Evaluating Models for Gesture Recognition

Luke Corcoran

G00410404

1 Introduction

This paper aims to evaluate **convolutional neural network (CNN)** models for image-based gesture recognition using the Rock-Paper-Scissors dataset. It follows a structured approach, beginning with an overview of the dataset, its distribution, and preprocessing steps used to facilitate successful model training. Several CNN architectures are developed and assessed, including a basic CNN, a deeper CNN, and regularized variants. Their performance is compared using generalization, accuracy, loss and speed metrics to identify the strongest configuration. Following this, the impact of image resolution and colour space on model performance is examined. Transfer learning is also explored using pretrained ResNet50 and VGG16 models. The final model is evaluated on unseen test data, with its performance analyzed using a classification report and confusion matrix. The document then concludes with a final summary of the evaluation outcomes and core insights.

2 Dataset and Preprocessing

The Rock-Paper-Scissors dataset consisted of hand gesture images representing the three standard game gestures; rock, paper and scissors. The original dataset was structured with predefined training, validation, and test sets, but was reorganized for this project to balance data distribution.

2.1 Data Extraction and Distribution

The dataset containing images of hand gestures was organized in three distinct classes for rock, paper, and scissors. As shown in Figure 1, the original dataset was unevenly distributed, with 2,520 training images, 372 test images, and 33 validation images. This distribution was problematic due to the significant imbalance between the sets, with the validation set being particularly small (only 33 images), which is inadequate for reliable model evaluation [1]. When combined, this resulted in a total dataset of 2,925 images distributed across the three gesture classes. The data was reorganized using a **70-10-20** split into training (70%), validation (10%), and test (20%) sets, with a fixed seed value used for dataset shuffling, guaranteeing that the splits would remain consistent across multiple runs.

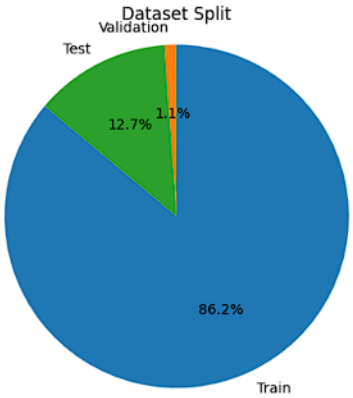

Figure 1: Original distribution of the dataset.

This balanced distribution provided sufficient data for training while reserving a portion for validation and testing, allowing for reliable training and accurate performance measurement of the trained models [1]. To improve training efficiency, the datasets were prefetched using a dynamic buffer size. As noted by Park (2021), this allows for reduced input delays by maintaining a steady flow of data to the model [2].

2.2 Image Preprocessing

All images were rescaled to a size of 80×80 pixels with three colour channels (RGB). This resolution was selected to balance visual clarity while minimizing computational demands during model training. The images were organized into

batches of 32, a batch size that Kandel & Castelli (2020) note offers a balance between computational efficiency and model convergence rates [3]. The dataset was visualized to confirm that the images were appropriately labeled and contained distinguishable features suitable for classification tasks.
For experiments involving different image sizes (50×50 and 250×250) and colour spaces (grayscale), separate data pipelines were created with the following transformations:

- **For smaller images (50×50),** the dataset was processed by downscaling the original images.
- **For larger images (250×250),** the dataset was upscaled to provide more detailed representations.
- **For grayscale experiments,** the original RGB images were converted to grayscale, reducing the channel dimension from 3 to 1.

These preprocessing steps ensured that all models received correctly formatted inputs while allowing for the investigation of the impact of image size and colour space on model performance.

3 CNN Architecture Development

In order to determine the best CNN architecture for classifying the Rock-Paper-Scissors dataset, an initial model was constructed using a basic architecture. This model was then compared with a deeper CNN containing additional layers, to evaluate which structure offered better generalization, classification accuracy, and loss performance, as well as faster training and inference times. Once an architecture depth was selected based on these criteria, additional CNN architecture variants - such as versions with data augmentation, dropout, and L2 regularization - were investigated to assess their impact on model performance. The best-performing model was subsequently used in further experiments involving changes in image size and colour.

3.1 Basic CNN Architecture

The first CNN developed served as a baseline for comparison with more complex architectures. It was implemented using a sequential model and consisted of the following layers:

- **Input Layer**: Accepts images of shape 80x80x3, representing resized RGB input.
- **Rescaling Layer**: Normalizes pixel values by dividing by 255, mapping them to the [0, 1] range.
- **Convolutional Layer 1**: Applies 16 filters of size 3x3 with ReLU activation and 'same' padding.
- **Max Pooling Layer 1**: Reduces spatial dimensions using 2x2 max pooling.
- **Convolutional Layer 2**: Uses 32 filters of size 3x3 with ReLU activation and 'same' padding.
- **Max Pooling Layer 2**: Further reduces spatial resolution using 2x2 max pooling.
- **Flatten Layer**: Flattens the 2D feature maps into a 1D vector.
- **Dense Layer**: Fully connected layer with 128 units and ReLU activation.
- **Output Layer**: Dense layer with 3 units (one per class: rock, paper, scissors), with logits output.

The model was trained using a gradient-based optimization algorithm with a low learning rate and employed a loss function suitable for categorical classification using raw class scores rather than probabilities. Early stopping and validation-based checkpointing were integrated during model training, to prevent overfitting and retain the best model weights. Although the final layer of the model did not apply a softmax activation, the chosen loss function internally incorporates this step, contributing to improved numerical stability [4].

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 80, 80, 3)	0
conv2d (Conv2D)	(None, 80, 80, 16)	448
max_pooling2d (MaxPooling2D)	(None, 40, 40, 16)	0
conv2d_1 (Conv2D)	(None, 40, 40, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 20, 20, 32)	0
flatten (Flatten)	(None, 12800)	0
dense (Dense)	(None, 128)	1,638,528
dense_1 (Dense)	(None, 3)	387

Figure 2: Structure of initial CNN.

3.2 Deeper CNN Architecture

To investigate the impact of increased depth on model performance, additional layers were added to the baseline architecture. The deeper model introduced the following enhancements:

- **Three convolutional layers** were used instead of two, with progressively larger filter counts: 32, 64, and 128.
- Each convolutional block was followed by a **max pooling layer,** adding an additional pooling stage compared to the baseline.
- The fully connected section was expanded to include **three dense layers** with 512, 256, and 128 units respectively, all using ReLU activation.

Despite its added complexity, the deeper CNN outperformed the baseline CNN. It achieved a lower validation loss (**0.0142** vs. **0.0405**) and nearly matched the baseline in validation accuracy (**99.66%** vs. **100%**), while also having a shorter overall training time (**59.25 seconds** vs. **74.57 seconds**). Although the deeper model had a slightly higher average inference time (**686.4 ms** vs. **681.1 ms**), this was negligible compared to the performance gains. The accuracy curves in Figure 3 show that while both models exhibit strong generalization, the deeper model demonstrates faster convergence. Based on these results, the structure of the deeper CNN was selected for use in subsequent architecture and image variations.

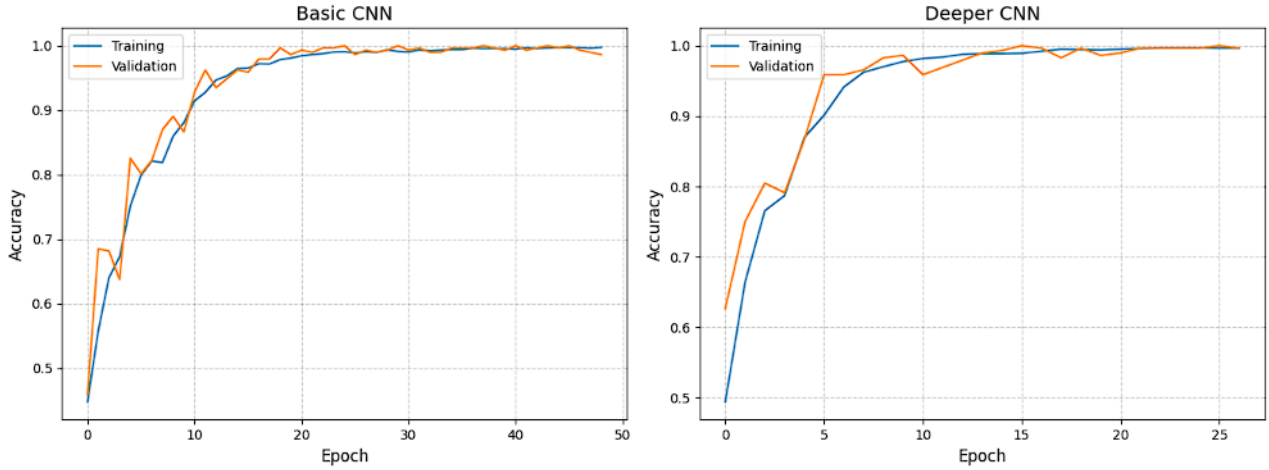

Figure 3: Comparison of accuracy graphs between the standard and deeper CNN models.

3.3 Tested Regularization Strategies

Several regularization techniques were tested to evaluate their impact on the model's generalization and performance. These included data augmentation, L2 regularization, and dropout.

Data augmentation was implemented by applying random horizontal flips and rotations to the training images in an effort to expand the dataset and reduce any minor overfitting. This model achieved a validation accuracy of **100.00%** with a validation loss of **0.0292**, and an inference time of **672.19 ms**. However, the training accuracy continued to increase while validation accuracy plateaued, suggesting some degree of overfitting. Its training time was **58.30 seconds**, making it one of the faster models.

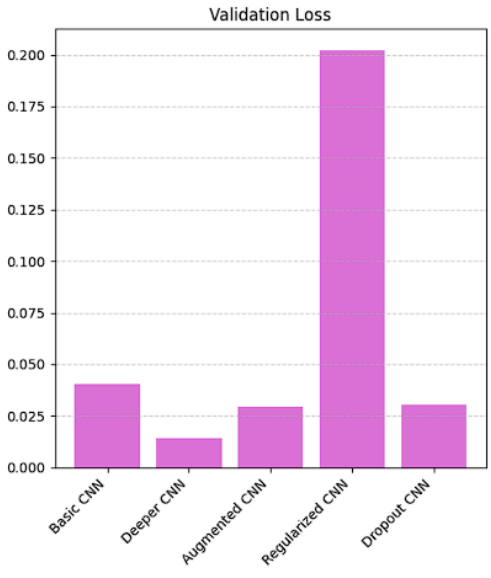

Figure 4: Comparison of validation loss between all models.

L2 regularization added a penalty to the weights of the dense layers to encourage smaller weight values. This model had the highest validation loss of all at **0.2024**, with a validation accuracy of **99.31%** and the slowest inference time at **695.81 ms**, indicating possible underfitting and inefficiency. Its training time was **53.40 seconds**.

Dropout was applied by randomly deactivating 30% of dense layer neurons during training. It produced consistent results with a validation accuracy of **99.66%**, a low validation loss of **0.0304**, and the fastest training time at **50.90 seconds**. Despite this, it did not outperform the deeper CNN overall.

As shown in Figure 4, the deeper CNN recorded the lowest validation loss among all models by a large margin (**0.0142**), reflecting strong generalization. It also maintained a similar validation accuracy, inference time and training time to the regularized models, justifying its selection for subsequent experimentation.

4 Impact of Image Size and Colour Space

4.1 Image Size Variation

The effect of image resolution was analyzed by training the model with smaller (50x50) and enlarged (250x250) inputs. The model trained on 250x250 images achieved validation accuracy of **100.00%**, but also required the longest training time at **166.19 seconds** and a higher inference time of **919.36 ms**. In contrast, the small 50x50 model had the fastest training time at **49.63 seconds** but slightly lower performance, with a validation accuracy of **99.66%** and loss of **0.0463**.

The 80x80 model (the deeper CNN used in previous experiments) offered the best performance of the three, achieving the same validation accuracy as the 50x50 model but with a much lower validation loss of **0.0142** and the lowest inference time of **765.22 ms**. These results show that increasing input resolution has adverse negative affects on validation loss, training and inference times, while decreasing input resolution leads to an improvement in training time, but at the cost of inference time and validation loss. The 80x80 configuration was therefore retained due to its superior performance metrics in comparison to the other configurations.

4.2 RGB vs. Grayscale Comparison

To evaluate the impact of colour information on model performance, RGB images were converted to grayscale, reducing the number of input channels from three to one. The grayscale model achieved a validation accuracy of **100.00%**, slightly outperforming the RGB model's **99.66%**. It also trained slightly faster (**56.18 seconds** vs. **59.25 seconds**) and had a marginally lower inference time (**704.60 ms** vs. **744.91 ms**). Despite this, its validation loss was significantly higher at **0.0268** compared to the RGB model's **0.0142**, suggesting that its predictions were slightly less confident.

These results indicate that while grayscale input offers slight improvements in speed and accuracy, the colour information preserved in RGB images contributes to improved generalization. For this reason, the RGB configuration was retained.

5 Transfer Learning Approaches

Transfer learning was explored using two CNN architectures pretrained on ImageNet: **ResNet50** and **VGG16**. Adapting these models to the dataset involved removing their top layers and appending classification layers suited for the rock-paper-scissors task. Both models were frozen during training to preserve the learned representations, allowing only the appended classification layers to be trained.

5.1 ResNet50 Implementation

The **ResNet50** model was integrated with a global average pooling layer and a dense output layer with three units. The backbone was frozen to retain pretrained features. The model reached a validation accuracy of **98.97%**, with a validation loss of **0.1141** and a training loss of **0.1284**. The accuracy and loss curves showed parallel trends for training and validation sets, suggesting strong generalization with minimal overfitting/underfitting. The total training time was **89.44 seconds**, while inference time was **699.11 ms**.

5.2 VGG16 Implementation

The **VGG16** model included a preprocessing layer to match the format expected by the pretrained weights. It was integrated with a global average pooling layer and two dense layers - one with ReLU activation and the other with three output units (for each class). Initial training showed better results than the ResNet50 model, but further improvement was observed after applying a 1 epoch fine tune.

The fine-tuned **VGG16** model achieved the best overall performance. It reached a perfect validation accuracy of **100.00%** and the lowest validation loss at **0.0034**. Despite having the longest training time at **114.10 seconds**, it delivered the fastest inference time at **672.05 ms**. These metrics indicate that VGG16, although more computationally intensive to train, provided the best performance, despite displaying slight overfitting.

Despite this, as shown in Figure 5, the significantly shorter training time of the **Deeper CNN** model (**59.25 seconds**) made it the more practical choice for deployment, balancing high performance with training speed. For this reason, the **Deeper CNN** was selected as the final model.

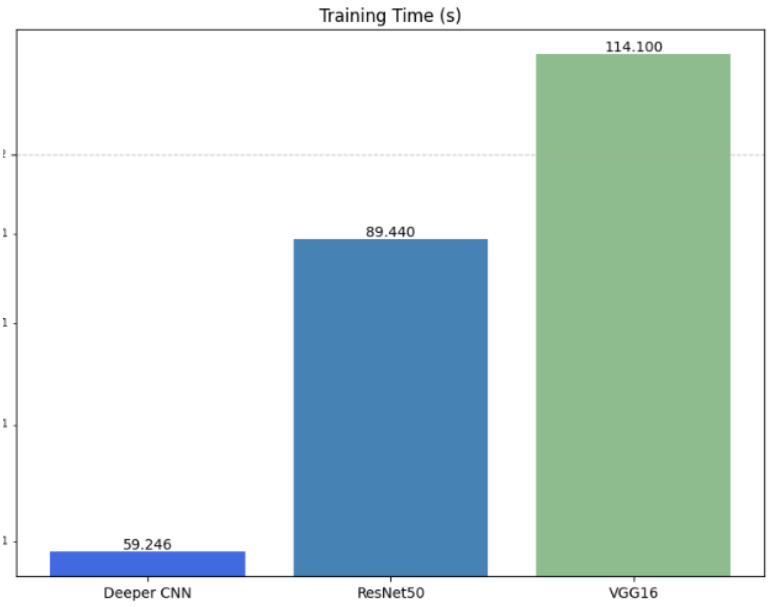

Figure 5: Comparison of training time between transfer learning and deeper CNN models.

6 Model Evaluation

The Deeper CNN model was evaluated using the unseen test set to validate its performance. It achieved a final test accuracy of **95.39%** and a test loss of **0.1428,** confirming strong generalization. A softmax activation layer was appended to the final model to change its output format from raw logits to probability-based predictions, allowing it to return confidence scores for each class during inference. When tested on four individual images, the model correctly classified three, with the misclassification occurring on a "rock" gesture predicted as "scissors". This result highlights that although the model performs reliably overall, occasional misclassifications can arise in certain cases.

Classification Report:

	precision	recall	f1-score	support
paper	0.98	0.88	0.93	191
rock	0.93	0.98	0.95	196
scissors	0.97	1.00	0.98	199
accuracy			0.96	586
macro avg	0.96	0.95	0.95	586
weighted avg	0.96	0.96	0.96	586

Figure 6: Classification report.

The classification report, as shown in Figure 6, highlights the high level of performance of the final model, with an overall accuracy of **96%.** The report shows strong precision, recall, and F1-scores across all three classes. The lowest F1-score was **93%** for the “paper” class, which still represents strong performance. The macro and weighted averages were both **95%** or higher, confirming that the model generalizes well without signs of overfitting or underfitting.

The confusion matrix shown in Figure 7 illustrates that most of the test inputs were correctly classified. The majority of predictions aligned exactly with their true labels, and no “scissors” inputs were misclassified. The only minor misclassifications occurred between the “paper” and “rock” classes, suggesting the model occasionally struggles to distinguish between these visually similar gestures.

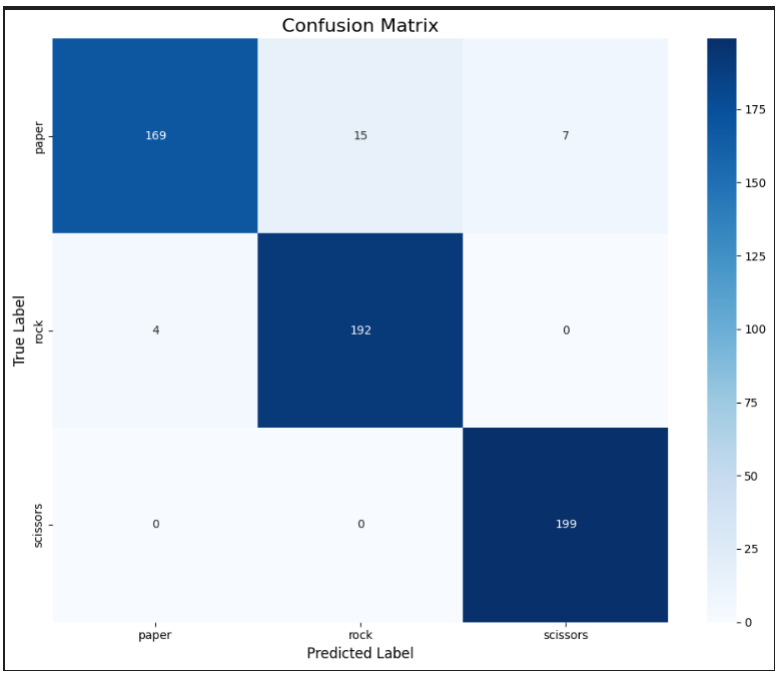

Figure 7: Confusion Matrix

7 Conclusion

This study explored and compared a range of CNN architectures and transfer learning approaches for the task of image-based gesture recognition. Models were evaluated using various metrics, including generalization, accuracy, validation loss, training time, and inference performance. While transfer learning with VGG16 achieved the best overall accuracy and lowest validation loss, the deeper CNN model was selected as the final model due to its favourable trade-off between performance and speed.

Upon evaluation on the unseen test set, the deeper CNN achieved a high test accuracy with minimal loss, confirming strong generalization. The model demonstrated fast inference and no signs of overfitting or underfitting, making it suitable for real-world applications. Overall, the findings within this study highlight the importance of careful model selection and regularization in building accurate and effective gesture recognition systems.

References

[1] Samet Oymak, Mingchen Li, and Mahdi Soltanolkotabi. Generalization Guarantees for Neural Architecture Search withTrain-Validation Split. URL: https://proceedings.mlr.press/v139/oymak21a/oymak21a.pdf.

[2] Jungkyu Park. Data prefetching in deep learning, 2021. [Online], accessed 09-May-2025. URL: https://www.jpatrickpark.com/post/prefetcher/.

[3] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6(4):312–315, December 2020. URL: https://linkinghub.elsevier.com/retrieve/pii/S2405959519303455, https://doi.org/10.1016/j.icte.2020.04.010 doi:10.1016/j.icte.2020.04.010.

[4] TensorFlow – Quickstart for beginners, 2025. Online, accessed 21-April-2025. URL: https://www.tensorflow.org/tutorials/quickstart/beginner.