Alphabet Classification Using Deep Learning Architectures

A comprehensive study on classifying alphabet characters using various deep learning models, comparing their performance and architectural nuances. This document explores the implementation of VGG-19 from scratch and the application of transfer learning with ResNet, Inception V1, and MobileNet.

Table of Contents

Introduction

Project overview and objectives.

Dataset Description

Details of the EMNIST / A–Z dataset.

Preprocessing

Image preparation steps for model training.

Model Architectures

In-depth look at VGG-19, ResNet, Inception V1, and MobileNet.

Training Setup

Configuration and parameters for model training.

Evaluation & Results

Performance metrics and comparative analysis.

Discussion & Conclusion

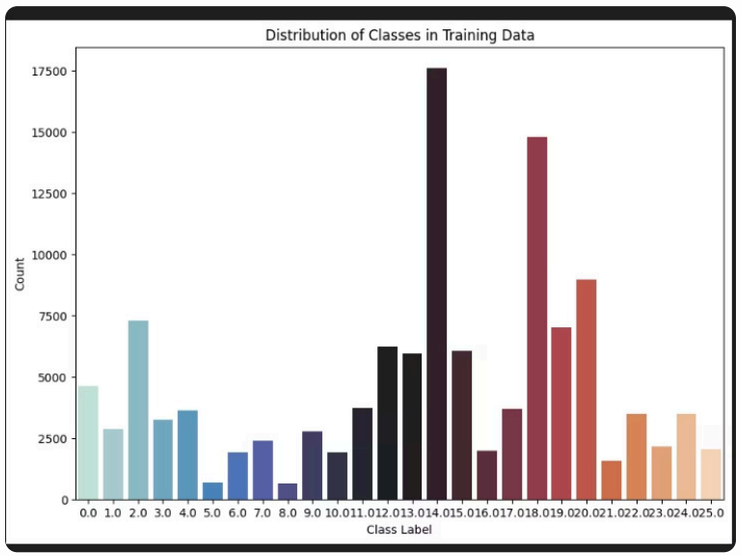
Insights, challenges, and future work.

References

Sources and acknowledgments.

Introduction to Alphabet Classification

This project focuses on the intricate task of classifying alphabet characters, encompassing both handwritten and printed forms. We delve into the application and comparative analysis of four distinct deep learning architectures to achieve this goal. The primary objectives are multifaceted, designed to provide a comprehensive understanding of each model's capabilities and limitations in this domain.



- Build VGG-19 from scratch to understand foundational CNN principles.
- Utilize pre-trained models (ResNet, Inception V1, MobileNet) via transfer learning.
- Apply transfer learning techniques effectively to leverage existing knowledge.
- Evaluate and compare models rigorously using various performance metrics.
- Visualize performance through confusion matrices, ROC curves, and AUC scores.

The study aims to contribute to the understanding of deep learning effectiveness in character recognition, highlighting architectural strengths and weaknesses.

Dataset Description: EMNIST / A–Z

For this classification task, we utilize the EMNIST / A–Z handwriting dataset, a challenging collection designed to test robust character recognition. This dataset provides a rich variety of handwritten and printed characters, serving as an ideal benchmark for our deep learning architectures.

1	2
Classes 26 (A-Z)	Training Samples Over 300,000
3	**4**
Testing Samples Approximately 74,000	Image Size 28x28 pixels

The dataset presents several inherent challenges that require careful consideration during model training and evaluation. These challenges are crucial for developing robust and accurate classification models.

Essential Preprocessing Steps

Effective preprocessing is paramount for preparing the EMNIST / A–Z dataset for deep learning models, ensuring optimal performance and mitigating the challenges posed by raw image data. Each step is carefully designed to transform the input images into a format suitable for neural network consumption.

Convert to Grayscale

Convert images to grayscale to reduce dimensionality and focus on character structure.

Resize Images

Uniformly resize images to match the input dimensions required by specific models (e.g., 224x224 for VGG-19).

Normalization

Scale pixel values to a standard range (e.g., [0, 1] or [-1, 1]) to aid model convergence and stability.

Train-Test Split

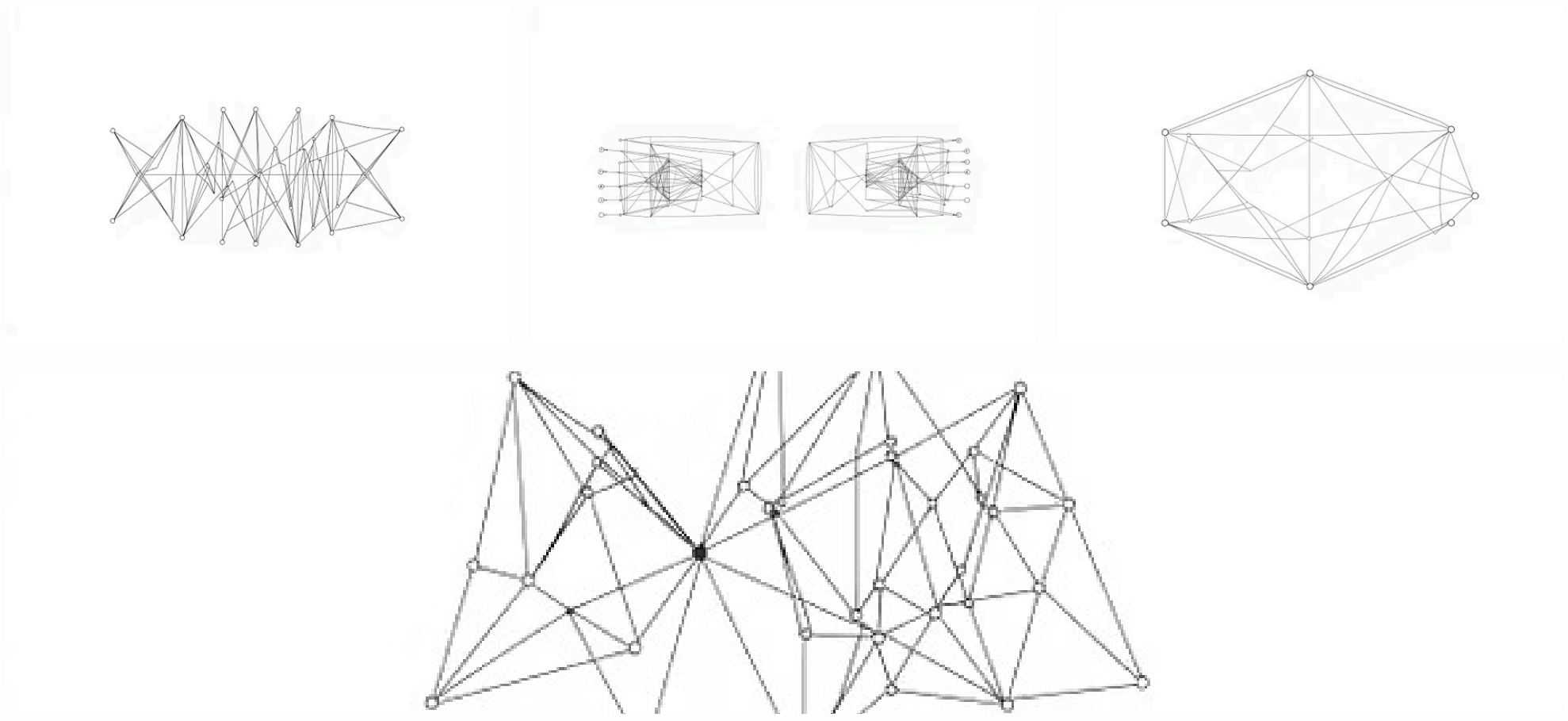
Divide the dataset into training and testing sets to evaluate model generalization.

Data Augmentation

Apply transformations like flipping, rotation, shifting, and zooming to increase dataset size and improve model robustness.

Deep Learning Architectures Explored

Our project investigates the efficacy of four prominent deep learning architectures, each offering unique approaches to image classification. This selection allows for a comprehensive comparative analysis of their performance on the alphabet classification task.



VGG-19 (from scratch)

Built entirely from scratch to understand foundational convolutional layers and their impact on learning hierarchical features.

ResNet (transfer learning)

Leverages residual connections for deeper networks, adapted via transfer learning to our specific character classification problem.

Inception V1 (transfer learning)

Utilizes inception modules for efficient computation and multi-scale feature extraction, fine-tuned for alphabet recognition.

MobileNet (transfer learning)

Designed for mobile and embedded vision applications, employed for its efficiency and adapted using transfer learning.

Training Parameters

Parameter	Value	Notes
Input Shape	224 * 224 * 3	All input images (EMNIST, A-Z) must be resized to this standard VGG input.
Optimizer	**Adam**	A robust choice for most deep learning tasks.
Loss Function	**Categorical Cross-entropy**	Standard loss function for multi-class classification with one-hot encoded labels.
Learning Rate	0.001 (or 10^{-3})	A common starting point; may require tuning or decay.
Number of Epochs	20 - 50	Depends on the dataset size (EMNIST is large) and available computational resources.
Final Layer Output	26 units	N equals the total number of alphabet classes (e.g., 26 for English A-Z, or more for combined Arabic/English).

VGG-19 (from scratch)

1. Model Details

Model Name: Custom VGG-19-Inspired Convolutional Neural Network (CNN)

Frameworks & Versions:

- Python 3.9
- TensorFlow 2.x (Keras API)
- NumPy, Pandas, Scikit-learn, Matplotlib

Date: 2025

2. Intended Use

This model is designed for: Task Handwritten character recognition for A–Z English alphabet letters (26 classes). Target Audience Researchers, ML students, and developers working on:

- Optical Character Recognition (OCR)
- Handwriting digitization
- Document processing
- Education-based recognition systems

Use Cases

✔️Recognizing handwritten letters

✔️ Preprocessing & classification in OCR pipelines

✔️ Real-time character prediction

3. Training Data

Dataset Used A–Z Handwritten Alphabets Dataset (CSV format)

- Source: A_Z Handwritten Data.csv
- Total samples in raw dataset: ~372,000 Each row: Column 0 → Label (0–25) Columns 1–784 → Pixel values (28×28 grayscale)

Data Preprocessing Missing value removal

- Duplicate removal
- Removal of invalid pixel values
- Removal of empty images
- Removal of uniform/low-contrast images
- Stratified split into Train/Test/Validation
- Pixel normalization (0–1)
- Reshaping to 28×28×1
- Resizing to 224×224×3 for VGG
- Data augmentation using:
- RandomRotation (12%)
- RandomZoom (12%)
- RandomTranslation (12%)
- RandomContrast (12%)

Final Data Split Split Percentage Notes

Training	68%	after augmentation
Validation	12%	stratified
Test	20%	final evaluation

4.Model Architecture

a custom VGG-19 variant with:

Convolutional Blocks (VGG Style)

Each block uses:

- 3×3 Convolution
- ReLU activation
- BatchNormalization
- L2 Regularization (1e-4)
- MaxPooling at the end of each block

Block 1

- Conv2D — 64 filters
- Conv2D — 64 filters
- MaxPooling

Block 2

- Conv2D — 128 filters
- Conv2D — 128 filters
- MaxPooling

Block 3

- Conv2D — 256 filters × 4 layers
- MaxPooling

Block 4

- Conv2D — 512 filters × 4 layers
- MaxPooling

Block 5

- Conv2D — 512 filters × 4 layers
- MaxPooling

Fully Connected Layers

- Flatten
- Dense (4096) + BatchNorm + Dropout(0.5)
- Dense (4096) + BatchNorm + Dropout(0.5)
- Dense (26) + Softmax

Input / Output

- Input shape: 224×224×3
- Output: 26 classes (A-Z)

5. Hyperparameters & Training Environment

Hyperparameters

Parameter	Value
Optimizer	Adam
Learning Rate	1e-4
Regularization	L2 = 1e-4
Batch Size	16
Epochs	10
Loss Function	Categorical Crossentropy
Data Augmentation	Enabled
Early Stopping	patience = 6
LR Scheduler	ReduceLROnPlateau (factor=0.5)

6. Performance Metrics

computes the following:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix
- Class-wise ROC Curves & AUC

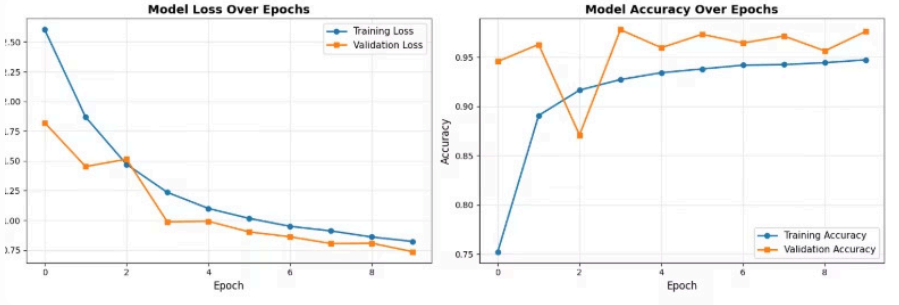
Reported Results

Metric	Score
Accuracy	~0.9762
Macro Precision	~0.9622
Macro Recall	~0.9790
Macro F1	~0.9700

7. Quantitative Analysis

Loss & Accuracy Curves

- Track learning behavior
- Detect underfitting or overfitting



2. Classification Report

Includes:

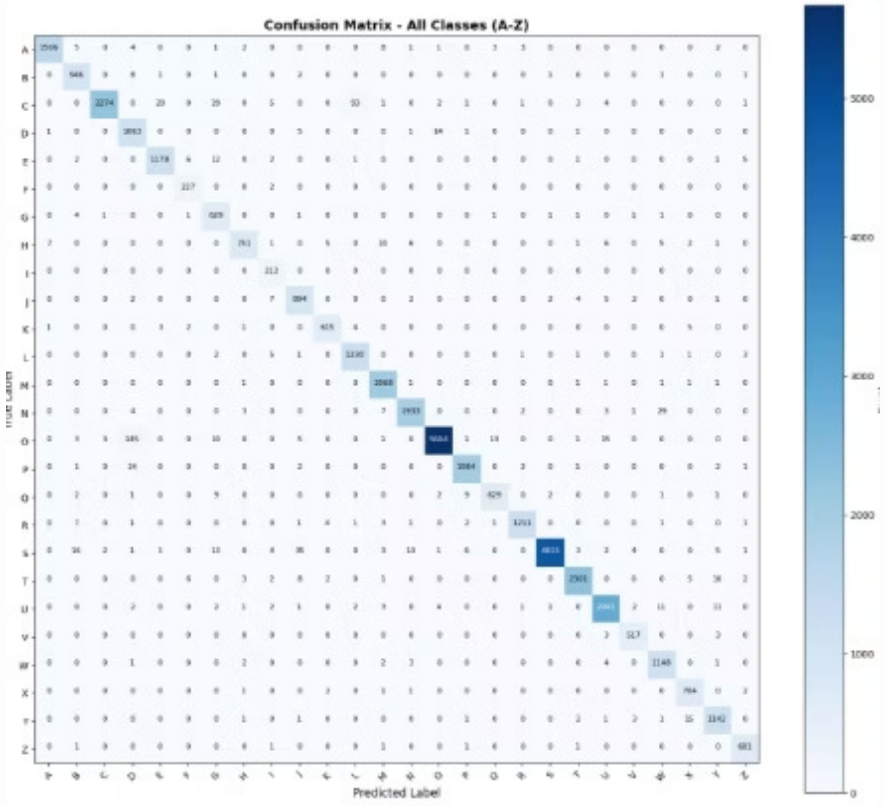
- Precision
- Recall
- F1
- Support

DETAILED CLASSIFICATION REPORT:

	precision	recall	f1-score	support
0	0.99	0.98	0.99	1536
1	0.96	0.98	0.97	961
2	1.00	0.93	0.96	2434
3	0.85	0.98	0.91	1086
4	0.98	0.98	0.98	1288
5	0.94	0.99	0.96	229
6	0.89	0.98	0.93	641
7	0.98	0.94	0.96	795
8	0.87	1.00	0.93	212
9	0.94	0.97	0.95	919
10	0.98	0.97	0.98	631
11	0.92	0.99	0.95	1245
12	0.98	1.00	0.99	2075
13	0.99	0.98	0.98	1982
14	1.00	0.97	0.98	5861
15	0.99	0.98	0.99	2017
16	0.97	0.96	0.97	656
17	0.99	0.98	0.99	1234
18	1.00	0.98	0.99	4025
19	0.99	0.98	0.99	2340
20	0.99	0.99	0.99	2084
21	0.98	0.99	0.98	523
22	0.96	0.99	0.97	1161
23	0.96	0.99	0.98	711
24	0.97	0.98	0.97	1167
25	0.98	0.99	0.98	686
accuracy			0.98	48219
macro avg	0.96	0.98	0.97	48219
weighted avg	0.98	0.98	0.98	48219

3. Confusion Matrix

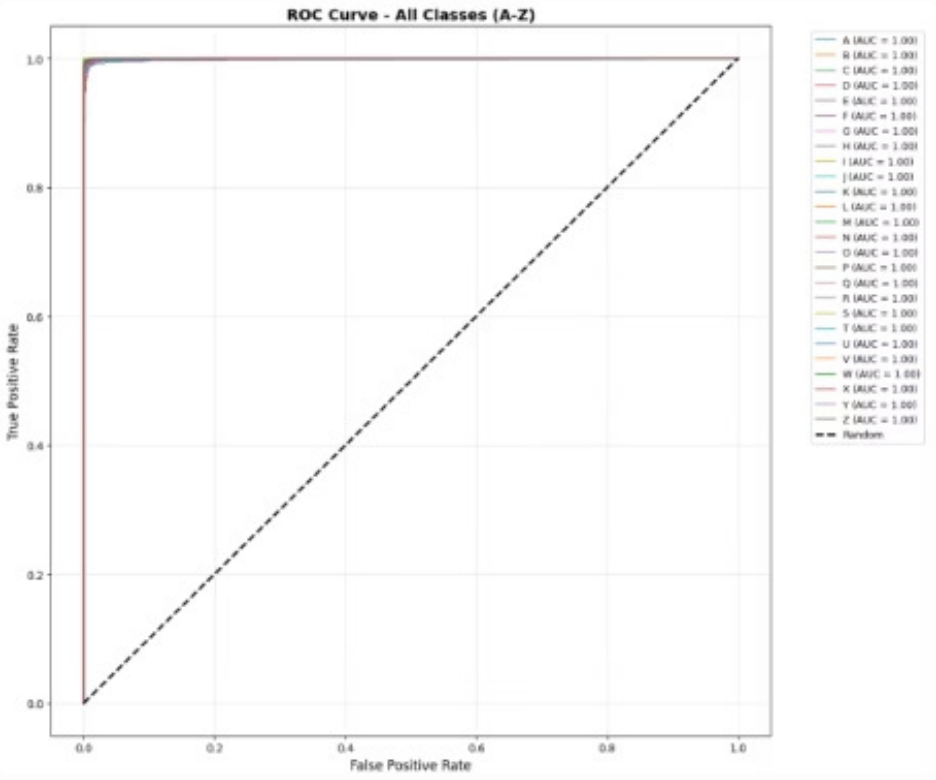
Shows class-to-class errors for all 26 letters.



4. ROC-AUC Analysis

Each class has:

- FPR
- TPR
- Class-wise AUC



Original VGG Research Paper

Simonyan, K., & Zisserman, A. (2014).

Very Deep Convolutional Networks for Large-Scale Image Recognition.

arXiv.org

Very Deep Convolutional Networks for Large-Scale Image Recogni...

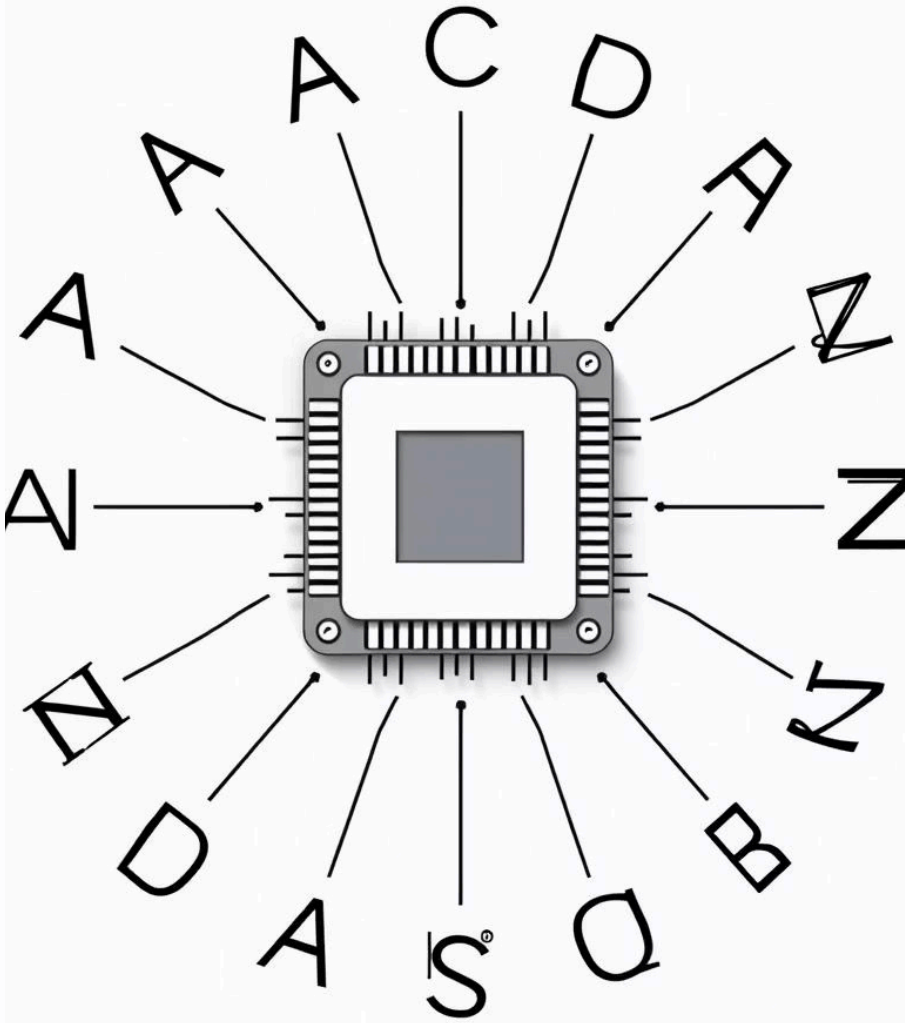
In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main...

Handwritten Alphabet Classifier (ResNet50)

Task: Handwritten Alphabet Classification (A–Z)

1. Executive Summary

This project presents a Deep Learning classifier capable of recognizing 26 handwritten English letters (A-Z) with an accuracy of approximately 93%. The model is built using Transfer Learning with ResNet50, incorporating dataset balancing, preprocessing, regularization, and advanced evaluation metrics.



2. Dataset & Preprocessing

2.1 Dataset Details

- Name: A_Z Handwritten Data.csv
- Format: CSV with flattened pixel values (784 features + label)
- Image Size: 28 × 28 grayscale
- Number of Classes: 26

2.2 Data Cleaning & Balancing

To avoid bias and produce a uniform dataset:

- ✔️ Duplicate images removed
- ✔️ Target samples per class: 3000 images
- ✔️ Minority classes augmented using:
  - Rotation (±15°)
  - Zoom (20%)
  - Width/height shift (10%)
- ✔️ Majority classes downsampled to reduce memory usage and improve training speed

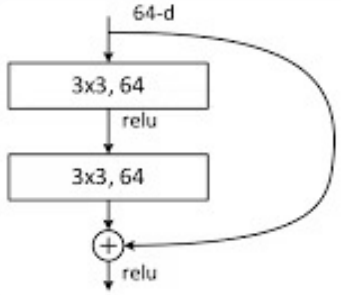
2.3 ResNet-Compatible Preprocessing

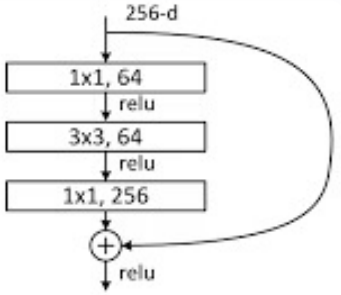
Since ResNet50 expects RGB ImageNet-like inputs, the following transformations were applied:

- Resizing: 28×28 → 32×32
- Channel Expansion: 1-channel grayscale → 3-channel RGB
- Normalization: Using preprocess_input (ImageNet mean subtraction)
- Label Encoding: One-hot vectors (length 26)

3. Model Architecture

The model uses transfer learning with a frozen ResNet50 feature extractor and a custom classification head.





3.1 ResNet-50 (Base Model)

- Type: Residual Network (50 layers)
- Pretrained On: ImageNet
- Input Shape: (32, 32, 3)
- Trainable: ❌ Frozen

Residual Learning (Core Concept)

ResNet solves:

- Vanishing gradients
- Degradation in deep networks

Instead of learning $H(x)$ each block learns a residual function $F(x)$ so:

$$H(x) = F(x) + x$$

Skip Connections

These allow:

- Better gradient flow
- Faster and more stable training
- Extremely deep architectures

3.2 Bottleneck Block (ResNet-50 Specific)

A bottleneck block has:

$$[1 \times 1, 64] \rightarrow [3 \times 3, 64] \rightarrow [1 \times 1, 256]$$

- 1×1 conv (reduce dimensions)
- 3×3 conv (feature extraction)
- 1×1 conv (restore dimensions)
- Skip connection adds input directly to output

3.3 ResNet-50 Composition (Full Architecture)

Stage	Description
Conv1	7×7 conv, 64 filters
Max Pool	3×3
Conv2_x	3 Bottleneck Blocks
Conv3_x	4 Bottleneck Blocks
Conv4_x	6 Bottleneck Blocks
Conv5_x	3 Bottleneck Blocks
GAP	Global Average Pool
FC	Dense layer
Softmax	Final classification

3.4 Custom Classification Head

- GlobalAvgPooling2D
- Dense Layer (512, ReLU, L2 regularization)
- Dropout (0.5)
- Dense Layer (26, Softmax)

Summary

[Input 32x32x3]

↓

[ResNet50 Base (Frozen)]

↓

[Global Average Pooling]

↓

[Dense 512 + ReLU + L2]

↓

[Dropout 0.5]

↓

[Dense 26 + Softmax]

4. Training Configuration

4.1 Hyperparameters

- Optimizer: Adam
- Learning Rate: 1e-4
- Loss: Categorical Crossentropy
- Batch Size: 64
- Epochs: 15
- Gradient Clipping: clipnorm = 1.0

4.2 Callbacks

- EarlyStopping: patience = 3
- ReduceLROnPlateau: factor = 0.5, patience = 2

5. Performance Metrics

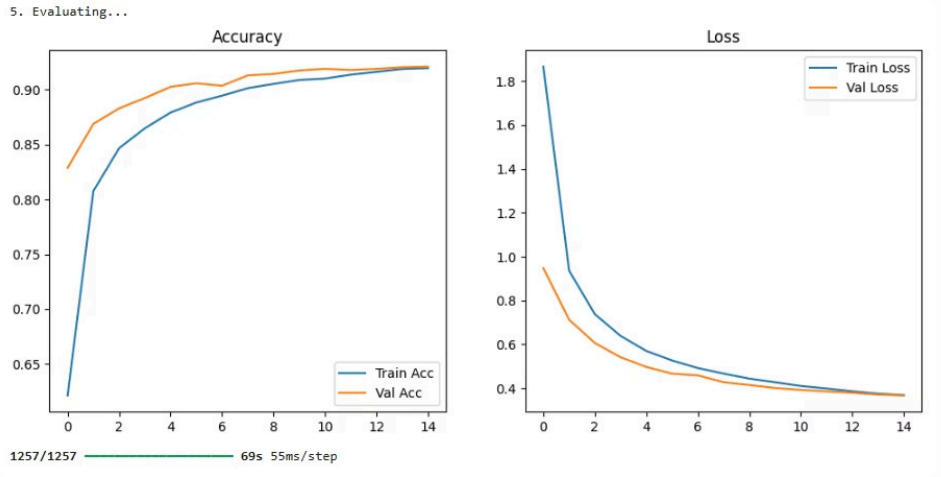
5.1 Overall Evaluation

Metric	Score
Training Accuracy	~92%
Validation Accuracy	~92%
Test Accuracy	~93%
Validation Loss	~0.367

6. Advanced Evaluation

6.1 Loss & Accuracy Curves

- Track learning behavior
- Detect underfitting or overfitting



6.2 Classification Report

Includes metrics per letter:

- Precision → Prediction correctness
- Recall → Detection rate
- F1-score → Balanced performance
- Support → Count per class

Reveals difficult letters (e.g., O/Q, C/G).Your model showed stable convergence

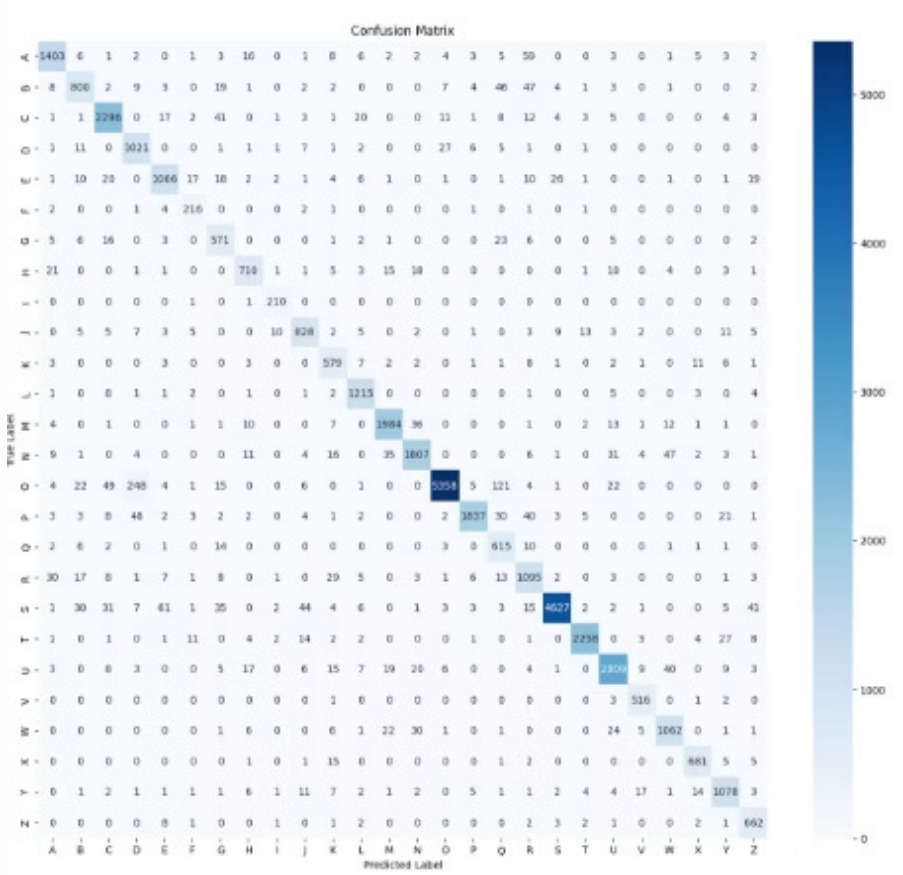
Classification Report:

	precision	recall	f1-score	support
A	0.93	0.91	0.92	1536
B	0.87	0.83	0.85	961
C	0.93	0.94	0.94	2434
D	0.75	0.94	0.84	1086
E	0.90	0.88	0.89	1208
F	0.82	0.94	0.88	229
G	0.78	0.89	0.83	641
H	0.90	0.89	0.89	795
I	0.91	0.99	0.95	212
J	0.88	0.90	0.89	919
K	0.82	0.92	0.86	631
L	0.94	0.98	0.96	1245
M	0.95	0.96	0.95	2075
N	0.94	0.91	0.93	1982
O	0.99	0.91	0.95	5861
P	0.98	0.91	0.94	2017
Q	0.70	0.94	0.80	656
R	0.82	0.89	0.85	1234
S	0.99	0.94	0.96	4925
T	0.98	0.96	0.97	2340
U	0.95	0.94	0.95	2984
V	0.92	0.99	0.95	523
W	0.91	0.91	0.91	1161
X	0.94	0.96	0.95	711
Y	0.91	0.92	0.92	1167
Z	0.86	0.97	0.91	686
accuracy			0.93	40219
macro avg	0.90	0.93	0.91	40219
weighted avg	0.93	0.93	0.93	40219

6.3 Confusion Matrix

- Diagonal dominance → strong performance
- Off-diagonal values show confusing pairs

Essential for multi-class tasks

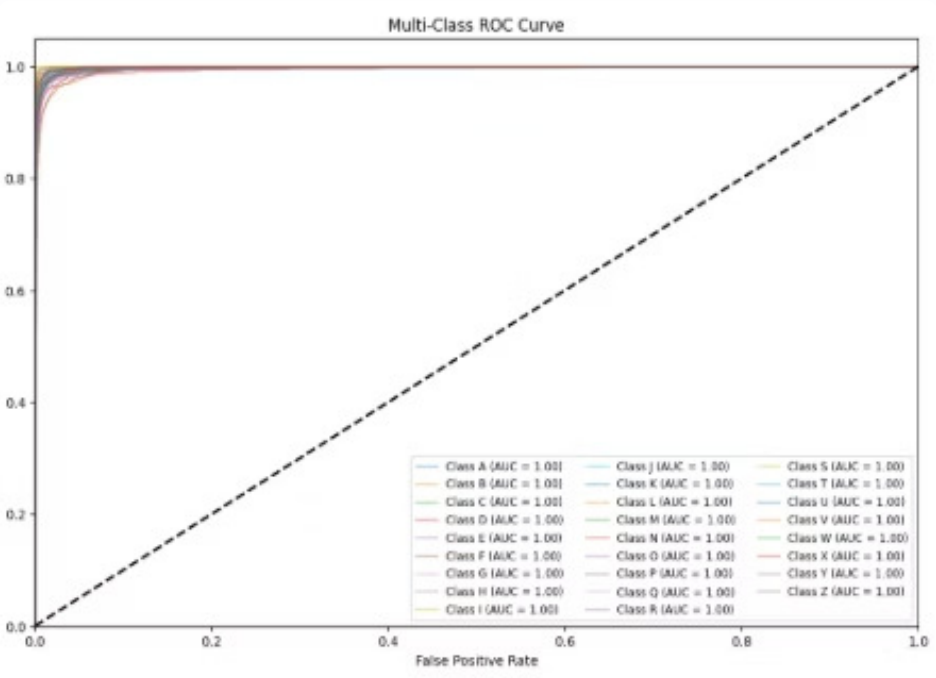


6.4 Multi-Class ROC Curve

For each of 26 classes:

- Plots TPR vs FPR
- Measures separability using AUC

Shows which letters the model distinguishes well



7. References

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition (CVPR).
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity Mappings in Deep Residual Networks (ECCV).

InceptionV1 (GoogLeNet) Handwritten Alphabet Classifier

1.Model Details & Core Concept

Model Name: InceptionV1 (GoogLeNet) – Transfer Learning Variant

Core Concept: The model leverages a pre-trained InceptionV1 architecture from TensorFlow Hub to extract high-level features from images, followed by a small fully connected head for classification into 26 classes (A–Z). The original InceptionV1 design is aimed at capturing multi-scale features efficiently through Inception modules.

Key Innovation:

- **Inception Module:** Performs parallel convolutions of different sizes (1×1, 3×3, 5×5) and 3×3 max pooling, followed by concatenation.
- **Dimensionality Reduction:** 1×1 convolutions reduce input channels before expensive convolutions, improving efficiency.
- **Global Average Pooling:** Reduces parameters in the original architecture.
- **Auxiliary Classifiers:** Improve gradient flow during training (not used in this transfer learning variant).

Intended Use:

- Handwritten alphabet recognition (26 classes, A–Z)
- Adaptable to other image classification tasks via transfer learning

Training Data (for the feature extractor):

- Source: ImageNet (1.2M images, 1000 classes)

Code Architecture Overview

The project is structured in the following logical stages:

- **Imports & Setup:** Loads essential libraries (TensorFlow, TensorFlow Hub, sklearn, matplotlib, seaborn) and clears memory/GPU cache.
- **Configuration:** Centralized hyperparameters stored in Config class:

```
IMG_HEIGHT = 224
IMG_WIDTH = 224
CHANNELS = 3
BATCH_SIZE = 32
LEARNING_RATE = 0.0001
EPOCHS = 5
NUM_CLASSES = 26
```

- **Data Loading & Preprocessing:**
- Loads the A-Z Handwritten dataset CSV.
- Cleans data (removes duplicates, fills NaNs).
- Reshapes 28×28 grayscale images to 28×28×1.
- Splits into train (80%), validation (10%), test (10%) using stratified splits.
- Visualizes class distributions and sample images.
- **TF.Data Pipeline:**
- Resizes images to 224×224 (required by InceptionV1).
- Converts grayscale images to RGB.
- One-hot encodes labels.
- Builds efficient pipelines with batching, shuffling, and prefetching.
- **Model Construction (Transfer Learning):**
- Loads pre-trained InceptionV1 from TensorFlow Hub.
- Adds a custom classification head:

```
Dense(256, ReLU) → Dropout(0.3) → Dense(26, Softmax)
```

- Compiles the model using Adam optimizer and categorical cross-entropy.
- **Training:**
- Uses EarlyStopping (patience=3) and ReduceLROnPlateau (factor=0.5, patience=2) callbacks.
- Trains for 5 epochs (configurable).
- Monitors validation accuracy and loss.
- **Evaluation:**
- Plots training curves for accuracy and loss.
- Generates predictions on the test set.
- Computes classification report and confusion matrix.
- Provides detailed analysis with per-class accuracy and visualizations.

3.Step-by-Step Architecture

Input Preprocessing

Input: 224×224×3 RGB images (rescaled to [0,1])

Feature Extraction (InceptionV1)

Extracts high-level features using pre-trained InceptionV1 feature vector.

Does not include original Inception auxiliary classifiers.

Classification Head

Dense layer: 256 units, ReLU

Dropout: 0.3

Dense layer: 26 units, Softmax (output probabilities for each letter)

Output

Softmax probability vector of shape (26,) for each input image.

4.Hyperparameters & Training Environment

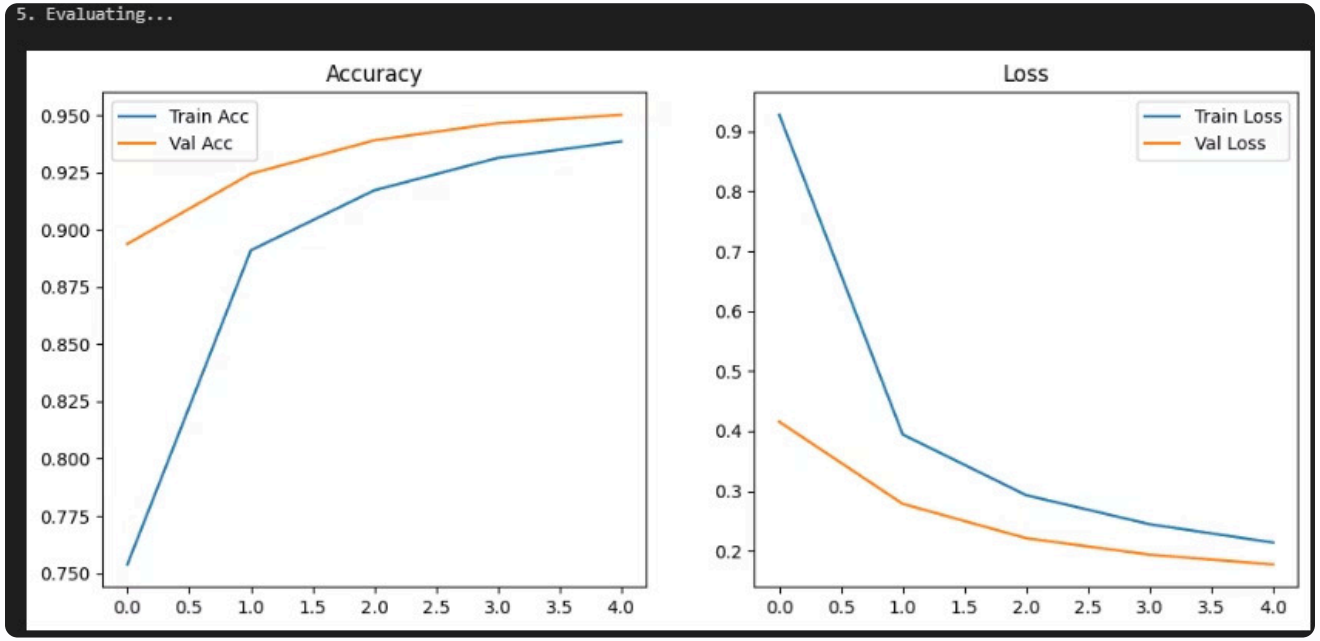
Parameter	Value
Input Size	224 × 224 × 3
Batch Size	32
Learning Rate	0.0001
Epochs	5
Optimizer	Adam
Loss	Categorical Cross-Entropy
Dropout Rate	0.3
Callbacks	EarlyStopping, ReduceLROnPlateau
Hardware	GPU (if available)

5.Performance Metrics

Metric	Value / Observation
Test Accuracy	Computed on final test set (reported in evaluation)
Loss	Computed on final test set
Classification Report	Provides precision, recall, F1-score per class
Confusion Matrix	Visualizes correct/incorrect predictions per class

6.Visualization:

- Accuracy & Loss curves for training vs validation



Classification Report

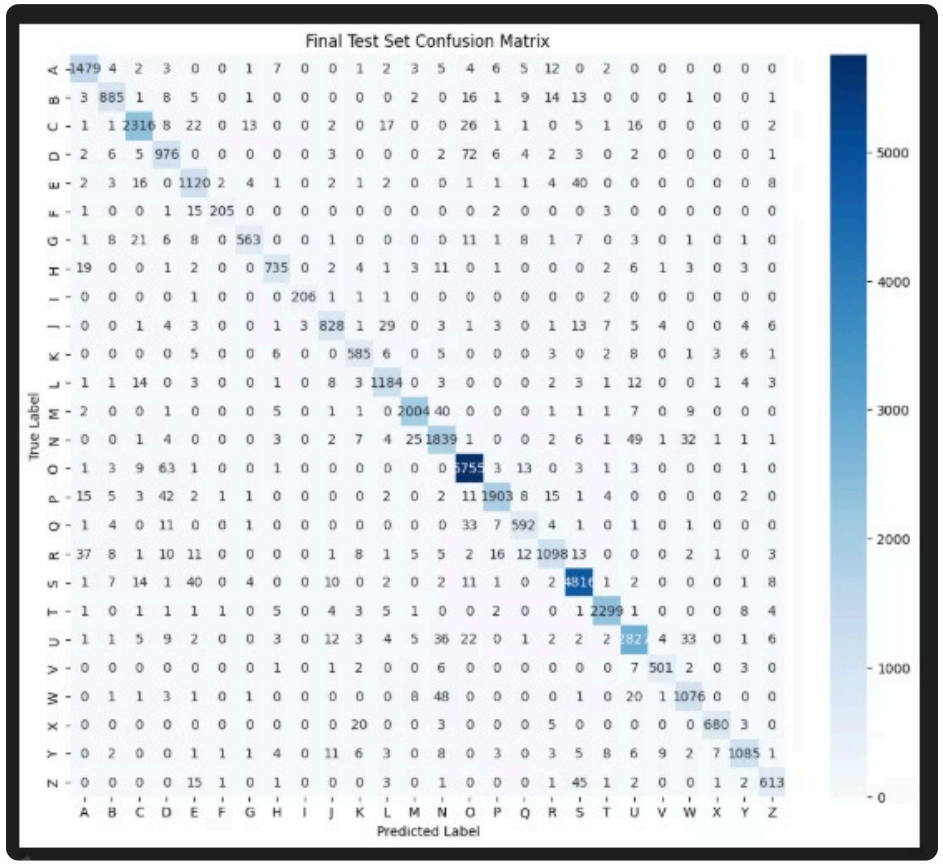
Includes:

- Precision
- Recall
- F1
- Support

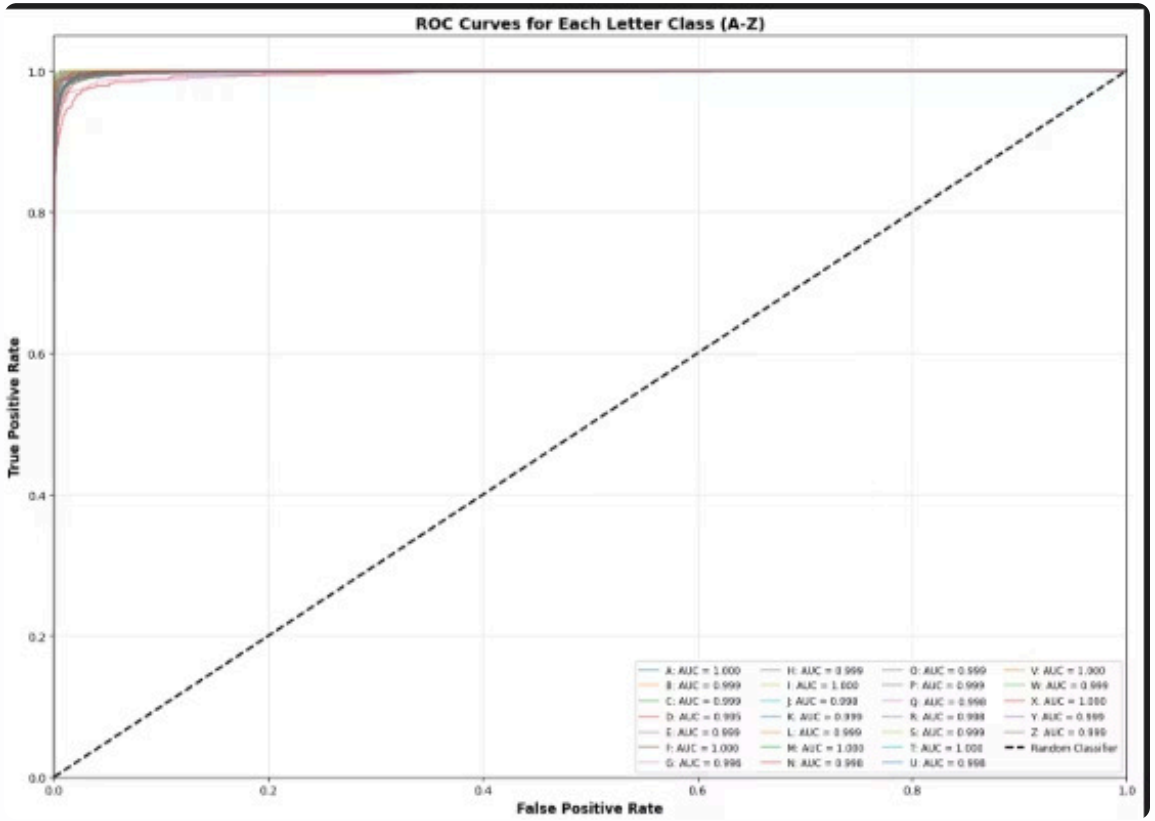
Classification Report:

	precision	recall	f1-score	support
A	0.94	0.96	0.95	1536
B	0.94	0.92	0.93	960
C	0.96	0.95	0.96	2432
D	0.85	0.90	0.87	1084
E	0.89	0.93	0.91	1208
F	0.97	0.90	0.94	227
G	0.95	0.88	0.91	641
H	0.95	0.93	0.94	794
I	0.99	0.97	0.98	212
J	0.93	0.90	0.92	917
K	0.91	0.93	0.92	631
L	0.94	0.95	0.94	1244
M	0.97	0.97	0.97	2073
N	0.91	0.93	0.92	1980
O	0.96	0.98	0.97	5857
P	0.97	0.94	0.96	2017
Q	0.91	0.90	0.90	656
R	0.94	0.89	0.91	1234
S	0.97	0.98	0.97	4923
T	0.98	0.98	0.98	2338
...				
accuracy			0.95	40192
macro avg	0.94	0.94	0.94	40192
weighted avg	0.95	0.95	0.95	40192

- Heatmaps for confusion matrices



ROC curves



7.Benefits of Using Transfer Learning

Efficiency

Avoids training InceptionV1 from scratch.

High Accuracy

Leverages pre-trained ImageNet weights.

Reduced Overfitting

Fewer trainable parameters in classification head.

Flexibility

Can adapt to any image classification task by changing the output head.

8.Limitations

- Dataset images are originally 28×28, upscaled to 224×224, which may introduce interpolation artifacts.
- Limited epochs (5) may restrict performance; longer training can improve results.
- Dependent on TensorFlow Hub; requires internet connection for initial load.

9.Original Research Paper Reference

Title	Authors	Publication/Archive
Going Deeper with Convolutions	Szegedy, C., Liu, W., Jia, Y., et al.	arXiv:1409.4842 (2014)

MobileNet Model
Handwritten Alphabet Classifier

Model Details & Intended Use

Model Name: MobileNetV1 (pre-trained on ImageNet, used with Transfer Learning)

Intended Use: Efficient multi-class image classification of handwritten alphabet letters (A–Z) on grayscale images. The model is optimized for resource-constrained environments and can be deployed on GPUs, edge devices, or mobile systems.

Key Advantages & Target Environment

Key Advantages:

- High accuracy on small datasets with transfer learning.
- Efficient computation via depthwise separable convolutions.
- Small memory footprint suitable for real-time inference.

Target Environment:

- Kaggle/TensorFlow runtime for training.
- Potential deployment on embedded devices or mobile apps after quantization/optimization.

2.Selected Architecture: Depthwise Separable Convolution

The core MobileNet innovation is the Depthwise Separable Convolution, which splits standard convolutions into two efficient steps:

2.1 Standard Convolution

Applies a $D\_K \times D\_K$ kernel across all input channels, producing each output channel via a full convolution.

Computation Cost: $O(D\_K^2 \cdot M \cdot N)$ — expensive for large channels.

2.2 MobileNet Factorization

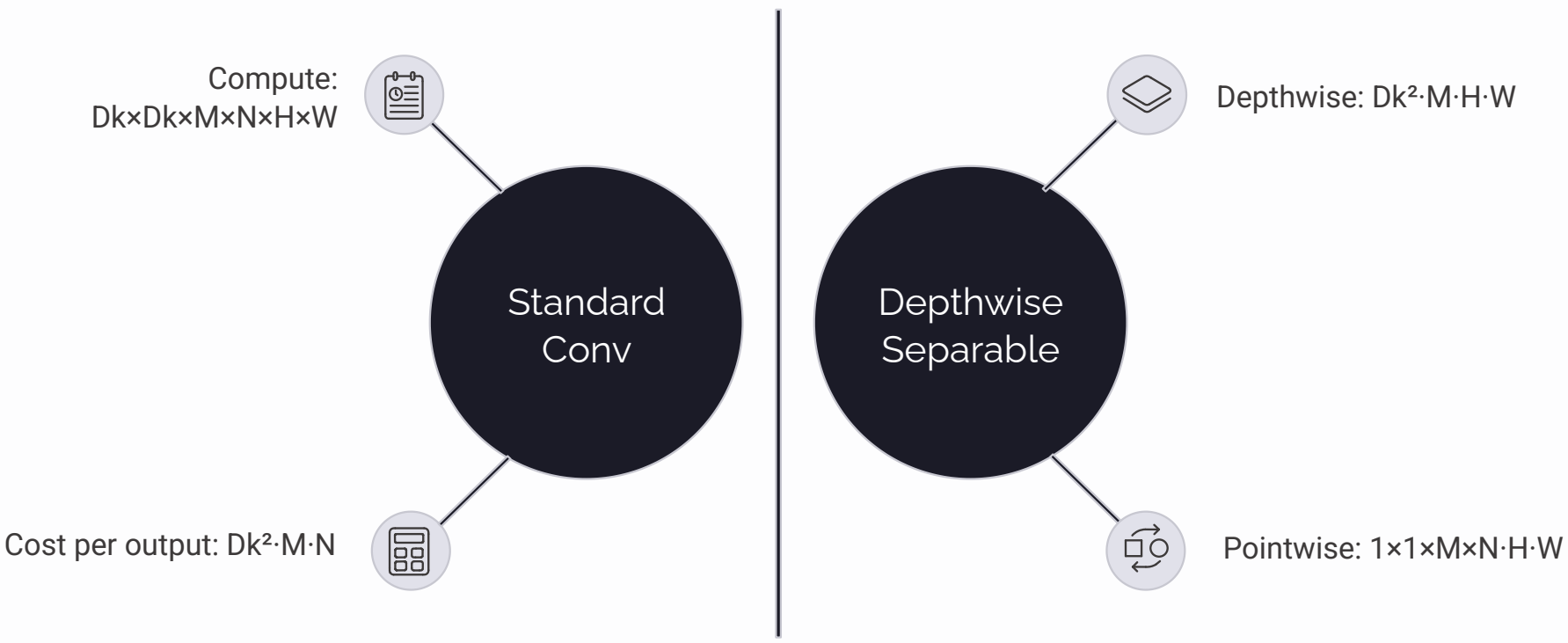
Step 1: Depthwise Convolution

- Applies a separate $D\_K \times D\_K$ filter per input channel.
- Outputs M feature maps.
- Computation Cost: $O(D\_K^2 \cdot M)$.

Step 2: Pointwise Convolution (1×1 Conv)

- Combines the M channels into N output channels.
- Performs channel mixing efficiently.
- Computation Cost: $O(M \cdot N)$.

Total Cost: $O(M \cdot (D\_K^2 + N))$ **Efficiency:** ~8–9× less computation than standard convolution for typical 3×3 kernels.



3.Model Implementation & Architecture Details

Input:

- Grayscale 28×28 images resized to 224×224×3 for MobileNet compatibility.
- Normalized to \[0, 1\].

Data Preprocessing:

- Conversion to RGB channels for MobileNet.
- Data augmentation for training: rotation, zoom, translation.

Architecture Layers:

Base Model	MobileNetV1 (weights='imagenet', top excluded)	Frozen during Stage 1
Global Average Pooling	-	Converts feature maps to vector
Dense Layer	512 units, ReLU	Adds capacity for task-specific features
Dropout	0.5	Regularization
Dense Layer	256 units, ReLU	Additional representation
Dropout	0.3	Regularization
Output Layer	26 units, Softmax	A-Z classification

Total Parameters: 3,891,674 **Trainable Parameters (Stage 1):** 662,810

4.Training Configuration

Optimizer

Adam (learning_rate=0.001)

Loss Function

Sparse categorical crossentropy

Batch Size

32

Epochs

10 (Stage 1)

Callbacks:

- EarlyStopping (patience=7)
- ReduceLROnPlateau (factor=0.5, patience=3, min_lr=1e-7)

Dataset Split:

- Train: 64%
- Validation: 16%
- Test: 20%

5.Hyperparameters for MobileNet

Width Multiplier (α)	1.0	Uses full depth of MobileNet channels
Input Resolution	224×224	Standard MobileNet input
Dropout Rates	0.5, 0.3	Regularization for Dense layers

6.Evaluation Metrics

98.97%	0.0416	0.9870	0.9877
Test Accuracy	Test Loss	Macro Precision	Macro Recall

0.9873	0.9999	0.9999
Macro F1-Score	Micro-average AUC	Macro-average AUC

Per-class AUC:

- Nearly 1.0 for all 26 letters (A–Z)
- Indicates excellent discriminative performance across all classes

Observations:

- Confusion matrices show very low misclassification.
- ROC curves demonstrate near-perfect class separation.

7.Data Pipeline & Visualizations

Data Pipeline

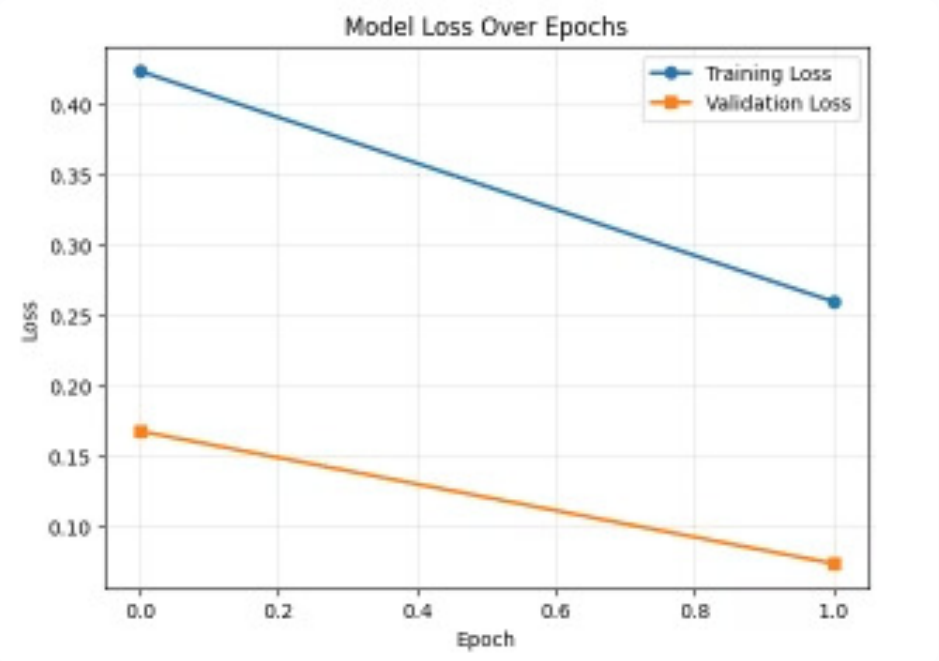
- Training: Shuffled, batched, resized, RGB-converted, augmented.
- Validation/Test: Shuffled (validation), batched, resized, RGB-converted, no augmentation.
- Prefetching: Applied for performance optimization.

Visualizations

- Loss and accuracy curves demonstrate stable convergence.
- Data augmentation illustrated via random rotation, zoom, and translation.
- Confusion matrices (raw and normalized) show low misclassification.
- ROC curves (per-class, micro/macro average) confirm excellent classification boundaries.

8. Visualizations

- Loss and accuracy curves demonstrate stable convergence.



2. Classification Report

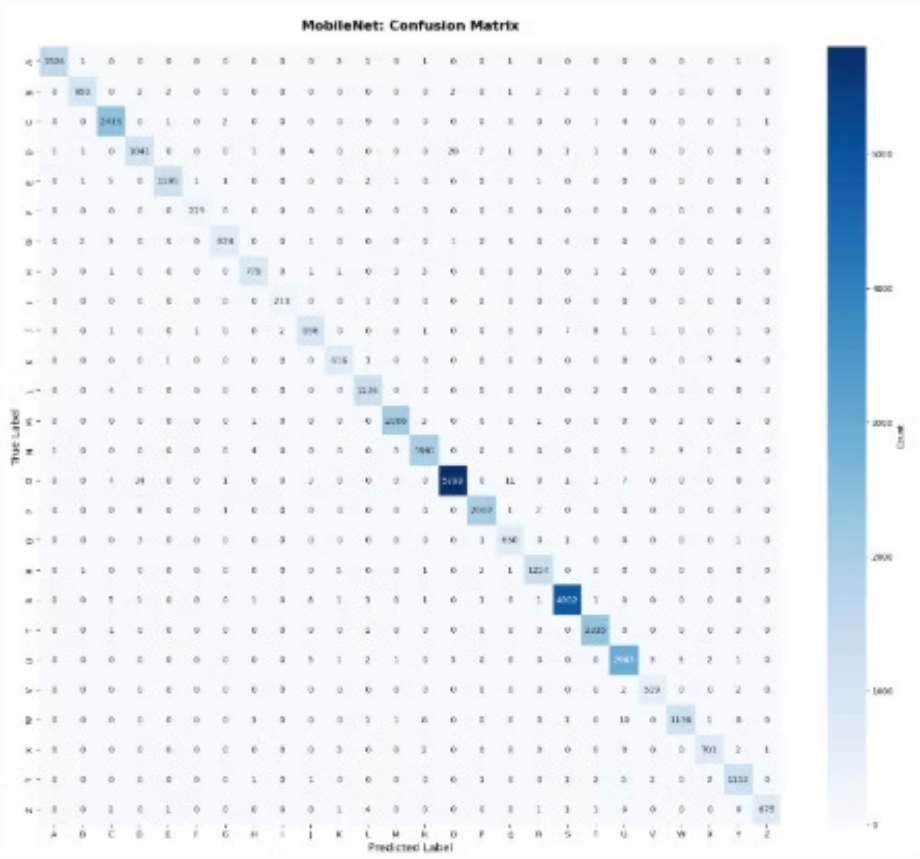
Includes:

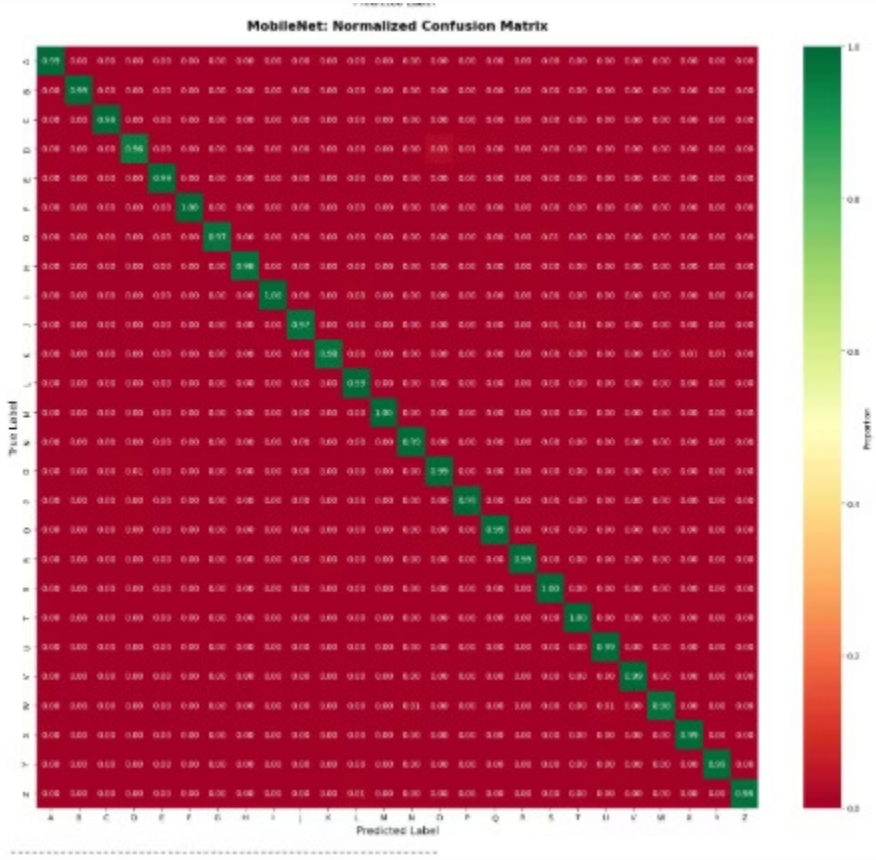
- Precision
- Recall
- F1
- Support

CLASSIFICATION REPORT

	precision	recall	f1-score	support
A	1.00	0.99	0.99	1536
B	0.99	0.99	0.99	961
C	0.99	0.99	0.99	2434
D	0.96	0.96	0.96	1086
E	0.99	0.99	0.99	1288
F	0.99	1.00	1.00	229
G	0.99	0.97	0.98	641
H	0.99	0.98	0.98	795
I	0.99	1.00	0.99	212
J	0.97	0.97	0.97	919
K	0.98	0.98	0.98	631
L	0.98	0.99	0.99	1245
M	0.99	1.00	1.00	2875
N	0.99	0.99	0.99	1982
O	0.99	0.99	0.99	5861
P	0.99	0.99	0.99	2017
Q	0.97	0.99	0.98	656
R	0.99	0.99	0.99	1234
S	1.00	1.00	1.00	4925
T	0.99	1.00	1.00	2340
U	0.99	0.99	0.99	2084
V	0.98	0.99	0.99	523
W	0.99	0.98	0.99	1161
X	0.98	0.99	0.99	711
Y	0.98	0.99	0.98	1167
Z	0.99	0.98	0.99	686
accuracy			0.99	48219
macro avg	0.99	0.99	0.99	48219
weighted avg	0.99	0.99	0.99	48219

Confusion matrices

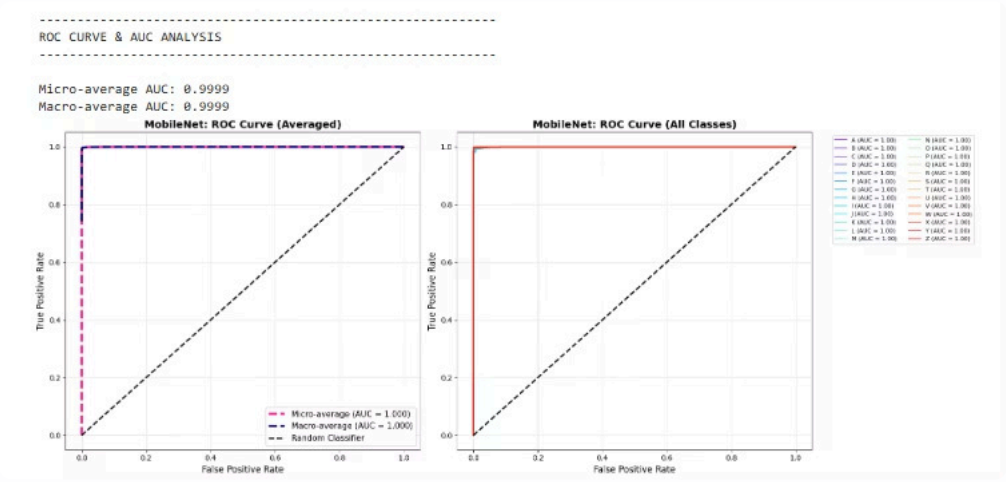




ROC curves

per-class, micro/macro average

confirm excellent classification boundaries.



9.Advantages over ResNet for This Task

Advantages over ResNet for This Task

- MobileNet is lightweight and faster on small datasets like handwritten alphabets.
- Requires less GPU memory than ResNet while maintaining high accuracy.
- Transfer learning from ImageNet provides strong feature extraction despite small grayscale input.

10.Original Research Paper References

MobileNetV1	MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications	Howard, A. G., et al.	arXiv:1704.04861 (2017)
MobileNetV2	MobileNetV2: Inverted Residuals and Linear Bottlenecks	Sandler, M., et al.	arXiv:1801.04381 (2018)
MobileNetV3	Searching for MobileNetV3	Howard, A., et al.	arXiv:1905.02244 (2019)

Comparative Analysis of CNN Architectures

VGG-19 – InceptionV1 – ResNet-50 – MobileNetV1

This comparative analysis provides a structured overview of four foundational CNN architectures, highlighting their performance, computational complexity, strengths, limitations, and ideal use cases.

1. Performance & Complexity Comparison

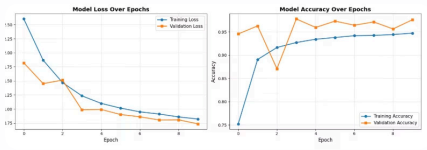
The table below summarizes each model's key characteristics, using ImageNet benchmark results as reference.

Model	Key Innovation	Accuracy	Precision	Recall	F1-score	Computational Cost
VGG16	Uniform 3×3 Conv Stacking	0.9762	0.9622	0.9790	0.9700	Very high
InceptionV1 (GoogLeNet)	Inception Module + 1×1 Reduction	0.95	0.94	0.94	0.94	Medium
ResNet-50	Residual / Skip Connections	0.93	0.90	0.93	0.91	Medium-High
MobileNetV1	Depthwise Separable Convolution	0.99	0.9870	0.9877	0.9873	Very low

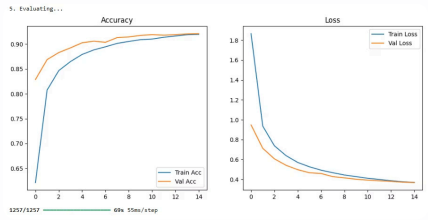
2. Model Complexity & Efficiency

Model	Parameters	FLOPs	Speed	Suitability
VGG	★★★★★ huge	★★★★★ very high	Slow	Servers / high-power GPUs
Inception	★★ medium	★★ moderate	Fast	Real-time + GPU
ResNet	★★★ medium-high	★★★ high	Moderate	Most ML tasks
MobileNet	★ lowest	★ lowest	Very fast	Mobile & embedded systems

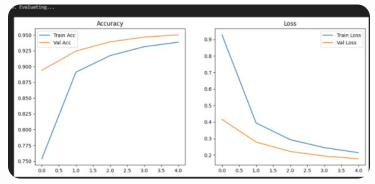
Loss&Accuracy



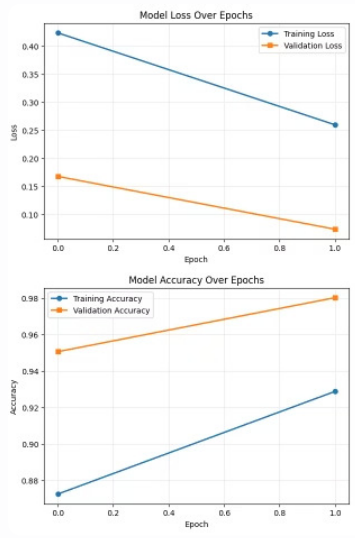
Vgg-19



ResNet



Inception



MobileNet

3. Pros & Cons of Each Architecture

1. VGG (VGG16/19)

Pros

- Extremely simple and uniform design; easy to understand and implement.
- Excellent as a **feature extractor** in transfer learning due to its deep, clean structure.

Cons

- Very high number of parameters → large model size and high memory usage.
- Highest FLOPs → slow inference and heavy computational cost.
- Prone to overfitting on smaller datasets.

2. InceptionV1 (GoogLeNet)

Pros

- Highly efficient: achieves strong accuracy with a very small number of parameters.
- Multi-scale feature extraction using parallel 1×1, 3×3, and 5×5 filters.
- 1×1 convolutions reduce dimensionality, lowering computation.

Cons

- More complex architecture with parallel branches, requiring careful implementation.
- Performance depends heavily on channel configuration in reduction (1×1) layers.

3. ResNet-50

Pros

- Skip connections solve the vanishing gradient problem, enabling very deep training.
- Typically provides the **highest accuracy** among the compared architectures.
- Depth allows capturing highly abstract and hierarchical features.

Cons

- Requires strong GPU/TPU hardware for efficient training.
- Bottleneck blocks make it more complex than VGG and MobileNet.

4. MobileNetV1

Pros

- Extremely lightweight and fast → ideal for mobile and embedded systems.
- Lowest number of parameters and FLOPs of all four models.
- Highly tunable through **Width** and **Resolution Multipliers**, depending on hardware limits.

Cons

- Lower peak accuracy compared to ResNet and Inception.
- Depthwise convolution reduces cross-channel information flow compared to full convolutions.

3. Explanation of Model Selection

Choosing the “best” architecture depends entirely on your task constraints and available resources.

Case 1: When Highest Accuracy Is the Priority → Choose ResNet-50

Best for:

- Complex classification tasks
- High-resolution images
- Medical imaging
- Industrial inspection
- Large datasets

Why ResNet-50?

- Skip connections allow deep and stable training.
- Extracts highly abstract, powerful features.
- Achieves the **lowest error rate** among the four models.

Case 2: When Computational Efficiency Matters → Choose MobileNetV1

Best for:

- Mobile applications
- Drones
- IoT devices
- Real-time systems with limited power

Why MobileNetV1?

- Depthwise Separable Convolutions reduce computation by almost **9×**.
- Lowest latency and smallest model size.
- Ideal for real-time inference.

Case 3: When Multi-Scale Feature Extraction Is Needed → Choose InceptionV1

Best for:

- Satellite imagery
- Microscopy
- Scene parsing
- Images containing objects of varying sizes

Why InceptionV1?

- Parallel 1×1, 3×3, and 5×5 filters capture features at different scales.
- Very efficient while maintaining strong accuracy.
- More compact than VGG while offering richer spatial features.