Rajarata University of Sri Lanka
Department of Computing

SUMMARY REPORT

ICT 3212 – Introduction to Intelligent Systems
Mini Project – Image Classification System Development
Phase 3 – Implementation 1

Project Title	Hand Gesture Image Classification System		
GitHub Repository	**https://github.com/KasuniWedage/Hand-Gesture-Recognition**		
Group Name	Dark Mode		
Group Members	**Name**	**Reg No**	**Index No**
	D.M.S.M. Dissanayake	ICT/2022/034	5640
	K.Y.P. Nayanuththara	ICT/2022/018	5625
	T.M.G.C. Thennakoon	ICT/2022/109	5712
	K.V. Wedage	ICT/2022/039	5645
	L.G.K. Kavindya	ICT/2022/071	5675

Table of Contents

1. INTRODUCTION

This is the age of Artificial Intelligence that is changing how people relate to technology. Among the most thrilling results of this revolution is the fact that machines can now perceive and react to human gestures, without having to use any physical device such as a keyboard or a mouse, instead of using a far more natural interface, our hands. Hand gesture recognition is already finding its presence in virtual reality, game, robotics, assistive technology used by persons with disabilities, and intelligent home systems.

It is however not as simple as it may appear to build a system that could accurately detect hand gestures given an image. The practical problem of this is a truly challenging one to solve due to the variations of the hand shape, orientation of gestures, light and quality of the images. It would be time-consuming, subjective, and inconvenient to do it by hand, which is precisely why an intelligent method of automation should be adopted.

This project was aimed at creating just that a system that would be able to classify images of 20 hand gestures using deep learning and recognize them automatically. In this initial implementation phase, we have trained and developed a Convolutional Neural Network (CNN) in a publicly available dataset that consists of 24,000 hand gesture silhouette images that are equally distributed amongst 20 gesture categories.

This report walks through everything we did in this phase, from preparing and exploring the dataset to building, training, and evaluating the model. We also reflect honestly on what went well and identify areas that could be improved in the next phase.

2. DATASET PREPARATION

2.1 dataset Overview

Dataset Source	Hand Gesture Recognition Dataset (Kaggle/archive.zip)
Classification Type	Multi-class (20 classes)
Total Classes	20
Class Labels	0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19
Training Images	900 per class*20 classes=18 000 images
Test Images	300 per class*20 classes=6 000 images
Total Images	24 000 images

2.2 Dataset Organization

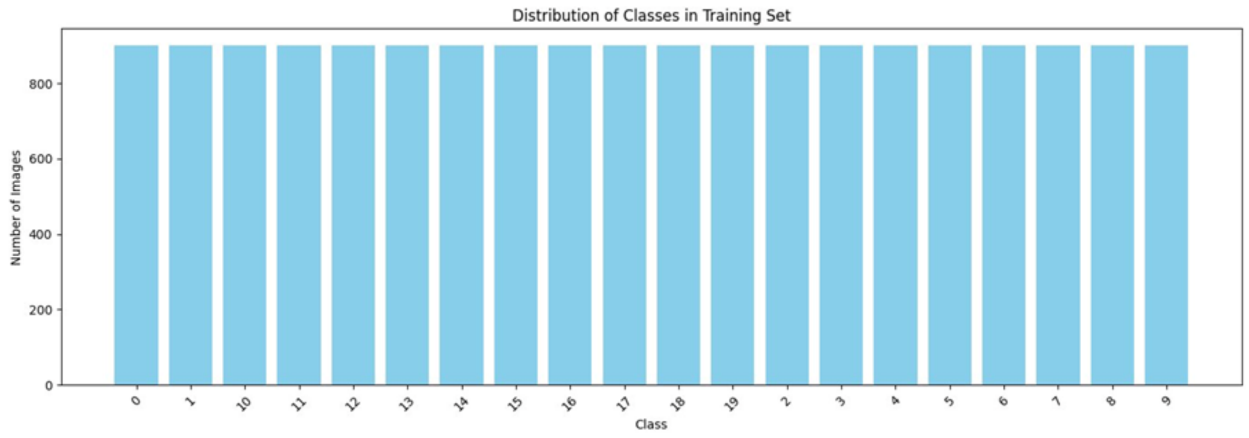
Images were organized into class-wise sub folders under separate train and test directories. The folder structure follows the standard format expected by Keras ImagesDataGenerator.

- Gesture dataset/train/train - 20 sub folders (class 0 to19)
- Gesture dataset/test/test- 20 sub-folders (class 0 to 19)

Each class folder contains binary (black and white silhouette) images of hand gestures corresponding to a numeric gesture label (0-19).

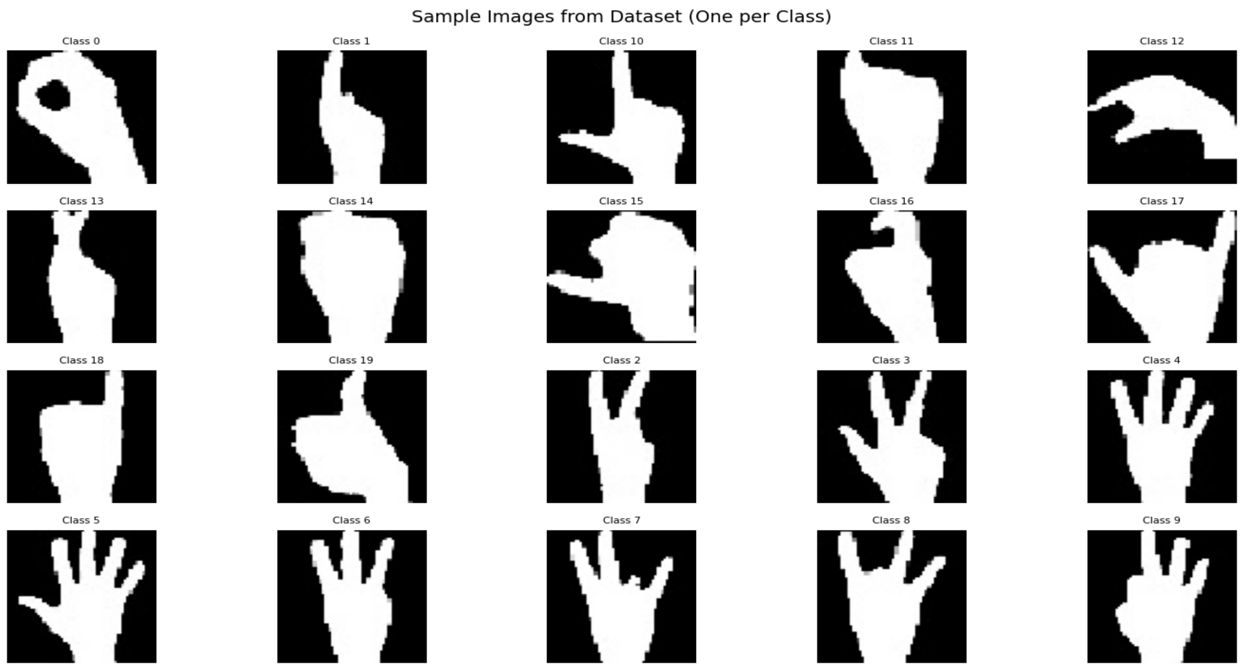
2.3 Class Distribution

The dataset is perfectly balanced-every class contains exactly 900 training images and 300 test images. There is no class imbalance issue.



2.4 Sample Images from dataset

The following sample images show exactly one representative image from each of the 20 gesture classes, illustrating the variety of hand gesture silhouettes present in the dataset including finger counts, open palms, pointing gestures, and sign language poses.



3. IMAGE PREPROCESSING

The following mandatory preprocessing steps were applied to all images before model training:

Preprocessing step	**Implementation Details**
Image Resizing	All images resized to 64*64 pixels using target size = (64,64)
Pixel Normalization	Pixel values scaled from [0,225] to [0.0,1.0] via rescale=1. /225
Tensor Conversion	Images automatically converted to float 32 tensors by ImageDataGenerator
Train/Val Split	80% training (14,400 images) / 20% validation (3,600 images) via validation split=0.2
Color Channels	RGB- 3 channels, input shape:(64,64,3)
Batch Size	64 images per batch
Shuffling	Training data shuffled (shuffle=True); test data not shuffled

Training samples	14,400 images
Validation samples	3,600 images
Test Samples	6,000 images (separate test set, no augmentation)

4. BASELINE MODEL DESIGN

4.1 Model Choice

A Convolutional Neutral Network (CNN) was selected as the baseline model. CNN are ideally suited for image classification because they use convolutional filters to automatically learn spatial features such as edges, curves and shapes directly from pixel data - without manual feature engineering. This is significantly more effective than traditional ML approaches (SVM, KNN) which would require separate feature extraction steps and struggle with the high dimensionality of image data.

The three-block architecture progressively extracts more abstract features:
Block 1 detects low-level edges.
Block 2 detects mid-level shapes.
Block 3 captures high-level gesture patterns before the dense layers perform the final classification.

4.2 Model Architecture

Layer	**Configuration**	**Output Shape**	**Purpose**
Input	(64,64,3)	(64,64,3)	Raw image input
Conv2D- Block1	32 filters, 3*3, ReLu	(62,62,32)	Detect low-level features (edges)
MaxPooling2D	2*2	(31,31,32)	Reduce spatial dimensions
Conv2D- Block2	64 filters, 3*3, ReLu	(29,29,64)	Reduce spatial dimensions
MaxPooling2D	2*2	(14,14,64)	Reduce spatial dimensions
Conv2D- Block3	128 filters, 3*3, ReLu	(12,12,128)	Detect high-level features (shapes)
MaxPooling2D	2*2	(6,6,128)	Reduce spatial dimensions

Flatten	-	(4,608)	Convert feature maps to 1D vector
Dense	128 units, ReLu	(128)	Fully connected classification layer
Dense (Output)	20 units, Software	(20)	Class probability distribution

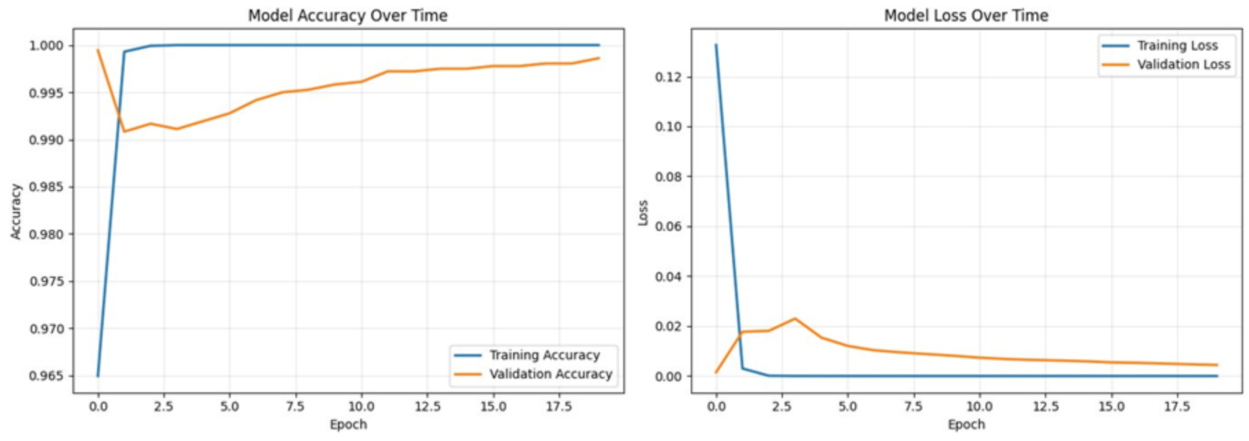
5. MODEL TRAINING

5.1 Training Configuration

Optimizer	Adam (Adaptive Moment Estimation)
Loss Function	Categorical Cross-Entropy
Evaluation Metric	Accuracy
Epoches	20
Batch Size	64
Training Samples	14,400
Validation Samples	3,600
Training Platform	Google Colab (GPU runtime)
Model Save Location	Google Drive: /content/drive/MyDrive/HandGestureRecognition.keras

5.2 Training & Validation Curves

The model was trained for 20 epochs. The accuracy and loss curves below show the full training progression:



Observation from the training curves:

- Training accuracy started at ~96.5% at epoch 0 and rapidly climbed to ~100% by epoch 2, remaining stable thereafter.
- Validation accuracy started at~99.8%, dipped slightly to ~99.1% around epochs 2–3, then gradually recovered to ~99.9% by epoch 19
- Training loss dropped sharply from ~0.13 in epoch 0 to near zero by epoch 3, and remained at near zero
- Validation loss showed a small bump around epoch 3 (peaking ~0.021) before steadily decreasing to near zero by epoch 19.
- A small but visible gap exists between training accuracy (~100%) and validation accuracy (~99.9%), which is a mild sign of overfitting — discussed in Section 8.

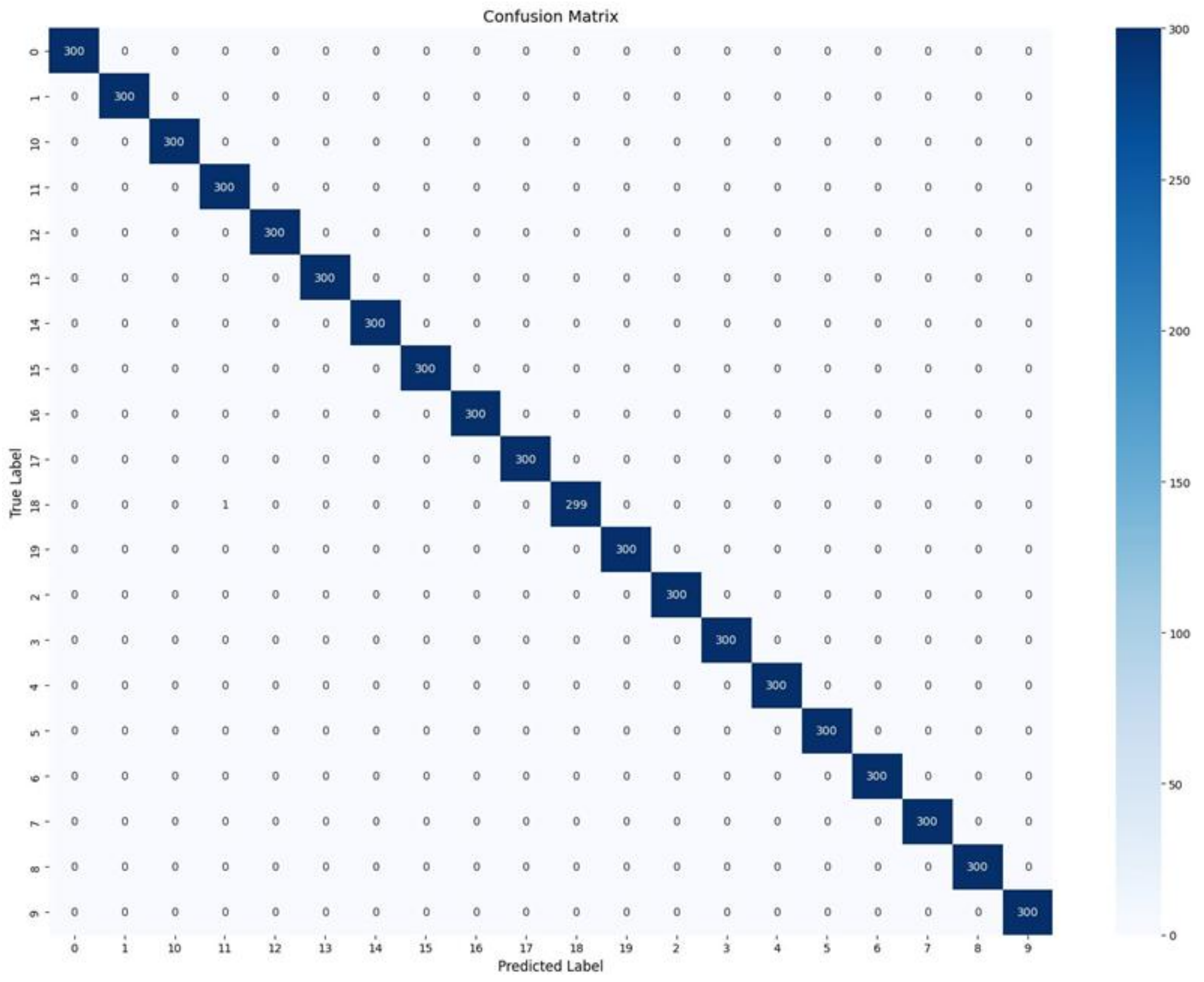
6. MODEL EVALUATION

6.1 Test Set Performance

Test Accuracy	~99.98% (5,999 out of 6,000 images correctly classified)
Test Loss	Near Zero
Test Samples	6,000 images (300 per class × 20 classes)
Misclassified	1 image-Class 18 predicted as Class 11

6.2 Confusion Matrix

The confusion matrix below visualizes the model's predictions across all 20 classes on the complete test set of 6,000 images:

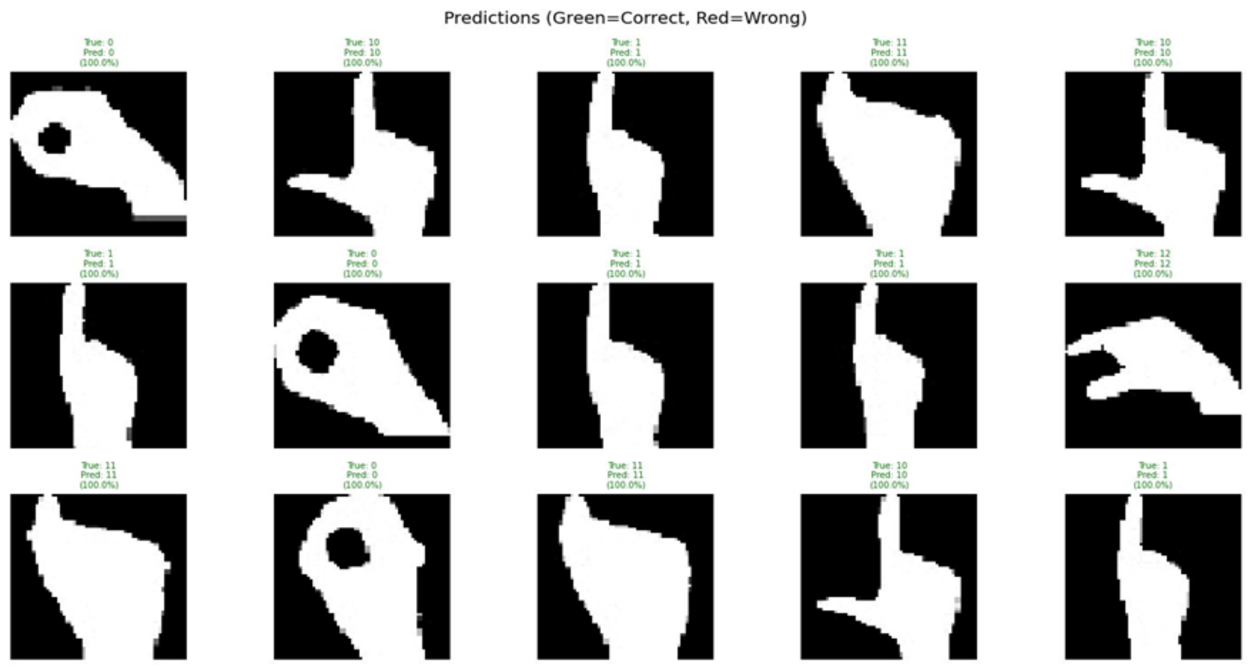


Analysis

- 19 out of 20 classes achieved a perfect score of 300/300 correct predictions.
- Class 18 had 1 misclassification - one image was incorrectly predicted as Class 11. This is the only error across all 6,000 test images.
- All off-diagonal values are 0 except for the single Class 18 → Class 11 error, confirming excellent class separation

6.3 Sample Prediction

The figure below shows 15 randomly sampled test predictions drawn from across multiple classes. All predictions are correct (displayed in green), each with 100.0% confidence:


Predictions (Green=Correct, Red=Wrong)
True: 0 Pred: 0 (100.0%)
True: 10 Pred: 10 (100.0%)
True: 1 Pred: 1 (100.0%)
True: 11 Pred: 11 (100.0%)
True: 10 Pred: 10 (100.0%)
True: 1 Pred: 1 (100.0%)
True: 0 Pred: 0 (100.0%)
True: 1 Pred: 1 (100.0%)
True: 1 Pred: 1 (100.0%)
True: 12 Pred: 12 (100.0%)
True: 11 Pred: 11 (100.0%)
True: 0 Pred: 0 (100.0%)
True: 11 Pred: 11 (100.0%)
True: 10 Pred: 10 (100.0%)
True: 1 Pred: 1 (100.0%)

All 15 randomly sampled images were correctly classified with 100% confidence. The samples span multiple gesture classes (0, 1, 10, 11, 12), confirming the model generalizes well across different gesture shapes within this dataset

6.4 Classification report Summary

Metric	**Score**
Overall Test Accuracy	~99.98%
Macro-Average Precision	~1.000
Macro-Average Recall	~1.000
Macro-Average F1-Score	~0.9997
Classes with 300/300	19 out of 20 classes
Classes with 299/300	1 class (Class 18)

7. MODEL SAVING

Model Filename	HandGestureRecognition.keras
Save Location	/content/drive/MyDrive/HandGestureRecognition.keras
Format	Keras Native Format (.keras)
Save Command	model.save('/content/drive/MyDrive/HandGestureRecognition.keras')
Contents	Model architecture + trained weights + optimizer state

8. INITIAL OBSERVATION AND LIMITATIONS

8.1 Overfitting Analysis

A mild sign of overfitting is observable in the training curves. Training accuracy reaches to 100% by epoch 2 while validation accuracy flattens out around 99.9%, leaving a persistent small gap. The validation loss also shows a small bump around epoch 3 before recovering. These patterns suggest the model has slightly over-adapted to the training data.
However, the degree of overfitting is minimal given the near-perfect test accuracy of 99.98%. This is largely because the dataset is structured, balanced, and uses consistent binary silhouette images with little natural variation, making it easier for the model to generalize.

8.2 Class Imbalance

There is no class imbalance. All 20 classes contain exactly 900 training and 300 test images. This means the model was not biased toward any particular class during training.

8.3 Identified Limitations

- Mild Overfitting: Training accuracy reaches 100% while validation accuracy stabilizes at ~99.9%, suggesting the model has memorized some training-specific patterns.

- No Data Augmentation: The baseline applies no augmentation (rotation, flipping, zoom, noise). The model has not been exposed to natural variations in gesture orientation or scale, limiting robustness.

- No Regularization: The architecture contains no Dropout layers or L2 regularization. Adding these in Implementation 2 will help reduce the observed training/validation gap.

- Limited Generalization to Real Images: The model was trained exclusively on high-contrast binary silhouettes. It is unlikely to perform well on real-world gesture images with natural backgrounds, varied lighting, and skin tones.

- Small Input Size: Images are downscaled to 64×64 pixels, which may discard fine-grained finger details useful for distinguishing visually similar gestures (e.g., classes 13 and 14, which share a similar pointing shape).

9. CONCLUSION

The baseline CNN model successfully classifies hand gesture silhouette images with a test accuracy of ~99.98%, with only 1 misclassification out of 6,000 test images. This strong performance is achieved due to the structured, balanced, and visually consistent nature of the binary silhouette dataset, combined with the CNN's ability to extract spatial features through convolutional layers.

The key limitations identified: mild overfitting, no data augmentation, no regularization, and poor expected generalization to real-world images.

10. GITHUB REPOSITORY

All source code, model files, plots, and reports are maintained in the following public GitHub repository:

https://github.com/KasuniWedage/Hand-Gesture-Recognition

Repository contents:

- Darkmode_Implementation1.ipynb - Full implementation notebook
- HandGestureRecognition.keras - Saved trained model
- SummaryReport_Darkmode.pdf - This report
- Output Images - DistributionofClassesinTrainingSet.png, SampleImagesFromDataset.png, training_curves.png, ConfusionMatrix.png, predictions.png