Comparative Analysis of Handwritten Alphabet Recognition Models

1. Overview

This document presents a comparative analysis of four deep learning architectures for handwritten alphabet (A-Z) recognition using the **A-Z Handwritten Data.csv** dataset (28x28 grayscale images, 26 classes).

2. Model Performance Summary

Model	Test Accuracy	Precision	Recall	F1-Score	AUC (Avg)	Parameters
MobileNet (Custom)	**99.03%**	**98.66%**	**98.74%**	**98.69%**	**~0.999**	**3.27M**
VGG19 (Scratch)	98.47%	97.88%	97.91%	97.88%	~0.998	3.19M
ResNet50V2	97.06%	96.40%	95.62%	95.98%	~0.997	24.68M
InceptionV1 (GoogLeNet)	95.27%	94.72%	93.25%	93.94%	~0.995	6.20M

3. Detailed Model Analysis

3.1 VGG19 (Built from Scratch)

Architecture: Classical deep CNN with small 3×3 filters, multiple convolutional blocks, and fully connected layers.

Pros:

- Simple, interpretable architecture
- Excellent performance (98.47%) despite being built from scratch
- Minimal preprocessing required (28x28 grayscale input)
- Lower parameter count (3.19M) than pretrained models

Cons:

- Less efficient than MobileNet
- Can be prone to overfitting (requires careful regularization)
- No transfer learning benefits

Why it performed well:
The dataset (28x28 images) is well-suited for VGG's hierarchical feature extraction. The model depth (16 layers) captures intricate patterns in handwritten characters effectively without being too complex for the task.

3.2 ResNet50V2 (Pretrained)

Architecture: Deep residual network with 50 layers, skip connections, pretrained on ImageNet.

Pros:

- Transfer learning from ImageNet provides good feature extractors

- Residual connections help with gradient flow
- Robust to vanishing gradient problem

Cons:

- **Performance bottleneck**: Only 97.06% accuracy
- **Input mismatch**: Requires upsampling 28x28→224x224 and grayscale→RGB conversion
- **Parameter-heavy**: 24.68M parameters (89.89MB non-trainable)
- Pretrained features may not be optimal for simple grayscale characters

Why it underperformed:

1. **Domain mismatch**: ResNet was trained on natural color images (ImageNet), while handwritten characters are simple, high-contrast grayscale patterns
2. **Information loss**: Upscaling 28x28→224x224 creates artificial pixels without adding real information
3. **Overkill architecture**: 50-layer network is excessive for 28x28 character recognition

3.3 InceptionV1/GoogLeNet (Pretrained)

Architecture: Inception modules with parallel convolutions at different scales.

Pros:

- Multi-scale feature extraction
- Efficient parameter usage
- TF Hub integration simplifies implementation

Cons:

- **Lowest accuracy**: 95.27% among all models
- Similar input mismatch issues as ResNet
- Transfer learning benefits limited for this specific task

Why it underperformed:
Same issues as ResNet (domain mismatch, upscaling artifacts) combined with Inception's complexity being unnecessary for simple character recognition.

3.4 MobileNet (Custom Implementation)

Architecture: Lightweight CNN using depthwise separable convolutions.

Pros:

- **Best performance**: 99.03% accuracy (highest of all models)
- **Optimized for small images**: Input size 64x64 (better than 224x224 for ResNet/Inception)
- **Parameter efficient**: Depthwise separable convolutions reduce computation

- **No domain mismatch**: Trained from scratch on the actual dataset

Cons:

- Custom implementation required (not pretrained)
- More complex architecture than VGG
- Requires gradient clipping for stability

Why it performed best:

1. **Architecture suitability**: Depthwise separable convolutions are ideal for simple, high-contrast patterns
2. **Input size**: 64x64 is a better fit than 224x224 (ResNet/Inception) or 28x28 (VGG)
3. **Efficiency**: More computation spent on learning relevant features rather than unnecessary complexity
4. **Modern design**: Incorporates BatchNorm, ReLU, and proper regularization

4. Performance Hierarchy

1. **MobileNet (Custom)** → **Best choice** for accuracy and efficiency
2. **VGG19 (Scratch)** → Excellent alternative, simpler implementation
3. **ResNet50V2** → Overkill, not recommended for this task
4. **InceptionV1** → Least suitable for this dataset