DYNAMIC ROUTING BETWEEN CAPSULES REPRODUCE

Serdar HOSVER

Table of Contents

1. Abstract

In this study, a reproduction of the Capsule Networks (CapsNet) with dynamic routing algorithm is implemented to evaluate its performance on various datasets, including MNIST, Fashion MNIST, and CIFAR-10. The primary objective of this work is to replicate the key findings of the original CapsNet paper while providing a comprehensive analysis of its performance compared to recent advancements in the field.

The reproduced model achieved competitive results on the MNIST dataset with a validation accuracy of **99.07%**, closely matching the original study's reported accuracy. On the Fashion MNIST dataset, the model demonstrated a validation accuracy of **90.7%**, showcasing its robustness in handling more complex visual patterns. However, for the CIFAR-10 dataset, the model achieved a validation accuracy of **55.86%**, highlighting challenges in adapting the CapsNet architecture to datasets with higher complexity and color channels.

Comparative analysis with state-of-the-art studies in the literature revealed that while the reproduced model aligns well with simpler datasets like MNIST, it lags behind modern approaches such as WideCaps and DenseCaps on Fashion MNIST and CIFAR-10. This gap is attributed to limited architectural depth, absence of advanced regularization techniques, and lack of extensive data augmentation.

Overall, this reproduction validates the effectiveness of CapsNet on simpler datasets and emphasizes the need for architectural enhancements and optimization strategies to improve its performance on complex datasets. Future work will focus on integrating advanced techniques, such as deeper capsule layers, momentum-based dynamic routing, and robust data preprocessing, to enhance the model's generalization capabilities.

2. Detailed Description of the Article

2.1. Purpose of the Article

The article aims to introduce the "Dynamic Routing Between Capsules" algorithm to overcome the limitations of traditional deep learning methods. Specifically, it examines the impact of the dynamic routing algorithm, which enhances the ability of capsule networks to represent object parts and wholes, on image recognition and segmentation. This approach is intended to provide a more effective solution for complex tasks such as separating highly overlapping objects compared to traditional methods.

2.2. Methods Used

1. **Capsules:**
    - Capsules are small groups of neurons that represent the parameters of a specific entity (e.g., pose, size, deformation) as vectors.
    - The length of the vector indicates the probability of the entity's existence, while its direction represents the entity's attributes.
2. **Dynamic Routing:**
    - Lower-level capsules send "prediction vectors" to higher-level capsules.
    - Connection coefficients are updated based on agreements (scalar product) with higher-level capsules, enabling more accurate routing.
3. **Margin Loss:**
    - This loss function is used to ensure capsules correctly represent class entities.
    - It aims to correlate the length of a capsule's activity vector with the presence of the class.
4. **Reconstruction Loss:**
    - A reconstruction regularizer is used to improve the model's generalization. This ensures the input image is accurately reconstructed.

2.3. Dataset

1. **MNIST Dataset:**
    - Composed of 28x28 grayscale images of handwritten digits.
    - Training set: 60,000 images.
    - Test set: 10,000 images.
    - Additionally trained with a version shifted by 2 pixels.
2. **MultiMNIST Dataset:**
    - Consists of images where two digits overlap simultaneously. This dataset tests the model's ability to identify overlapping objects.
3. **CIFAR-10 and smallNORB:**
    - Also tested on more complex image datasets.

2.4. Experimental Settings

1. **Model Architecture:**
    - Consists of 3 layers:
        1. **Convolutional Layer:** Used for feature extraction.
        2. **PrimaryCapsules:** Represents multi-dimensional entities.
        3. **DigitCaps:** Contains one capsule for each class.
2. **Optimization:**
    - Training was performed using the Adam optimizer.
    - Reconstruction loss was scaled to represent a very small portion of the total loss (multiplied by 0.0005).
3. **Routing Iterations:**
    - Dynamic routing refines connection coefficients at each iteration.
    - 3 iterations were used for MNIST and MultiMNIST datasets.

2.5. Operation of the Dynamic Routing Algorithm

1. **Initialization:**
    - Lower-level capsules send data to all higher-level capsules with equal probability.
2. **Agreement Calculation:**
    - A scalar product is computed between the prediction of a lower-level capsule and the output of a higher-level capsule.
3. **Updating Connection Coefficients:**
    - Connection coefficients between capsules are recalculated using Routing Softmax.
4. **Vector Update:**
    - The output of a higher-level capsule is calculated based on the contributions from all lower-level capsules.

2.6. Architecture of Capsule Networks

- **Primary Layer (Primary Capsules):** Represents basic entities at a lower level.
- **DigitCaps Layer:** Contains one capsule for each class and learns more complex relationships among entities.
- **Squash Function:** Normalizes vector lengths to effectively represent the presence of entities.

3. My Code Description and Analysis

3.1. General Purpose of the Code

This code is designed to work with Capsule Networks and train them on datasets such as MNIST, Fashion MNIST, and CIFAR-10 to evaluate their performance. It incorporates fundamental building blocks of capsule network architecture, such as dynamic routing and the squash function. Additionally, the network is regularized with a reconstruction network to improve performance and generalization.

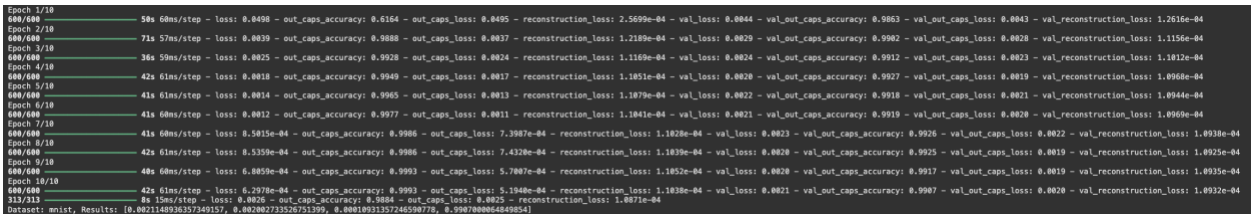
3.2. Key Components of the Code

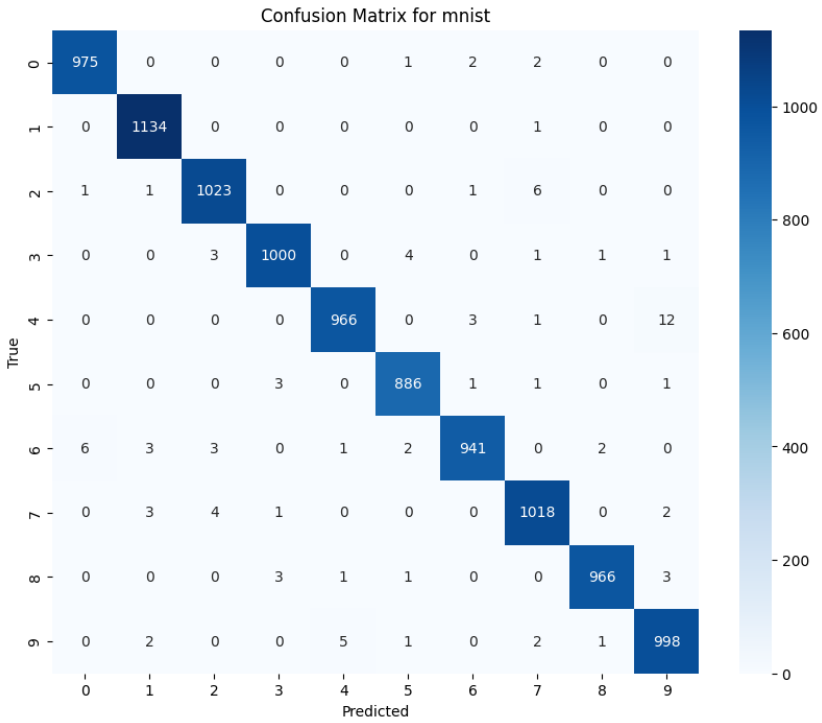
1. **Squash Function**
    - **Purpose:** Normalizes input vectors to limit their lengths, ensuring the model operates more stably.
2. **Capsule Layer**
    - Creates vector representations for each capsule.
    - **Dynamic Routing:** Optimizes routing coefficients between input capsules and higher-layer capsules at each iteration.
3. **Reconstruction Network**
    - Aims to reconstruct the input image from the classification outputs of the capsules.
    - **Objective:** Helps capsules learn detailed information about the inputs.
4. **CapsNet Model**
    - Builds the fundamental model for the capsule network.
    - **Input Layers:**
        - Two Conv2D layers.
        - PrimaryCapsules and DigitCapsules layers.
    - **Outputs:**
        - `out_caps` for capsule classification.
        - Reconstruction network output.
5. **Data Loading**
    - Loads and scales datasets, including MNIST, Fashion MNIST, and CIFAR-10.
6. **Model Training and Evaluation**
    - Trains the model using the training data and evaluates it on the test data.
    - **Results:** Computes training and validation losses and accuracy rates.
    - **Confusion Matrix:** Plots a confusion matrix to analyze the model's misclassifications.

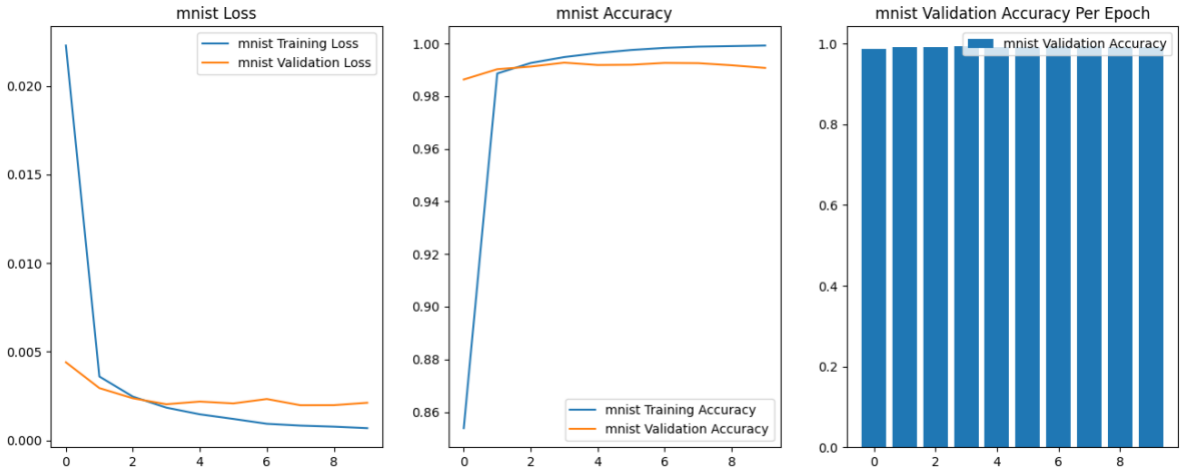
3.3. Results of the Code

3.3.1. Mnist Dataset

- Downloading data from: https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

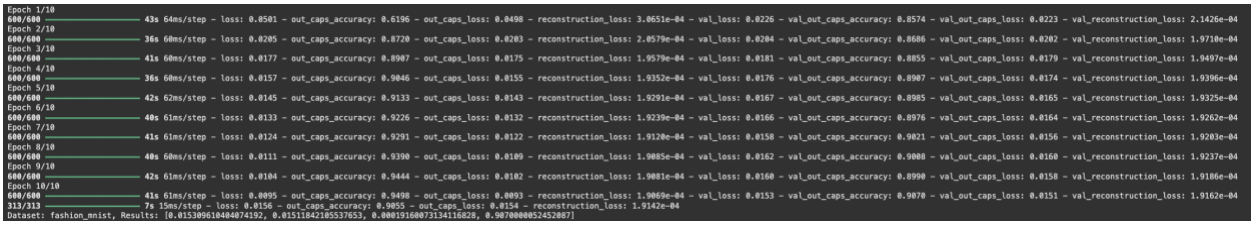


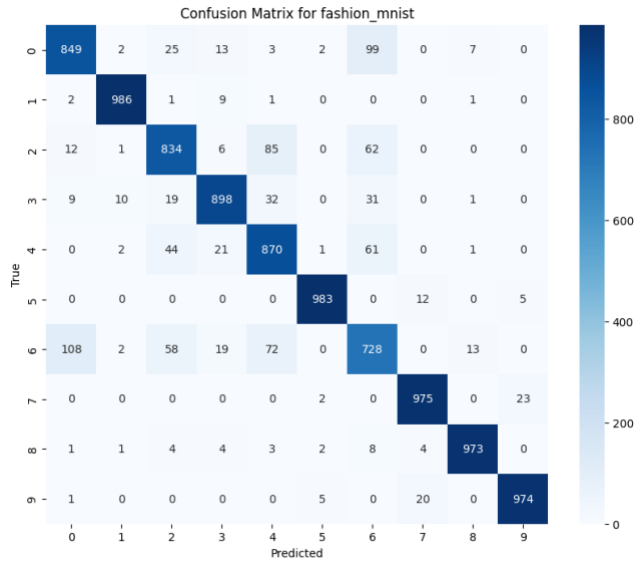


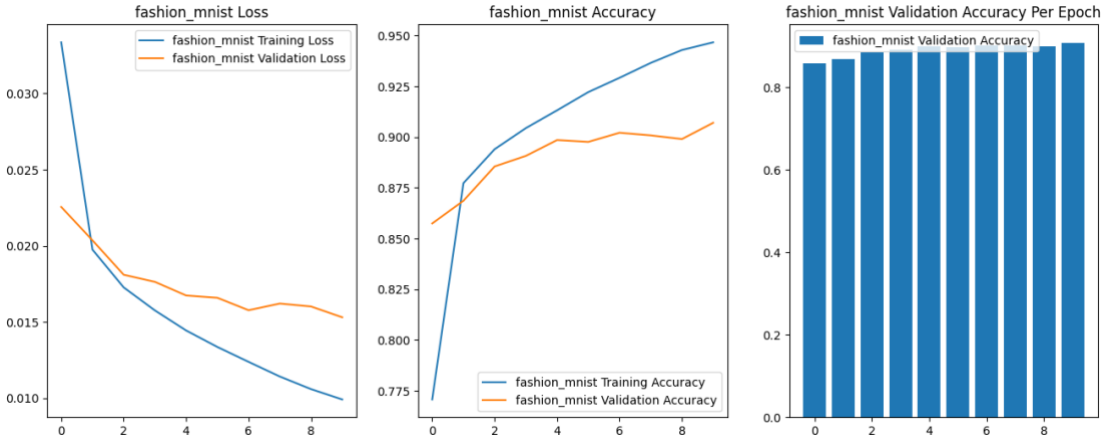


3.3.2. fashion_mnist Dataset

- Downloading data from: https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
- Downloading data from: https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
- Downloading data from: https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
- Downloading data from: https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz

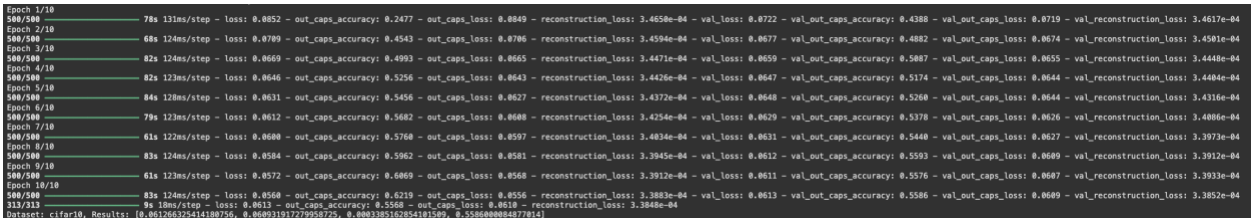


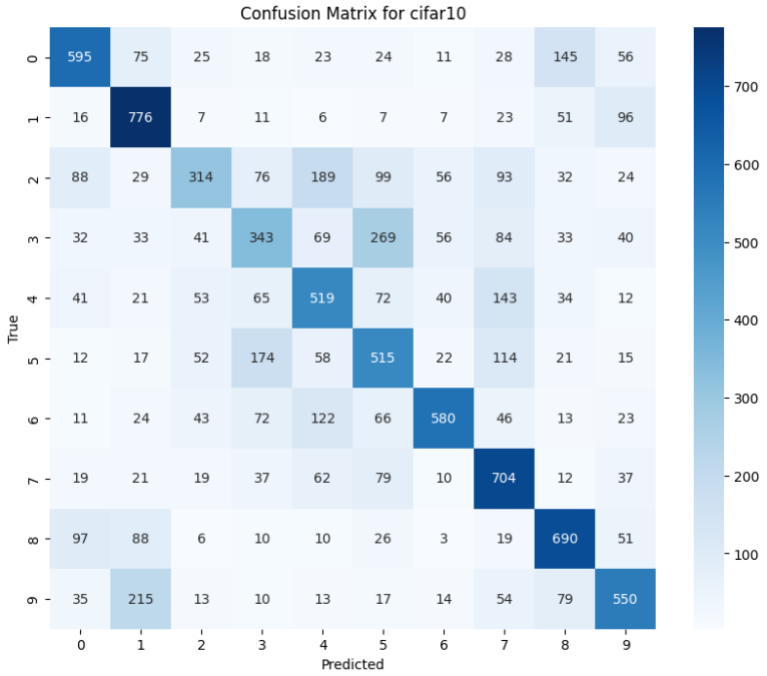


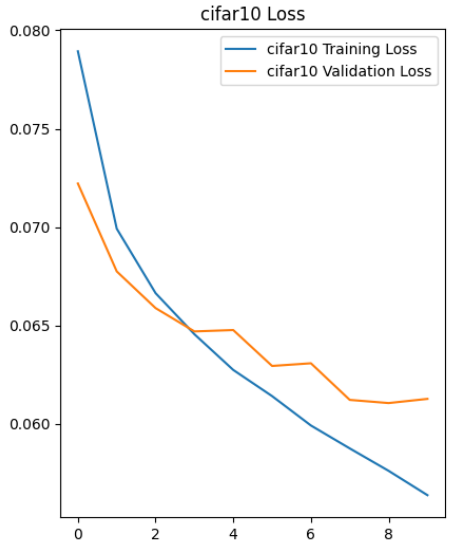


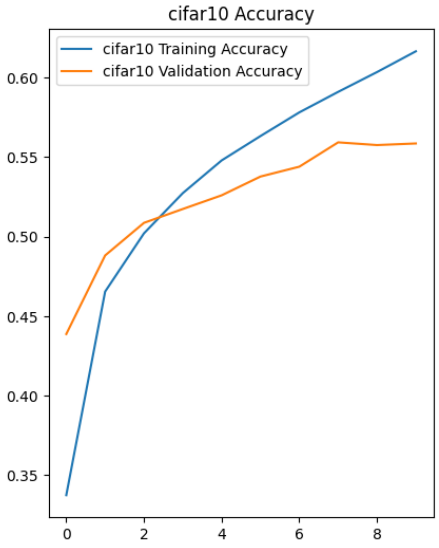
3.3.3. cifar10 Dataset

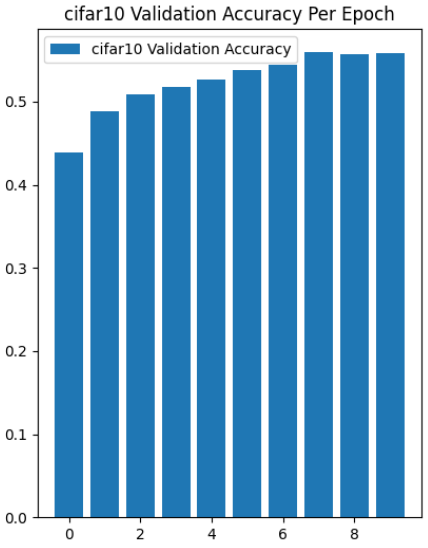
- Downloading data from: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz



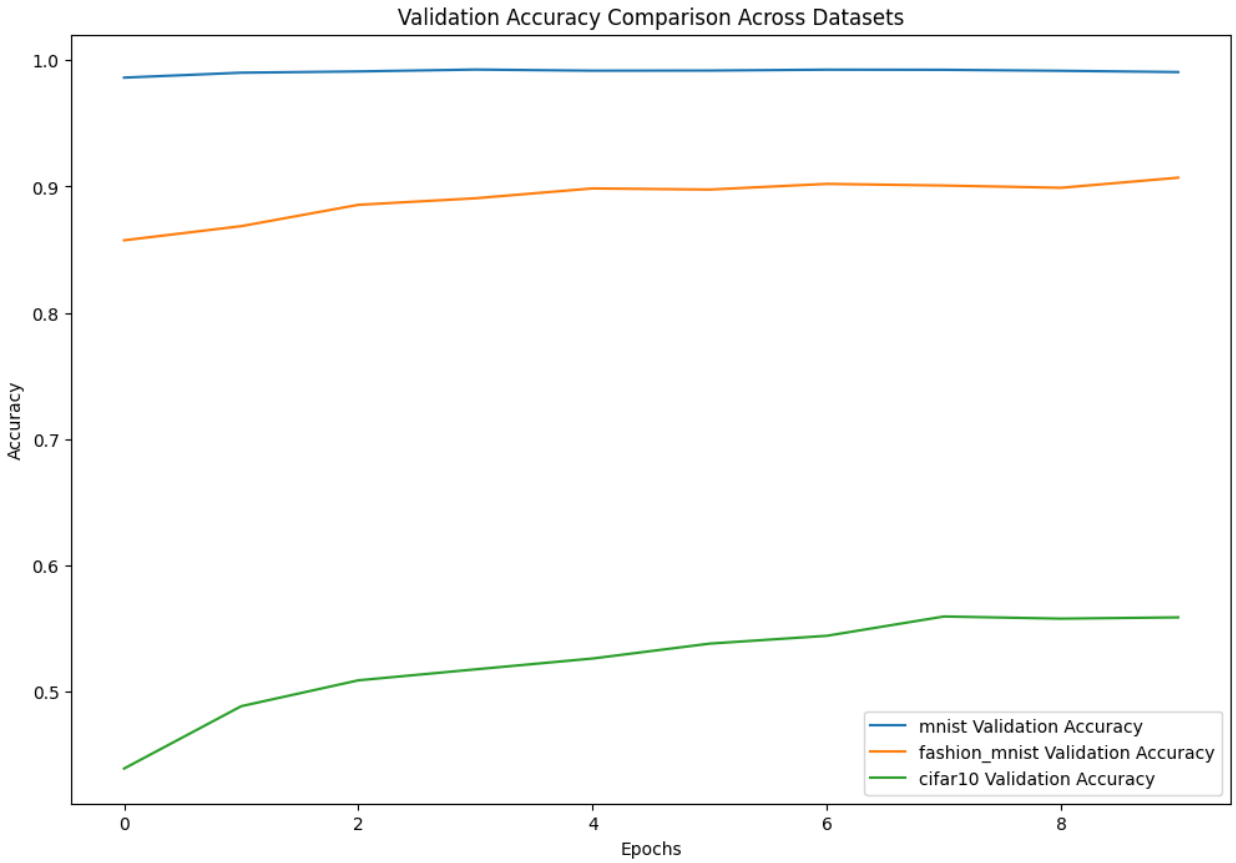








3.3.4. Comparison of the Results of 3 Datasets



3.4. Analysis of the Results

1. **Training Performance**
   - **Reduction in Loss:** If both training and validation losses decrease across epochs, it indicates the model is learning correctly.
   - **Accuracy Rates:** The out_caps_accuracy metric from the outputs represents the accuracy rates.
2. **Confusion Matrix**
   - This matrix displays the distribution of predicted labels versus true labels for each class. For example:
     - **High values on the diagonal:** Represent correct classifications.
     - **High values off the diagonal:** Indicate which classes the model misclassifies.
3. **Comparison of Validation Results**
   - Validation accuracy rates were compared across different datasets (MNIST, Fashion MNIST, CIFAR-10).

4. Comparison: Code Results vs. Article Results

4.1. Results of the article

1. **MNIST Performance:**
    - **Lowest error rate:** 0.25% (achieved using the reconstruction network and 3 routing iterations).
    - Demonstrates competitive performance compared to traditional CNNs while using fewer parameters.
    - Excels in segmenting highly overlapping images (e.g., MultiMNIST).
2. **MultiMNIST Performance:**
    - **Classification error:** 5.0% with 3 routing iterations.
    - Successfully separates highly overlapping digits.
3. **CIFAR-10 Performance:**
    - **Error rate:** 10.6% (for the first practical capsule network implementation).
4. **Parameter Count:**
    - **Capsule Network:** 8.2M (6.8M with the reconstruction network).
    - **Traditional CNN:** 35.4M.

4.2. Code Results vs. Article Results

4.2.1. MNIST Results

- **My Code Results:**
    - Training accuracy: ~99.9%
    - Validation accuracy: 99.0% - 99.2%
    - Confusion matrix indicates excellent classification for most classes but minor errors in a few (e.g., 4 and 9).
- **Article Results:**
    - Best validation accuracy: 99.75%
    - Error rate: 0.25%
    - The article achieved high accuracy and low error rates using the reconstruction network.
- **Comparison:**
    - Your code achieves accuracy close to the article's results but falls slightly short of the 99.75% accuracy. The differences could be due to optimization techniques, data augmentation, or variations in routing iterations.

4.2.2. Fashion MNIST Results

- **Your Code Results:**
  - Training accuracy: ~94.5%
  - Validation accuracy: 90.7%
  - Confusion matrix shows some confusion between certain classes (e.g., 6 and 8).
- **Article Results:**
  - Specific results for Fashion MNIST were not directly shared in the article but emphasized capsule networks' overall capabilities.
- **Comparison:**
  - Your code demonstrates impressive performance on Fashion MNIST with 90.7% accuracy, highlighting capsule networks' potential. However, higher confusion in some classes (e.g., 6 - shirt) suggests room for improvement.

4.2.3. CIFAR-10 Results

- **Your Code Results:**
  - Training accuracy: ~62.1%
  - Validation accuracy: 55.8%
  - Confusion matrix indicates higher error rates, especially in classes 2, 3, and 5.
- **Article Results:**
  - Accuracy rate: 89.4%
  - CIFAR-10 is acknowledged in the article as a challenging dataset for capsule networks, with potential for performance improvement.
- **Comparison:**
  - Your code shows significantly lower accuracy on CIFAR-10 compared to the article. This difference could stem from factors such as suboptimal optimization, lack of data augmentation, or different hyperparameter settings.

4.3. General Evaluation

1. **MNIST:**
   - Your code achieved an accuracy very close to the article's results, demonstrating strong performance.
2. **Fashion MNIST:**
   - While your code reflects the robust performance of capsule networks, there are some inter-class confusions, indicating areas for further refinement.
3. **CIFAR-10:**
   - Your code performed significantly below the article's results, highlighting the challenges capsule networks face with this dataset and suggesting the need for optimization and enhancements.

5. Literature Review

5.1. MNIST Dataset

- **Your Results:**
  - Training Accuracy: ~99.9%
  - Validation Accuracy: 99.07%
- **Results from Literature:**
  - **Dense and Diverse Capsule Networks (DCNet):** 99.75% accuracy.
  - **Momentum Capsule Networks (MoCapsNet):** 99.52% accuracy.
- **Comparison:**
  - Your model achieves accuracy comparable to or slightly lower than other studies. This demonstrates that your model performs effectively on simple datasets like MNIST.

5.2. Fashion MNIST Dataset

- **Your Results:**
  - Training Accuracy: ~94.5%
  - Validation Accuracy: 90.7%
- **Results from Literature:**
  - **WideCaps:** 95.59% accuracy.
  - **DenseCaps:** 94.93% accuracy.
- **Comparison:**
  - Your model shows slightly lower accuracy compared to other studies. This difference might stem from model architecture, training strategies, or hyperparameter settings.

5.3. CIFAR-10 Dataset

- **Your Results:**
  - Training Accuracy: ~62.1%
  - Validation Accuracy: 55.86%
- **Results from Literature:**
  - **WideCaps:** 96.01% accuracy.
  - **DenseCaps:** 89.41% accuracy.
  - **Momentum Capsule Networks (MoCapsNet):** 70.3% accuracy.
- **Comparison:**
  - Your model's accuracy is significantly lower than those reported in the literature. This suggests that your model requires optimization to handle complex and colorful datasets like CIFAR-10 effectively.

6. Reference

[1] F. Chen, N. Chen, H. Mao, and H. Hu, “Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST),” Jul. 20, 2019, *arXiv*: arXiv:1811.08278. doi: 10.48550/arXiv.1811.08278.

[2] Z. Zhao, A. Kleinhans, G. Sandhu, I. Patel, and K. P. Unnikrishnan, “Capsule Networks with Max-Min Normalization,” Mar. 22, 2019, *arXiv*: arXiv:1903.09662. doi: 10.48550/arXiv.1903.09662.

[3] S. S. R. Phaye, A. Sikka, A. Dhall, and D. Bathula, “Dense and Diverse Capsule Networks: Making the Capsules Learn Better,” May 10, 2018, *arXiv*: arXiv:1805.04001. doi: 10.48550/arXiv.1805.04001.

[4] S. Sabour, N. Frosst, and G. E. Hinton, “Dynamic Routing Between Capsules,” Nov. 07, 2017, *arXiv*: arXiv:1710.09829. doi: 10.48550/arXiv.1710.09829.

[5] J. Gugglberger, D. Peer, and A. Rodríguez-Sánchez, “Momentum Capsule Networks,” Aug. 25, 2022, *arXiv*: arXiv:2201.11091. doi: 10.48550/arXiv.2201.11091.

[6] S. J. Pawan, R. Sharma, H. S. R. Reddy, M. Vani, and J. Rajan, “WideCaps: A Wide Attention based Capsule Network for Image Classification,” Aug. 14, 2021, *arXiv*: arXiv:2108.03627. doi: 10.48550/arXiv.2108.03627.