An Empirical Study of Convolutional Neural Networks for Hand Gesture Recognition

Alex Tulip

Abstract

This paper presents a systematic approach to developing a Convolutional Neural Network (CNN) for the classification of Rock-Paper-Scissors hand gestures. Adhering to a rigorous machine learning methodology, I explore the impact of architectural complexity and regularization techniques on model performance. I design three models of increasing complexity, starting with a simple baseline and progressively introducing dropout and data augmentation. My findings demonstrate a clear trade-off between model complexity and generalization, where the initial model exhibits significant overfitting. Regularization techniques, particularly data augmentation, prove highly effective in mitigating this issue, leading to a final model that achieves 97.7% accuracy on an unseen test set. A final generalization test on custom, out-of-distribution images reveals the model's sensitivity to domain shift, a critical consideration for real-world deployment. This study highlights the practical application of core statistical learning concepts, including the bias-variance trade-off, risk estimation, and the importance of a structured experimental process.

1 Introduction

The task of image classification is a cornerstone of modern machine learning, with Convolutional Neural Networks (CNNs) representing the state-of-the-art approach for a wide array of visual recognition problems. This project undertakes the classification of the Rock-Paper-Scissors dataset, a collection of images depicting the three distinct hand gestures. The primary objective is not merely to maximize classification accuracy, but to conduct a methodical investigation into the process of building a robust CNN, grounded in the principles of statistical learning theory.

As outlined in the course notes [1], the development of a successful learning algorithm involves navigating the fundamental bias-variance trade-off. A model that is too simple may underfit the data (high bias), failing to capture the underlying patterns. Conversely, a model that is too complex may overfit (high variance), memorizing the training data, including its noise, and failing to generalize to unseen examples. This project explicitly explores this trade-off by designing, training, and evaluating three distinct CNN architectures of incremental complexity.

My investigation is guided by the following research questions:

1. How does increasing model complexity affect the balance between approximation error (bias) and estimation error (variance) in a practical image classification task?
2. To what extent can standard regularization techniques, namely Dropout and data augmentation, mitigate overfitting and improve a model's generalization performance?
3. How robust is a highly accurate model, trained and tested on a specific data distribution, when subjected to a domain shift via evaluation on novel, out-of-distribution data?

I will analyze the performance of each model through its training and validation curves, using concepts of regularization to control model complexity and improve generalization. The final, optimized model is then evaluated on a held-out test set to provide an unbiased estimate of its true statistical risk.

2 Methods

My methodology was designed to be systematic and reproducible, adhering to sound machine learning practices at every stage. The entire project was implemented in Python 3 using the TensorFlow and Keras libraries, with experiments run on a T4 GPU provided by the Google Colab environment.

2.1 Dataset and Preprocessing

I utilized the Rock-Paper-Scissors dataset from Kaggle [2], which contains 2,188 images of hand gestures, evenly distributed across the three classes: 'rock', 'paper', and 'scissors'. The images are 300x200 pixels with varied lighting and backgrounds.

The dataset was first partitioned into three distinct, non-overlapping subsets: a training set (60%), a validation set (20%), and a test set (20%). This partitioning is crucial to prevent data leakage; the validation set is used for model selection and hyperparameter tuning, while the test set is reserved for a single, final evaluation of the chosen model. The resulting dataset sizes are summarized in Table 1.

Table 1 Dataset split counts

Set	Rock	Paper	Scissors	Total
Training	435	427	450	1312
Validation	145	142	150	437
Test	146	141	152	439

All images were resized to 150x150 pixels and their pixel values were normalized from the [0, 255] range to [0, 1] by scaling with a factor of 1/255. This normalization step is standard practice for stabilizing the training process of neural networks.

2.2 Model Architectures

I designed three CNN architectures with incrementally increasing complexity to study the effects on model performance and generalization.

2.2.1 Model 1: Baseline CNN

My first model was a simple, shallow CNN designed to establish a performance baseline and observe the behavior of a model with a relatively small hypothesis space. Its architecture consists of a single convolutional block followed by a dense layer.

- `Conv2D` (16 filters, 3x3 kernel, ReLU activation)
- `MaxPooling2D` (2x2 pool size)
- `Flatten`
- `Dense` (64 units, ReLU activation)
- `Dense` (3 units, Softmax activation for output)

The small number of filters (16) was chosen to limit the model's capacity, making it more likely to exhibit clear learning dynamics without being overly simplistic.

2.2.2 Model 2: Intermediate CNN with Dropout

To address the issues observed in the baseline, the second model was made deeper and introduced regularization. The filter count was increased in successive layers (32 then 64) to allow the network to learn a hierarchy of features, from simple edges to more complex shapes.

- `Conv2D` (32 filters, 3x3 kernel, ReLU activation)
- `MaxPooling2D` (2x2 pool size)
- `Conv2D` (64 filters, 3x3 kernel, ReLU activation)
- `MaxPooling2D` (2x2 pool size)
- `Flatten`
- `Dropout` (rate of 0.5)
- `Dense` (128 units, ReLU activation)
- `Dense` (3 units, Softmax activation)

The addition of a `Dropout` layer with a rate of 0.5 is a key regularization technique. It is designed to prevent overfitting by randomly setting half of the input units to 0 at each update during training, thus reducing the model's reliance on any single neuron.

2.2.3 Model 3: CNN with Data Augmentation

My third model used the same architecture as Model 2 but was trained using data augmentation to further enhance its generalization capabilities. The training data generator was configured to apply random transformations to the images in real-time:

- Rotation (up to 20 degrees)
- Width and height shifts (up to 20% of the dimension)
- Shearing and zooming (up to 20%)
- Horizontal flipping

This technique artificially expands the training dataset, exposing the model to a wider variety of intra-class variations and forcing it to learn features that are invariant to these transformations.

2.3 Training and Evaluation

All models were trained using the Adam optimizer, an adaptive learning rate method that is an advanced variant of stochastic gradient descent. The `categorical_crossentropy` loss function was used, as it is the standard surrogate loss for multi-class classification problems. Performance was monitored using the accuracy metric. For the final architecture, a grid search over learning rates (10^{-3}, 10^{-4}) was performed to select the optimal value, using performance on the validation set as the sole criterion.

3 Results

The three models were trained and their performance on the training and validation sets was recorded at each epoch.

3.1 Model 1: Baseline Performance and Overfitting

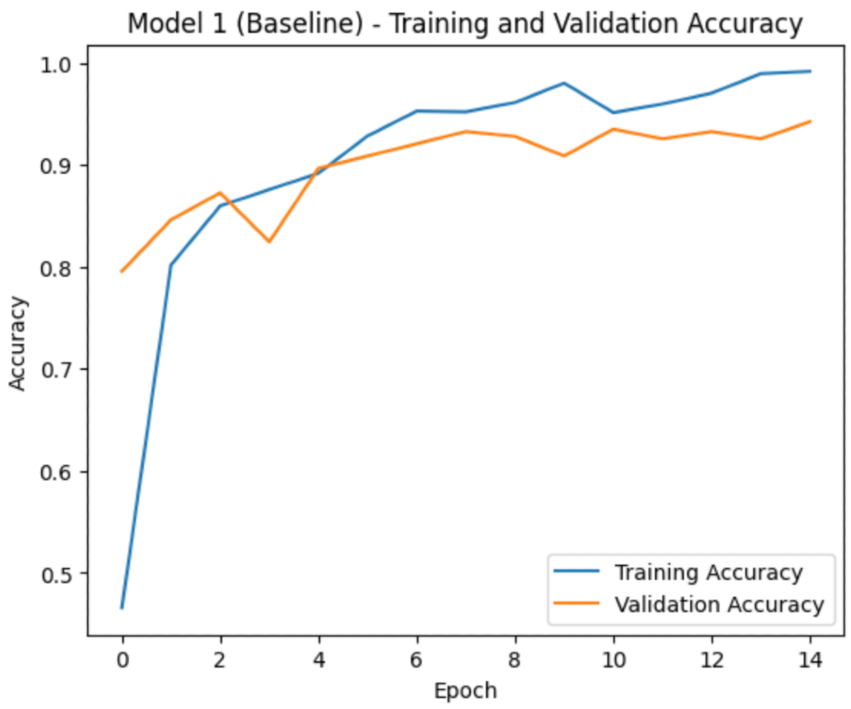
The baseline model trained very quickly, achieving a training accuracy of 99.3% but a peak validation accuracy of only 95.2%. The training and validation curves, shown in Figure 1, exhibit a significant divergence. This growing gap is a classic sign of **overfitting**, where the model learns the training data exceptionally well but fails to generalize to unseen examples. In the context of the bias-variance trade-off, this model demonstrates low approximation error (bias) but high estimation error (variance).

3.2 Model 2: The Effect of Dropout

The second model, incorporating an additional convolutional block and a dropout layer, showed improved performance. It achieved a peak validation accuracy of 96.2%. As seen in Figure 2, the gap between the training and validation curves is noticeably smaller compared to the baseline model. This demonstrates that the dropout layer successfully acted as a regularizer, reducing the model's variance and improving its generalization capability.

3.3 Model 3: The Power of Data Augmentation

The third model, trained with data augmentation, yielded the best results. It reached a peak validation accuracy of 98.6%. The training and validation curves in Figure 3 track each other very closely, indicating that the overfitting problem has been largely



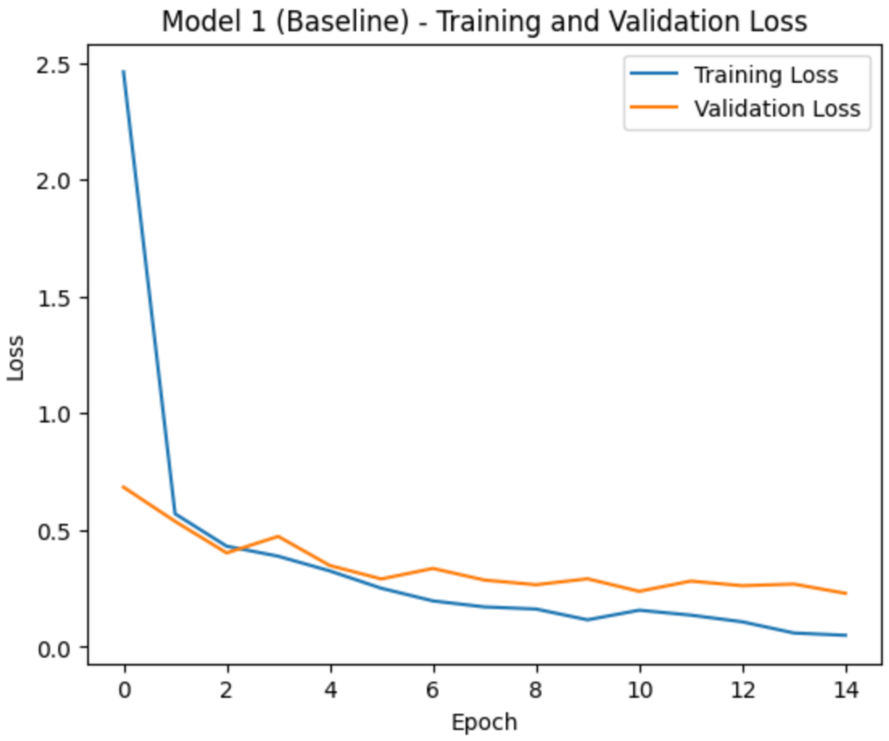


Fig. 1 Training and validation accuracy/loss for Model 1. The growing gap between the training and validation curves indicates significant overfitting.

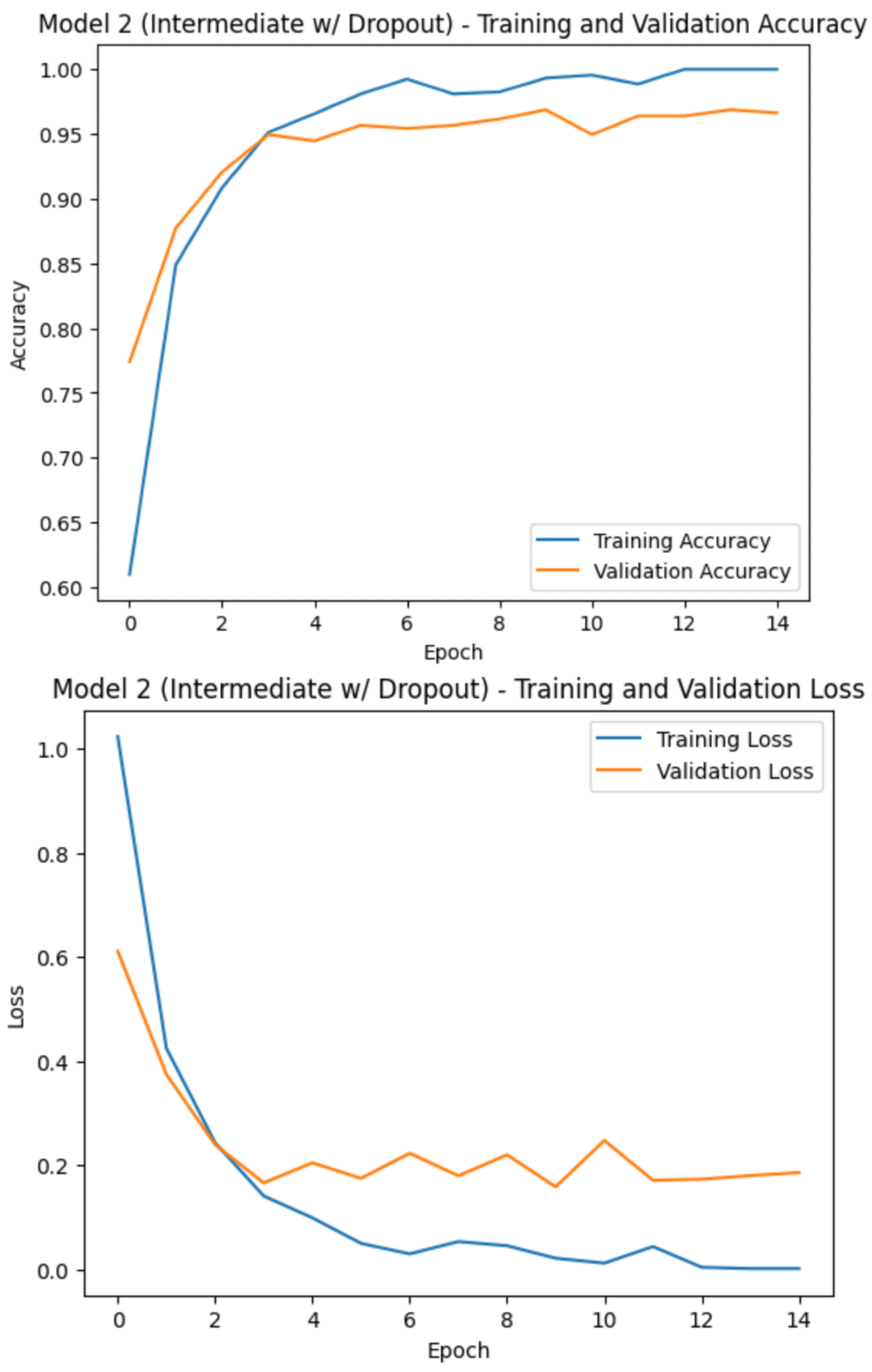


Fig. 2 Training and validation accuracy/loss for Model 2. The gap between the curves is reduced, showing that dropout has mitigated some overfitting.

resolved and the model generalizes well. The training accuracy is now much more aligned with the validation accuracy, suggesting a well-balanced model.

3.4 Final Model Evaluation

After hyperparameter tuning, my best model (Model 3 architecture) was evaluated a final time on the held-out test set. This provides an unbiased estimate of its performance. The model achieved a **test accuracy of 97.72%**. The detailed classification report is provided in Table 2, and the confusion matrix in Figure 4.

Table 2 Classification Report for the Final Model on the Test Set

Class	Precision	Recall	F1-Score	Support
rock	0.99	0.94	0.96	141
paper	0.95	1.00	0.97	150
scissors	0.99	0.99	0.99	148
Accuracy			0.98	439
Macro Avg	0.98	0.98	0.98	439
Weighted Avg	0.98	0.98	0.98	439

3.5 Generalization Test on Custom Images

To further probe the model's generalization capabilities, a small, challenging test set of 6 custom images (2 for each class) was created using a smartphone camera. These images were intentionally captured under different conditions than the training set, featuring real-world backgrounds and natural lighting to simulate a domain shift.

The model's performance on this out-of-distribution data dropped significantly. It correctly classified the 2 'paper' images but misclassified all 4 'rock' and 'scissors' images, also as 'paper'. This resulted in an accuracy of **33.3%** (2 out of 6 correct). The model was also highly confident in its incorrect predictions, highlighting a common failure mode for deep neural networks when faced with out-of-distribution samples.

4 Discussion

This project successfully demonstrated a structured approach to building a high-performance CNN classifier. The progression from a simple, overfitting model to a well-regularized, generalizable model highlights key concepts from statistical learning.

The initial baseline model was a clear example of high variance. This was addressed by introducing regularization. Model 2 showed that dropout is an effective technique for reducing variance. However, the most significant improvement came from Model 3 with data augmentation. By artificially enriching the training data, I forced the model to become invariant to minor visual variations, which dramatically improved its generalization. My experiments operated in the "classical" learning regime, where increasing

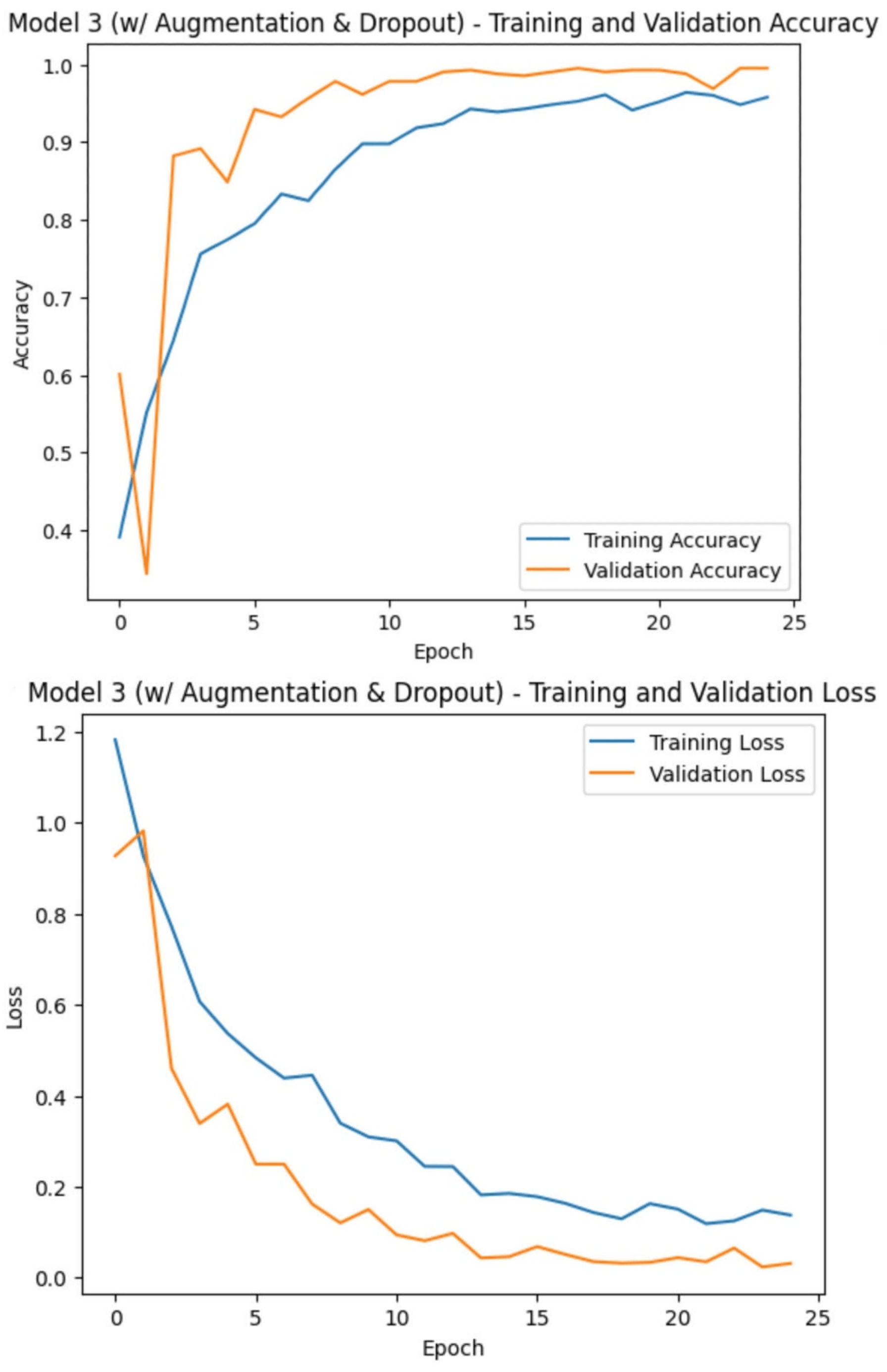


Fig. 3 Training and validation accuracy/loss for Model 3. The curves are closely aligned, indicating a well-regularized model with excellent generalization.

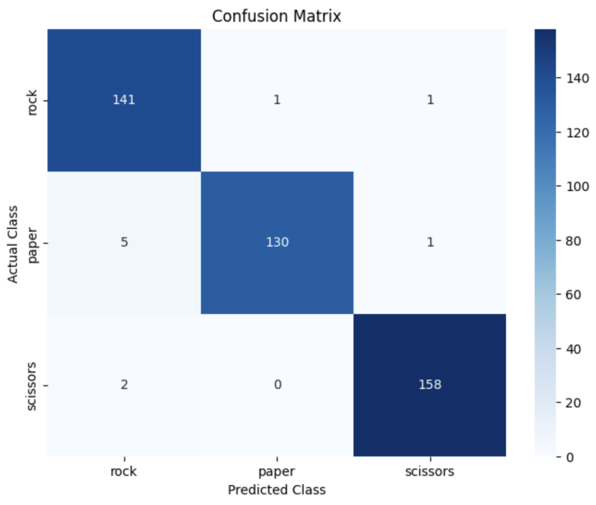


Fig. 4 Confusion matrix for the final model on the test set. The model shows high accuracy across all classes, with the most frequent error being the misclassification of 'rock' as 'paper'.

model capacity without proper regularization leads to worse test performance. I did not venture into the "modern" over-parameterized regime where the "double descent" phenomenon might be observed, as my goal was to build an efficient and robust model.

The final test accuracy of 97.7% is a strong result, but the optional generalization test provided the most critical insight. On the small set of 6 custom images, the model's accuracy plummeted to 33.3%. This drastic performance drop is a textbook example of **domain shift**, where a model trained on one data distribution fails to generalize to another. The model had clearly over-specialized on features specific to the Kaggle dataset (like plain backgrounds and studio lighting) rather than learning the abstract shape of the gestures themselves. This underscores that even a model with a low statistical risk on its test set may not be robust in real-world applications without being trained on more diverse data. This suggests that for a real-world application, the training set would need to be significantly expanded to include images from a wide variety of environments, or techniques like domain adaptation would be necessary.

5 Conclusion

In this project, I successfully developed a Convolutional Neural Network to classify Rock-Paper-Scissors hand gestures with high accuracy. Through a methodical process of incremental complexity and the application of regularization techniques—namely dropout and data augmentation—I systematically addressed the problem of overfitting and significantly improved the model's generalization capabilities. The final model achieved an accuracy of 97.7% on its in-distribution test set. However, a final test on out-of-distribution images revealed the model's sensitivity to domain shift, a critical limitation for real-world deployment. This work serves as a practical case study in applying fundamental machine learning principles to build and critically evaluate deep learning systems.

Appendix A Model Summaries

This appendix contains the detailed architecture summaries for the three models, as generated by Keras.

A.1 Model 1 Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 16)      448

 max_pooling2d (MaxPooling2D) (None, 74, 74, 16)       0

 flatten (Flatten)           (None, 87616)             0

 dense (Dense)               (None, 64)                5607488

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 5,608,131
Trainable params: 5,608,131
Non-trainable params: 0
_________________________________________________________________
```

A.2 Model 2 Summary

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 148, 148, 32)      896

 max_pooling2d_2 (MaxPooling  (None, 74, 74, 32)        0
 2D)

 conv2d_3 (Conv2D)           (None, 72, 72, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 36, 36, 64)        0
 2D)

 flatten_1 (Flatten)         (None, 82944)             0

 dropout (Dropout)           (None, 82944)             0
 dense_2 (Dense)             (None, 128)              10616960

 dense_3 (Dense)             (None, 3)                387

=================================================================
Total params: 10,636,739
Trainable params: 10,636,739
Non-trainable params: 0
_________________________________________________________________
```

A.3 Model 3 Summary

The architecture for Model 3 is identical to Model 2. The difference lies in the training data, which was augmented as described in the Methods section.

References

[1] Cesa-Bianchi, N. (2024). *Course Notebook: Statistical Methods for Machine Learning*. University of Milan.

[2] Dr. G. Freeman. (2019). Rock Paper Scissors Dataset. Kaggle. https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors