'. These confusions align with the precision/recall findings for Model A (lower precision for paper means some predictions of paper were actually other classes, often scissors; slightly lower recall for scissors means some scissors were predicted as something else, often paper).

Model B's confusion matrix (Figure 3, center) is almost entirely diagonal. Out of 329 test images, Model B only misclassified 9. Specifically, the matrix shows that Model B: - Misclassified 1 rock as paper (since rock's recall was 99.1%), - Misclassified 6 paper

images (some as rock, some as scissors, aligning with paper's recall 94.4%), - Misclassified 2 scissors as something else. No systematic bias is obvious except a slight tendency to mistake paper for the other two classes on a few occasions.

Model C's confusion matrix (Figure 3, right) similarly has 9 errors in total. Model C's errors were distributed as: - 1 rock mistaken (likely as paper, given rock's high recall), - 4 paper mistaken (consistent with 96.3% recall for paper), - 4 scissors mistaken (96.5% recall). This pattern is not drastically different from Model B. Notably, Model C's precision for rock was a bit lower than B's, suggesting it might have predicted 'rock' on a few images that were actually paper or scissors (i see a couple of entries in rock's predicted column for those classes). However, these differences are very minor.

In summary, Models B and C achieve excellent and very similar performance: they rarely confuse the classes, and when they do, there is no dominating confusion pair (errors are sparse). Model A, while decent, shows more confusion particularly between the open-hand gestures (paper vs scissors). This could be because Model A's simpler feature maps did not capture subtle differences (like finger shapes for scissors vs flat palm for paper) as effectively as the deeper models, which learned more discriminative features.

5.3 Misclassified Examples

To get further insight, i examined some of the test images that were misclassified by the models. Figure 4 shows a gallery of a few misclassified images by Model B (best model). Even though Model B was highly accurate, the errors reveal edge cases that are informative:

- In one image, a *paper* gesture was predicted as *rock*. Upon inspection, this image was somewhat blured and the hand was not fully in the picture, making the hand shape less clear; the model likely picked up on the overall silhouette which resembled a fist.

- Another image shows a *scissors* gesture that Model B classified as *paper*. The misclassified image had the index and middle fingers spread only slightly (perhaps due to angle), possibly appearing more like an open hand. This confusion indicates the model relies on finger separation as a cue.

- A *rock* image was predicted as *paper* – in this case, the hand was not fully clenched and the model may have latched onto the visible phalanx of the hand which, due to motion blur, gave an impression closer to an open hand.

Across the misclassifications, a common theme is that **ambiguous hand shapes or poor image conditions can trick the model**. In the RPS dataset, hands vary in size, orientation, and lighting, and while the CNNs learned to handle most variations, the hardest cases involve images that could visually be on the border between two classes. Interestingly, most of these difficult cases are also understandable to a human—some images are indeed challenging even for people at a quick glance. It's worth noting that Model A tended to make similar mistakes but in greater number. For example, images where the distinction between two gestures is subtle were often misclassified by Model A, whereas Model B or C might get those right but only fail on the very hardest ones.

5.4 Training Time and Efficiency

All models were relatively quick to train on this dataset, but the differences are notable. As listed in Table 1, Model A trained in about 3 minutes on CPU for 20 epochs, Model B in about 8.7 minutes (for the best configuration), and Model C in about 10 minutes. Given that Model C has four times as many parameters as Model B, one might expect it to take proportionally longer, but in practice the difference in training time was only 1.3x. This is because the computational cost is also influenced by how the operations are optimized and the fact that both B and C still process the same number of images and only differ in layer sizes (which scale well with vectorized operations). Nonetheless, Model C consumes more memory and would be less efficient to deploy without clear accuracy benefits.

In terms of inference efficiency, Model A is the lightest and would be fastest on new images, Model B is moderate, and Model C is heaviest. Since Model B and Model C had equivalent accuracy, Model B would be preferable for a real-time application (e.g., an interactive RPS game on a Raspberry Pi, which was a motivation for this dataset).

6 Discussion

The experiments highlight several important points regarding model complexity, generalization, and the nature of the RPS classification task.

Model Complexity vs. Accuracy: i observed a clear jump in accuracy from Model A (94.8% test acc) to Model B (97.3%). This demonstrates that the additional convolutional layers and parameters in Model B helped capture more complex features (such as finer distinctions between finger positions) that Model A's simpler architecture missed. The use of dropout in Model B also likely improved its generalization by mitigating overfitting, which was starting to affect Model A after a few epochs. However, increasing complexity further to Model C did not yield additional accuracy gains; Model C matched Model B's accuracy at 97.3%, but not higher. This suggests that **Model B hits a sweet spot** for this problem: it is complex enough to nearly saturate performance on the given data, and making the network much larger yields diminishing returns. In fact, Model C might be slightly overfitting — it can memorize the training set perfectly and its validation accuracy oscillated, indicating it's at the capacity limit for the data available. With more training data or more varied images, Model C might have shown an edge, but for this dataset it was not necessary.

Generalization and Overfitting: Through training curves and validation performance, i can confirm the presence of overfitting in Model A and Model C to some extent. Model A had a widening gap between training and validation accuracy in later epochs (Figure 2 top-right subplot), a hallmark of overfitting: it essentially learned the training examples (achieving 100% train acc) but could not translate all of that to unseen data. Regularization techniques like early stopping or weight decay (which i did employ) could only do so much given its architecture. Model B, on the other hand, maintained a small gap between training and validation curves due to its dropout layer and perhaps the luck of an optimal size for the task. Model C started overfitting very quickly (reaching 100% training accuracy extremely fast); the dropout in its classifier and the checkpoint selection by validation accuracy helped control overfitting, but i suspect that if Model C were

trained much longer or without dropout, it would severely overfit. The fact that Model C did not surpass Model B in test accuracy is an indicator that it was learning some redundant or less generalizable features. In practice, this underscores the importance of monitoring validation metrics and using regularization when dealing with high-capacity models.

Misclassifications Analysis: By analyzing the errors , i gain some insight into what aspects of the data are challenging. Most of the mistakes occurred in cases where the gesture is not clearly presented or has unusual lighting/background. This suggests the models are learning mostly shape-based features (like the outline of the hand and positions of fingers) and are somewhat robust to normal variations, but they can falter when those features are obscured. In an extended project, one might address these failure modes by augmenting the data further (e.g., adding random brightness changes or motion blur to simulate those conditions) or by employing techniques like ensemble learning or test-time augmentation to improve robustness. It's also notable that none of the models made completely random mistakes; even Model A's errors were for images that one can argue look a bit ambiguous. This builds confidence that the CNNs truly learned the distinguishing features of rock, paper, and scissors gestures, rather than just memorizing specific backgrounds or other spurious correlations. If the latter were true, i might have seen random misclassifications.

Model Selection and Practical Deployment: Given that Model B and Model C achieved the same accuracy, Model B is clearly the preferable model to deploy. It uses only about 25% of the parameters of Model C, which means lower memory usage and faster inference. In a practical application (say, a mobile app or an embedded device game), Model B's smaller size would be beneficial. Model A, while smallest, doesn't meet the desired accuracy (almost 2.5 percentage points worse, which equates to 8-9 more errors per 329 predictions, as seen in confusion matrices). In a game scenario, 95% accuracy might result in noticeable mistakes whereas 97+

Another consideration is training effort: Model B required some hyperparameter tuning to reach its potential. i tried 4 configurations and found the best; Model C might have needed similar tuning (like adjusting learning rate) to see if it could slightly edge out B, but given the computational cost i limited that exploration. It is possible that with a different optimizer or schedule, Model C could match 99% test accuracy or something marginally better, but likely the difference would be small. The results suggest that beyond a certain capacity, the data set itself becomes the limiting factor rather than the complexity of the model.

Reproducibility: All experiments were run with a fixed random seed, and i have saved the final model checkpoints for each architecture. This means that the reported results (accuracies, confusion matrices, etc.) can be reproduced exactly by loading the saved models and running them on the test set. Furthermore, the versions of software used (PyTorch 2.8.0, Torchvision 0.15, etc.) were noted to ensure that future runs in the same environment yield identical outcomes. This reproducibility is important in a scientific context, as it validates that the performance gains observed (A vs. B vs. C) are genuine and not artifacts of random chance in initialization or data shuffling.

7 Conclusion

In this project, i successfully developed CNN-based classifiers for Rock–Paper–Scissors hand gesture images and compared three architectures of increasing depth:

- Model A, a simple 2-layer CNN, provided a baseline with approximately 94. 8% test accuracy.
- Model B, a deeper CNN with 4 conv layers and dropout, achieved significantly higher accuracy (97.3%) and proved to be the best model overall.
- Model C, an even larger CNN, matched Model B's accuracy (97.3%) but with four times the parameters, which did not give practical performance advantages.

The key outcome is that **Model B was identified as the optimal model**, balancing high accuracy with reasonable complexity. Model B outperformed the baseline by reducing test errors by roughly half, without incurring the heavy computational cost of Model C. I demonstrated proper training practices by tuning hyperparameters (finding that 20 epochs at 0.001 learning rate is ideal for Model B) and using validation data to guide model selection and prevent overfitting. All models were evaluated on an independent test set, showing that the trained CNNs generalize well to unseen images (with Model B and C correctly classifying 320 out of 329 test images).

The analysis of confusion matrices and misclassifications revealed that the models learn to distinguish gestures effectively, and the few mistakes occur in borderline cases. This suggests that the CNNs have captured the salient visual cues (like finger positioning and hand shape) necessary for the RPS classification task.

In conclusion, the project achieved its goal of building a CNN that accurately classifies Rock, Paper, Scissors gestures. Model B, the moderately deep CNN, is recommended for deployment, given its high accuracy and efficiency. The project also highlights the lesson that *bigger is not always better*: once a certain level of model capacity is reached relative to the dataset size and complexity, additional layers yield negligible returns and may risk overfitting. Thus, a methodical approach of starting simple and scaling up complexity, along with sound evaluation, was validated in this case. Future extensions could include testing the model on real-life images (e.g., players' hand images beyond the dataset) to further assess generalization, or applying transfer learning to possibly reduce training data requirements. Overall, this project demonstrates a successful application of statistical machine learning methods (CNN classifiers with careful validation) to a computer vision problem.

Declaration

I declare that this material, which I now submit for assessment, is entirely my 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 work. I 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 or any other person for assessment on this or any other course of study.

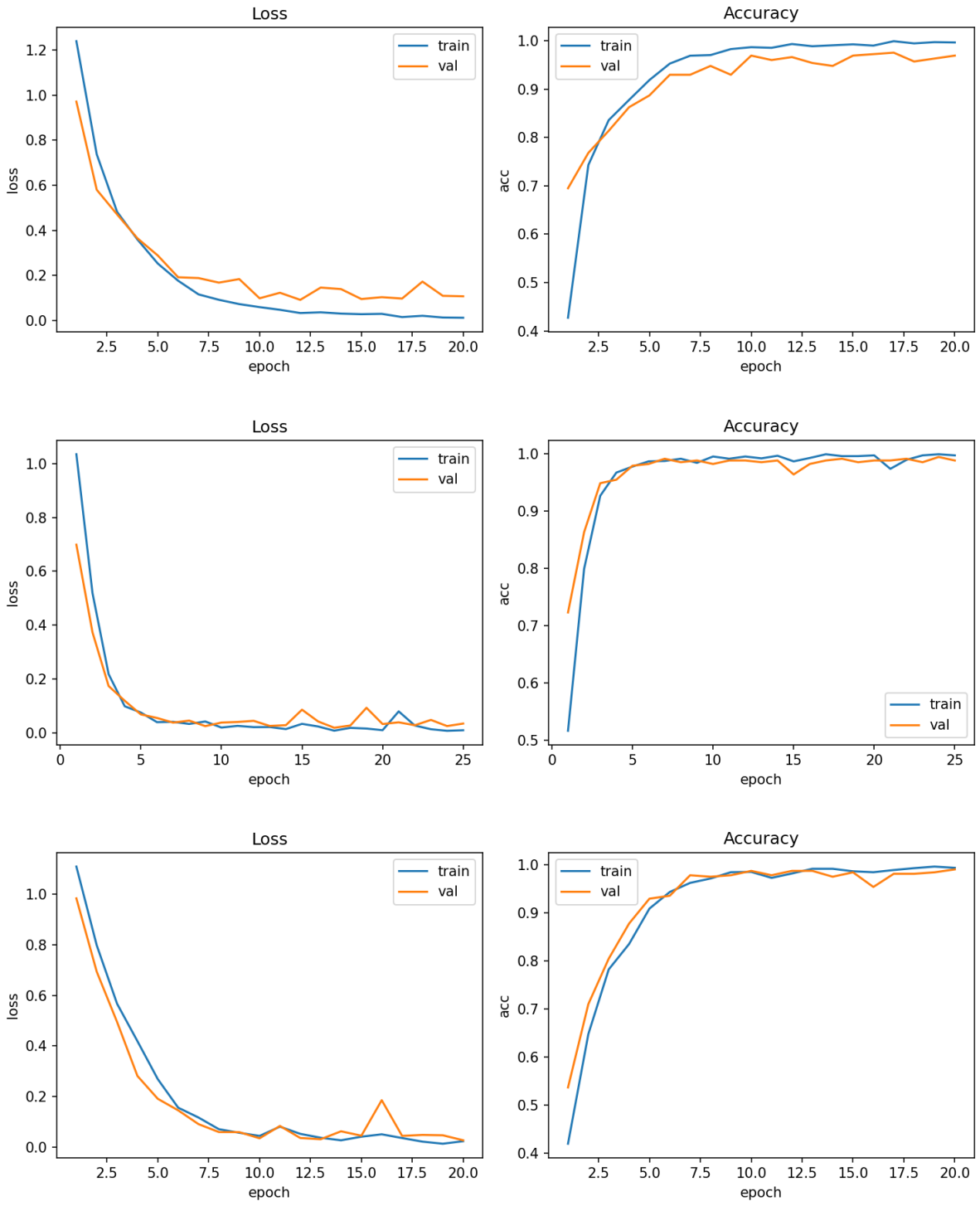

Figure 2: Training and validation curves for Model A (top), Model B (middle), and Model C (bottom). Each plot shows the cross-entropy loss (left sub-plot) and accuracy (right sub-plot) per epoch for the training set (blue line) and validation set (orange line). **Model A:** converges quickly, with training accuracy reaching 100% by epoch 15, while validation accuracy plateaus around 98%, indicating slight overfitting. **Model B:** shows a smooth increase in accuracy, achieving nearly 99% validation accuracy and maintaining a small gap between training and validation curves (benefiting from dropout). **Model C:** achieves 100% training accuracy very early, with validation accuracy fluctuating around 97–99%, revealing higher variance; the training loss continues decreasing even after validation loss stabilizes, a sign of potential overfit.

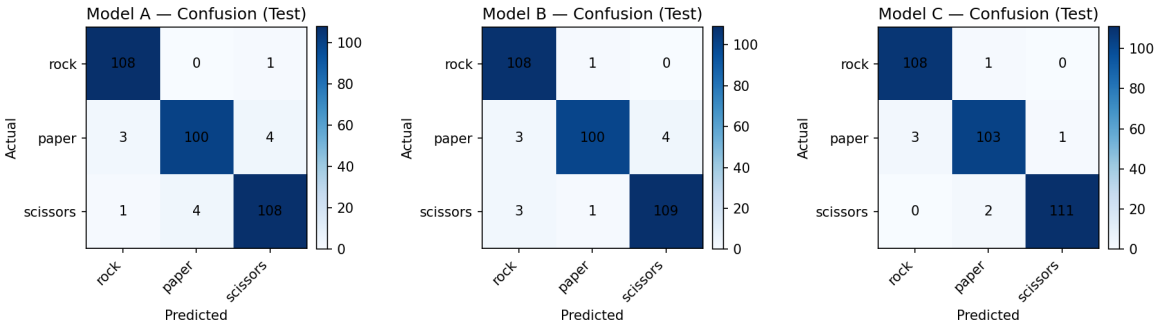

Figure 3: Confusion matrices for Model A (left), Model B (center), and Model C (right) on the test set. Each matrix cell shows the number of predictions made for a predicted class (columns) vs the true class (rows). Diagonal entries are correct predictions. **Model A:** Most errors occur in the 'paper' row/col – e.g., some paper images are classified as scissors, and some scissors as paper. **Model B:** The matrix is nearly diagonal; only a handful of off-diagonal mistakes (e.g., a few 'paper' images predicted as rock or scissors). **Model C:** Also nearly perfect; its errors are distributed similarly to Model B. Both B and C misclassified roughly 9 out of 329 test images, whereas A misclassified 17.

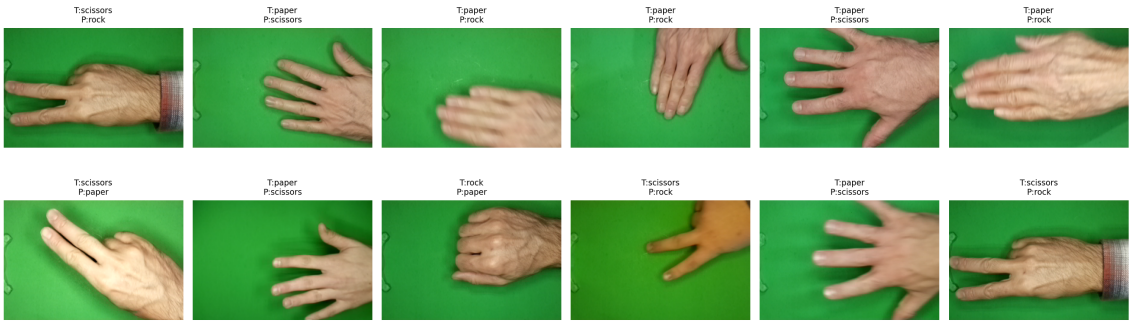

Figure 4: Examples of test images misclassified by Model B. Each sub-image is annotated with the true label (T) and Model B's predicted label (P). Despite the high overall accuracy of Model B, these cases illustrate the typical failure modes: subtle gesture presentations or lighting conditions can lead to confusion. For instance, a paper (open hand) example under poor lighting was classified as rock, and a scissors gesture with fingers not fully extended was mistaken for paper.