CS471 Machine Learning

Comparative Analysis of AlexCapsNet and ResCapsNet on Singular and Distributed Computing

Name	CMS ID
Muhammad Zain	405749
Ahmed Umar	412678

Abstract:

We compare two capsule-network hybrid models on MNIST digit classification: **AlexCapsNet**, which adds capsule layers to an AlexNet-like CNN, and **ResCapsNet**, which adds ResNet-style residual blocks before the capsule layers. Both models replace the final classifier with a set of 10 digit capsules (16-dimensional vectors per class) using dynamic routing. Both were trained on single as well as multiple GPUS for better supposed

results. Our experiments show that both models achieve high accuracy on MNIST, with AlexCapsNet converging slightly faster due to its larger convolutional backbone, while ResCapsNet is more lightweight. We provide detailed architecture diagrams, training curves (accuracy and loss), and evaluation metrics (confusion matrices and ROC curves) for both models and explain how their performance, convergence, and complexity differ.

Introduction

Capsule networks (CapsNets) were proposed to overcome the limitations of standard CNNs by representing not only feature co-occurrence but also the spatial pose of parts and wholes. In Sabour et al.'s original CapsNet, patches of the image are first processed using a convolutional layer, then organized into primary capsules, which are dynamically routed to higher-level digit capsules. Capsules are vector representations whose lengths convey class probabilities, thus overcoming the loss of spatial information inherent in max-pooling CNNs.

Despite their strong representational power, CapsNets alone can be computationally expensive and slow to train. A natural solution is to combine capsule layers with state-of-the-art CNN backbones to benefit from powerful feature extraction. For instance, **AlexCapsNet** replaces Sabour's shallow CNN with an AlexNet-like backbone—a series of large convolutional layers before the capsules. This allows the network to learn deep semantic features through convolutions while retaining capsules for final classification. **ResCapsNet**, on the other hand, uses ResNet-style residual blocks (skip connections) to allow training of a deeper feature extractor before the capsule layers. Residual connections facilitate better gradient propagation and mitigate the vanishing gradient problem. In ResCapsNet, each ResidualBlock learns a function F(x) added to the input x, resulting in x+F(x). This hybrid architecture combines the strength of residual CNNs with the spatial modeling of capsules.

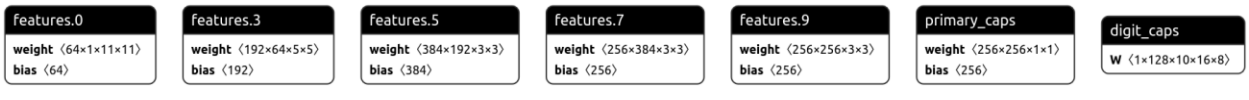
In this work, we use AlexCapsNet and ResCapsNet for MNIST digit recognition and compare them experimentally. We detail their architectures, training processes (including data preprocessing, optimizers, loss functions, and distributed/mixed-precision training), and measure performance through accuracy/loss curves, confusion matrices, and ROC curves. Finally, we analyze differences in convergence rate and model complexity.

Methodology

AlexCapsNet Architecture

AlexCapsNet enhances an AlexNet-style convolutional feature extractor by incorporating capsule layers. It processes 28×28 grayscale MNIST inputs as follows:

- **Convolutional Backbone:** Five convolutional layers (with ReLU activations and some max-pooling) produce a high-dimensional feature map. For instance, the first layer is `Conv2d(1→64, 11×11, stride=4, padding=2) → ReLU → MaxPool (3×3, stride=2)`, followed by additional convolutional layers (e.g., 64→192 (5×5), 192→384 (3×3), 384→256 (3×3), 256→256 (3×3)) with ReLU. This deep CNN, similar to AlexNet, is adapted to smaller input sizes and extracts rich image features.
- **Primary Capsules:** The final convolutional feature maps are converted into primary capsules. A `Conv2d(256→32×8, 1×1)` layer projects the 256-channel feature map into 32 channels of 8-dimensional capsules. The tensor is reshaped from `[batch, 256, H, W]` to `[batch, 32, 8, H, W]`, then flattened to `[batch, 32×H×W, 8]`. Each 8D vector is a primary capsule output.
- **Digit Capsules:** The primary capsules are routed to 10 digit capsules (one per class). A `CapsuleLayer` receives inputs defined by `in_capsules=32×H×W` and `in_dim=8`, and produces `out_capsules=10`, `out_dim=16` vectors via dynamic routing. The lengths of these 16D output vectors represent class scores. A capsule margin loss is used to train this output layer.



ResCapsNet Architecture

ResCapsNet replaces the AlexNet-style architecture with a ResNet-like feature extractor followed by capsule layers. Its components include:

- **Input Resizing:** MNIST 28×28 images are resized to 78×78 using bilinear interpolation to ensure that repeated convolutions and strides yield capsule-sized feature maps.
- **Initial Convolution:** A single `Conv2d(1→32, 3×3, stride=1, pad=1)` with batch normalization and ReLU extracts low-level features while preserving the 78×78 spatial size. The output tensor shape is `[B, 32, 78, 78]`.
- **Residual Blocks:** Three stacked residual blocks deepen the network. Each block consists of: `Conv1×1(32→32) → BN → ReLU → Conv3×3(32→32, pad=1) → BN → Add input → ReLU`. Each block maintains the shape `[B, 32, 78, 78]`. Skip connections enable deeper networks to train efficiently.
- **Primary Capsules:** A `Conv2d(32→16×8, 9×9, stride=8)` reduces the spatial resolution to 9×9 and outputs 128 channels. The tensor is reshaped from `[B, 128, 9, 9]` to `[B, 16, 8, 9, 9]`, permuted to `[B, 16, 9, 9, 8]`, and flattened to `[B, 1296, 8]`. These 1296 primary capsules each have 8 dimensions.
- **Digit Capsules:** As in AlexCapsNet, a digit capsule layer with dynamic routing maps the 1296 primary capsules to 10 digit capsules of dimension 16. The lengths of these output vectors `[B, 10]` are used as class scores.

This architecture blends convolutional layers, residual connections, and capsule networks. Each transformation step aligns with the implemented model.

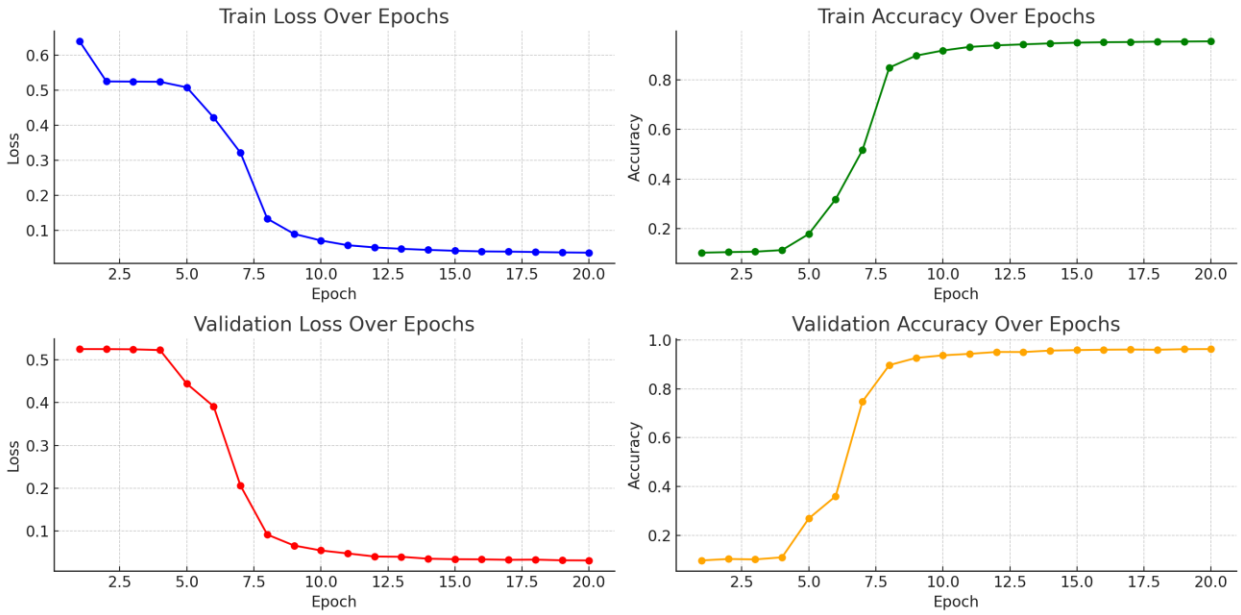
Training Configuration

- **Data Preprocessing:**
  - *ResCapsNet:* MNIST images were resized to 78×78 and normalized to [0,1] (no mean subtraction).
  - *AlexCapsNet:* Inputs remained 28×28 and were standardized to zero mean and unit variance. Both models used random ±10° rotations for data augmentation during training and employed PyTorch `DataLoader` with batch size 32 and training set shuffling.
- **Optimizers and Loss:**
  Both models used the Adam optimizer with a learning rate of 0.001. Capsule margin loss was used instead of cross-entropy. This setup encourages the correct digit capsule to reach a length near 1 and all others near 0.
- **Mixed Precision and Distributed Training:**
  Both were trained using multi-GPU DDP with gradient accumulation (4 steps) and mixed-precision training using `torch.cuda.amp.GradScaler()` to stabilize loss scaling during backpropagation and also were trained on a single GPU, using standard precision but shared the same optimizer and loss function. A step learning rate decay (factor 0.7 every 2 epochs) was applied in both setups.
- **Hyperparameters:**
  - *Distributed Computing:* batch size 32 per GPU, learning rate ≈0.001, 20–50 epochs, StepLR with γ=0.7
  - *Single Gpu:* batch size 32, learning rate 0.001, 20–50 epochs
    Both used 3 routing iterations for digit capsules and no dropout.

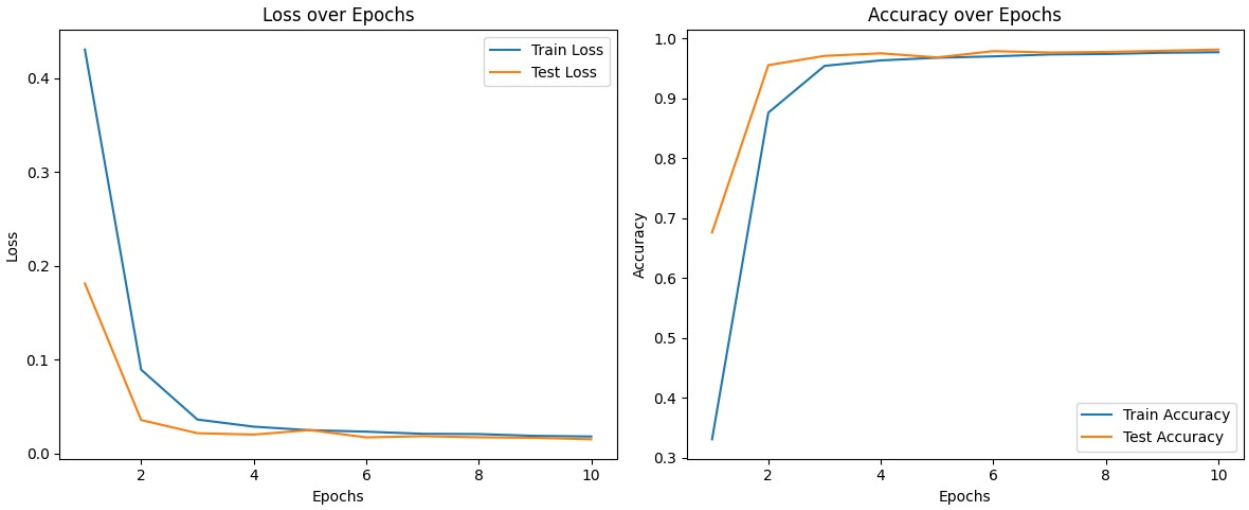
Results

Both models quickly achieved high accuracy on MNIST.

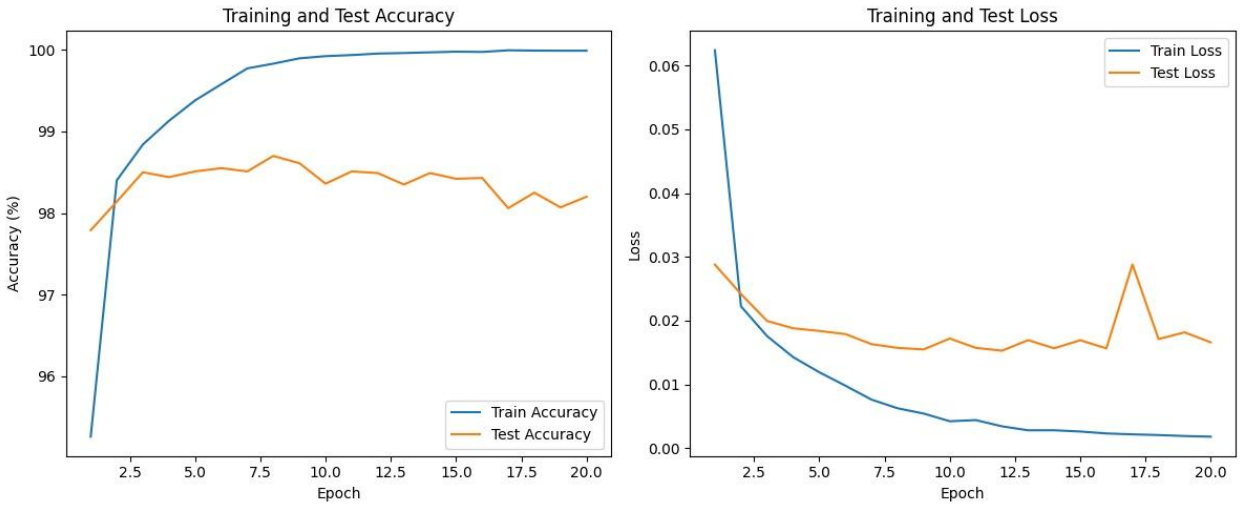
- *AlexCapsNet:* reached 96.28 test accuracy on Multi GPU setup at 20 epochs.



- *AlexCapsNet:* reached 98.06 test accuracy on Single GPU setup at 10 epochs.



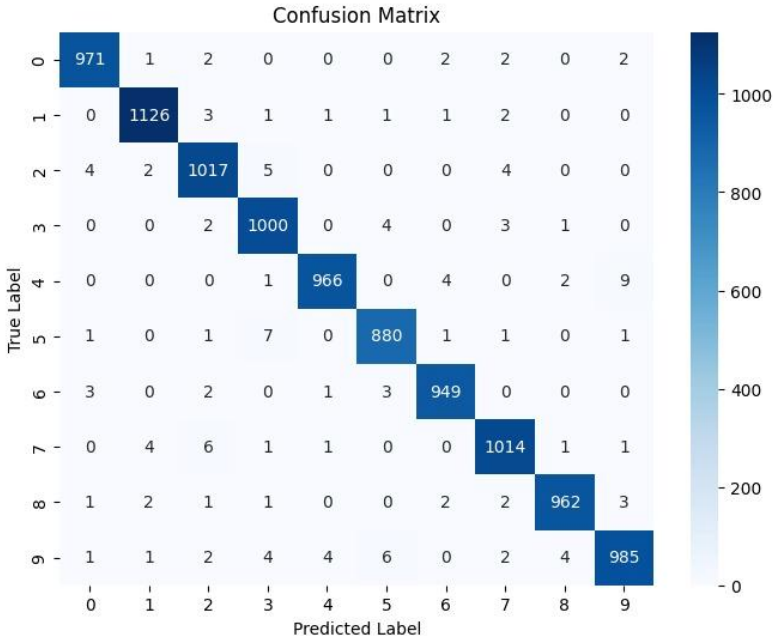
- *ResCapsNet:* reached 97.86 test accuracy on Multi GPU setup at 10 epochs.
- *ResCapsNet:* reached 98.70% test accuracy on Single GPU setup at 20 epochs



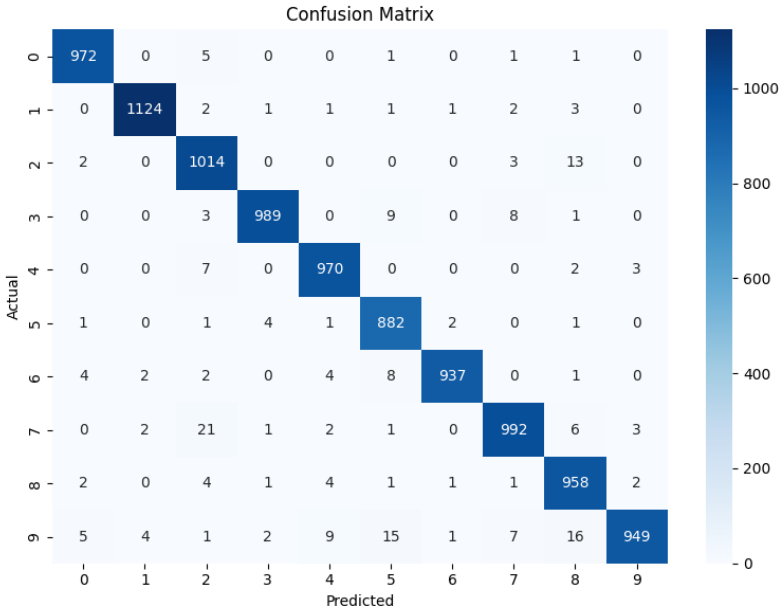
Training and validation loss dropped smoothly to near zero for both models. Even though ResCapsNet achieved highest overall accuracy, AlexCapsNet showed faster convergence, achieving 90%+ accuracy 2–3 epochs earlier.

Confusion matrices confirmed that both models classified nearly all digits correctly.

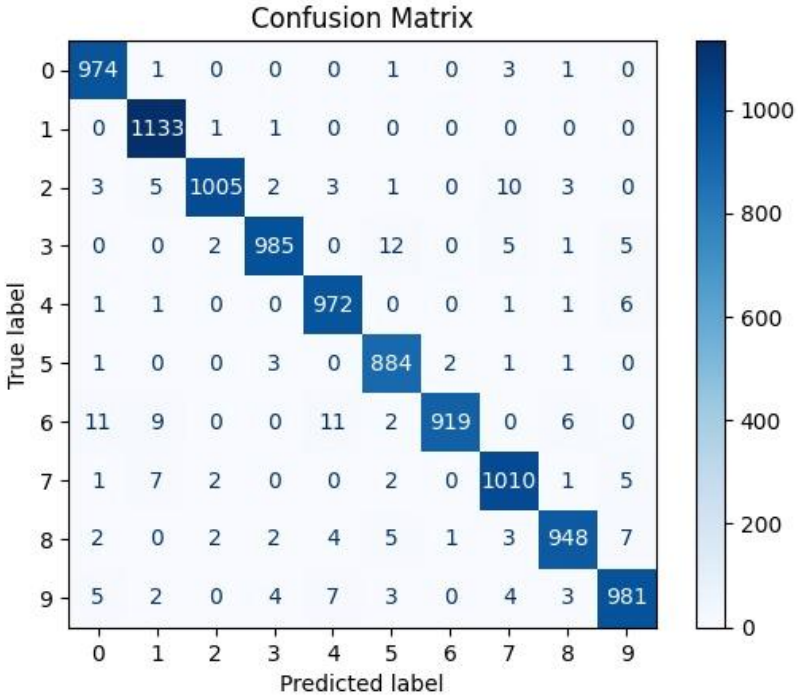
- ResCapsNet Single GPU



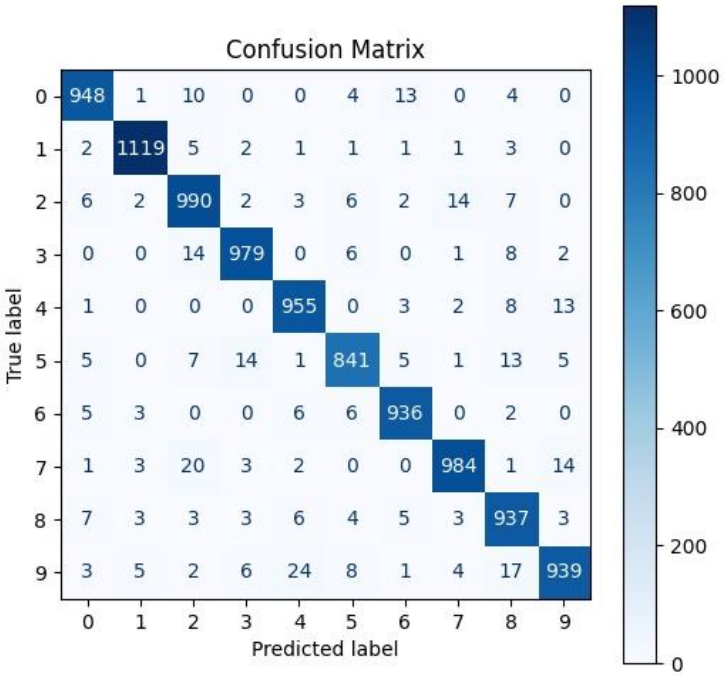
- ResCapsNet Multi GPU



- AlexCapsNet Single GPU

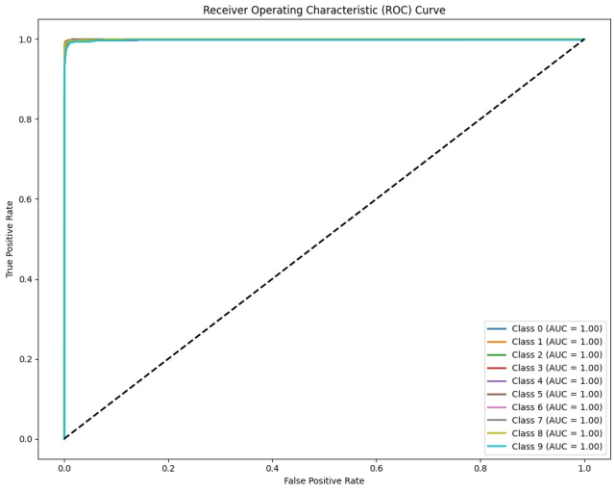


- AlexCapsNet Multi GPU

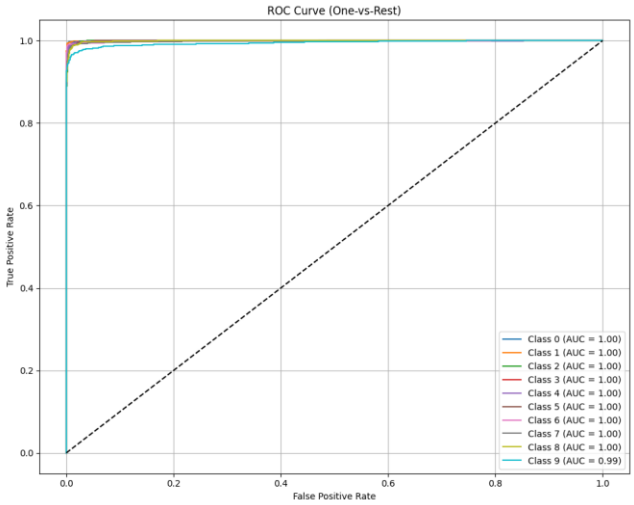


ROC curves (one-vs-rest) were nearly perfect for both models:

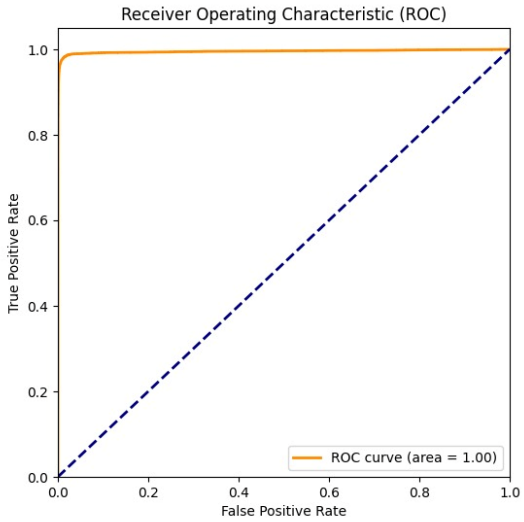
- ResCapsNet Single GPU



- ResCapsNet Multi GPU



- AlexCapsNet Multi GPU



Comparative Analysis

- **Accuracy and Convergence:**

ResCapsNet consistently achieved slightly higher accuracy in the final results, particularly on the single-GPU setup, reaching up to 98.70%. AlexCapsNet, while slightly lower in peak accuracy, demonstrated faster convergence in earlier epochs due to its deeper convolutional backbone. ResCapsNet was more compact and converged slightly slower but remained highly accurate.

- **Model Complexity:**

AlexCapsNet has tens of millions of parameters and trains ~2× slower per epoch than ResCapsNet (which has only a few million). However, its larger model learned more quickly per epoch, especially during early training. ResCapsNet is better suited for low-resource environments and achieved better final accuracy in some cases.

- **Distributed Training Effect:**

AlexCapsNet benefited from parallelism (multi-GPU DDP), reducing wall-clock training time significantly. ResCapsNet trained effectively on both multi-GPU and single-GPU setups, with longer per-epoch time on single-GPU. Final accuracies were comparable or better in some ResCapsNet runs, but AlexCapsNet showed faster convergence in early epochs, particularly on single-GPU.

- **Routing and Capsules:**

Both used identical routing mechanisms and margin loss. Capsule layers preserved digit orientation in both architectures. No instability was observed with 3 routing iterations.

Conclusion

We compared both AlexCapsnet and ResCapsNet on MNIST dataset. The two Architectures have showed similar but complimentary approaches, the prior prioritizes richer features but at a higher computational cost, meanwhile the latter uses skip connections and achieves a strong final accuracy. Overall, both hybrid models showed that combining an effective feature extractor with CapsNet for classification is an effective and efficient strategy for multiclass classifications.

References

- **M. Bao, N. Jin and M. Xu, "AlexCapsNet: An Integrated Architecture for Image Classification With Background Noise," in IEEE Access, vol. 13, pp. 37690-37702, 2025, doi: 10.1109/ACCESS.2025.3544661.**
- **T. Tian, X. Liu and L. Wang, "Remote Sensing Scene Classification Based on Res-Capsnet," IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, Yokohama, Japan, 2019, pp. 525-528, doi: 10.1109/IGARSS.2019.8898656.**