Gesture based UI project 2

Kevin McShane, Student ID: G00401808

Atlantic technological university

12-05-2025

1 Introduction

This is a project which demonstrated how the subset of AI, machine learning can be used to identify hand gestures. This is done using a smaller version of the HAnd Gesture Recognition Image Dataset (HAGRID).

2 Methodology

This chapter covers everything that pertains to data manipulation before the machine learning logic takes place.

2.1 Data Pre-Processing

Data pre-processing is when "Data preprocessing for machine learning (ML) refers to the preparation and transformation of raw data into a format suitable for training ML models. It's an essential step in an ML (or AI) pipeline because it directly impacts the performance and accuracy of the models."[1]

This process was partially automated using well known tools.

The process included:

- Unzipping the data-set contents.
- Using a helper method to enter the dataset and split its contents into a training set and validation set (for testing on unseen data), sized 80:20 respectively.
- Both the training and validation set has an input (the image) and expected output (name of pose detected).

2.2 Data Labeling

The dataset contained labeled data. What is meant by labeled data in this case is that the image (input into the model) is stored in a file named after the appropriate pose it contains, which can be used as reference for the correct answer in the models case.

As for new data that had to be created for testing purposes had to be categorized manually, however this was a quick small scale test only containing 4 new pictures.

After the data had been pre-processed the output had been translated to one-hot.

What is one-hot encoding

It is a way of representing categorical data in binary encoding, categorical data meaning the classifications. For example there are 18 possible classifications in this dataset, to represent this, the y value would be a 18 bit representation where only a single bit is a 1. The position of the 1 bit denotes which class the input belongs to.

2.3 Data Scaling

This process is also known as data normalization. This is done to scale the pixel values to a standard range (typically 0 to 1), making the network's computations more stable and efficient.

Input Normalization (CNN Model)

The custom Convolutional Neural Network (CNN) model uses a Rescaling layer (1./255) to normalize pixel values from the range [0, 255] to [0, 1]. This standardization mitigates large disparities in pixel magnitudes, aiding gradient stability during training.

Test Data Preparation

During inference, test images are manually rescaled by dividing pixel values by 255.0 before the prediction. Layers for Batch Normalization are also added to the CNN

Augmentation and Rescaling

Data augmentation (random flips and rotations) is applied to both models, it operates after rescaling in the CNN pipeline and before ResNet preprocessing in the transfer learning pipeline.

2.4 Data Analysis and Visualization

Analysis

In this project all of the model training process is split into many epochs, statistics are printed during this process to show the accuracy of the network on both the data it was trained on and the results after being tested on the unseen data. After each epoch the model retrains with adjustments and measures its accuracy again until all epochs are complete or the results of the early out condition is met (when validation loss fails to improve for 5 consecutive epochs).

Epoch statistics analyzed:

- Accuracy (training statistic)
- val_accuracy
- Loss (training statistic)
- val_loss

Accuracy

This represents the accuracy of the model's classifications.

Loss

This represents how incorrect the classification of the data is.

Visualization

The project includes a well-designed visualization component that outputs a portion of the dataset (9 images from the first stack) with their associated labels printed above them.

This was done to ensure the pre-processing of the image was correct, things such as height, width and color of the photo. This also helped ensure that the data was labeled correctly.

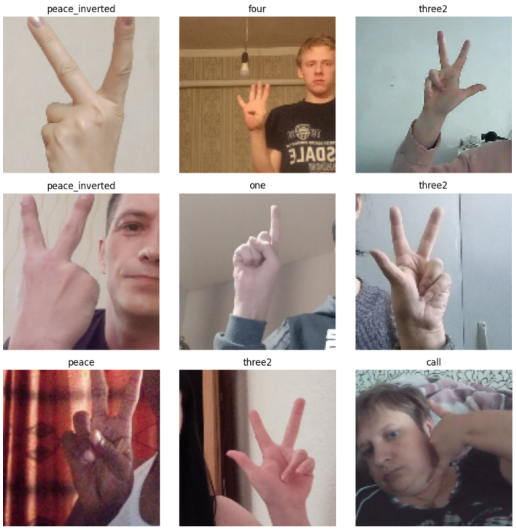

Figure 1: Example training set.

3 Experiments and results

3.1 Classifier Models

3.2 CNN experiments taken

Many different combinations of the pre-processing and layer configurations were used before getting the current CNN seen in the Jupyter notebook.

CNN Experiment 1

At first the CNN had 3 layers which consisted of:

- Input layer
- Rescaling layer
- 3 Conv2D() layers consisting of 16, 32 and 128 filters

All of the CNN Conv2D() layers throughout development use the relu activation function.

result: 0.8 train accuracy, 0.59 validation accuracy.

CNN Experiment 2

It was noticeable that the CNN from experiment 1 was overfit due to its training_accuracy score overshadowing its validation counterpart, to combat this too many changes were made, which made the model a lot less accurate. Overall it became harder to diagnose success.

Changes added:

- Augmentation to stop the data from becoming to familiar, preventing the overfitting of the model
- A learning rate
- Dropout layers

results: 0.59 train accuracy, 0.57 validation accuracy.

CNN Experiment 3-5

After regressing back to the state of experiment 1, changes made in experiment 2 were slowly experimented with until a combination worked.

There was no need to introduce an explicit learning rate, that was unnecessary complication.

The main improvement that showed results was adding a final dense layer with 256 neurons.

results: 0.8815 train accuracy, 0.8327 validation accuracy. (early out because of 5 epochs without the val_loss decreasing)

Developing the ResNet50 transfer learning model

The development process of this model had been based on researching medium's articles and guides.[2]

The training of the resnet model happened in two stages

First Model (Feature Extraction)

The first model used a pretrained ResNet50. Its layers were frozen. Only the new layers on top were trained. These new layers learned to classify gestures using ResNet's features.

Second Model (Fine-Tuning)

The second model unfroze some ResNet50 layers. This adjusted the pretrained features to better fit gestures. The model improved by refining both high-level and low-level patterns.

results of resnet: train accuracy: 0.9213, validation accuracy: 0.9311

4 Final test: new authentic data

The final task required in this project was to gather 4 images of hand gestures and use them to test the models.

Results: Both models, despite good scores failed to identify the gestures given as each model returned the same answer four times.

For CNN it was the "one" gesture and for the ResNet it choose the "fist" gesture.

4.1 Results

Clearly the more accurate model was ResNet according to the validation scores, however the tests revealed both of the models were flawed and had a tendency to lean toward one answer. An investigation could be done into the dataset to check if it is balanced, however it is not certain that this is the cause. It could have also been cause by the method used to preprocess the 4 images.

4.1.1 Results table

Table 1: Model Performance Comparison

Metric	**Custom CNN**	**ResNet50**
Training Accuracy	88%	92%
Validation Accuracy	83%	93%
Training Speed	Faster	Slower (two phases)
Overfitting	Moderate	Low

5 Conclusion

In conclusion the CNN built from scratch was less effective than the ResNet50 model, however good scores do not guarantee good results on unseen data.

References

[1] PureStorage. What is data preprocessing for machine learning? *PureStorage*, 2024.

[2] Dr. Partha Majumde. Supercharge your ai: Resnet50 transfer learning unleashed. Technical report, medium, 2024.