Report for MIDAS Internship Task 2

Animesh Gupta

Problem Statement

Train an image classification model for classifying digits and alphabets under 3 given subtasks.

1. For the first subtask, given a dataset containing digits and alphabets (capital and small), a total of 62 classes (Fig. 1), train a neural network from scratch with random weight initialization without including any external data or pretrained weights during the training process.
2. For the second subtask, select images from the given dataset containing only digits (0-9) as labels. Compare the performance of 2 neural networks (mentioned below in detail) on the dataset containing only digits (0-9) as labels.
    1. Train on the MNIST train set a network from scratch with random weight initialization and evaluate on the MNIST test set.
    2. Train on the dataset containing only digits (0-9) as labels then save the best network. Load the best-saved network weights to then train on the MNIST train set and evaluate on the MNIST test set.
3. For the third subtask, given a new dataset containing only digits as labels but each label contains images having all the digits from 0-9. For eg, a folder with the label 0 contains images with all the digits from 0-9. (Fig. 12) Compare the performance of 2 neural networks (mentioned below in detail) on the given new dataset.
    1. Train using the given new dataset from scratch with random weight initialization and evaluate on MNIST test set.

2. Load the best-saved network weights from subtask 2 to then train on the MNIST train set and evaluate on the MNIST test.

A well-documented source code is available on https://github.com/animesh-007/midas_internship_task2

First subtask

Implementation details

Given a dataset with images of digits and alphabets (capital and small).

Fig.1 An overview of the dataset.

The following transformations were applied to the dataset before using it as an input to train the network.

1. **Center Crop:** In the given dataset, the images are of size H=900 and W=1200, and as the images contain objects in the center, object

detail is not lost. Centercrop (900,900) was applied to scale the image uniformly thus maintaining the aspect ratio.

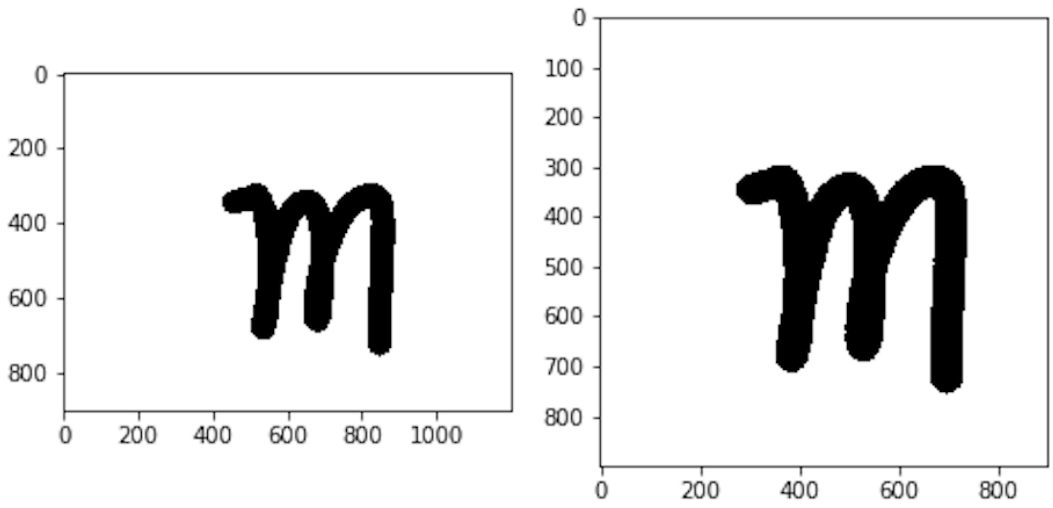

Fig.2 A comparison between the original image and centercrop image.

2. **Image Inversion:** To maintain the same data distribution as MNIST, Invert was applied to invert the colours of the images.

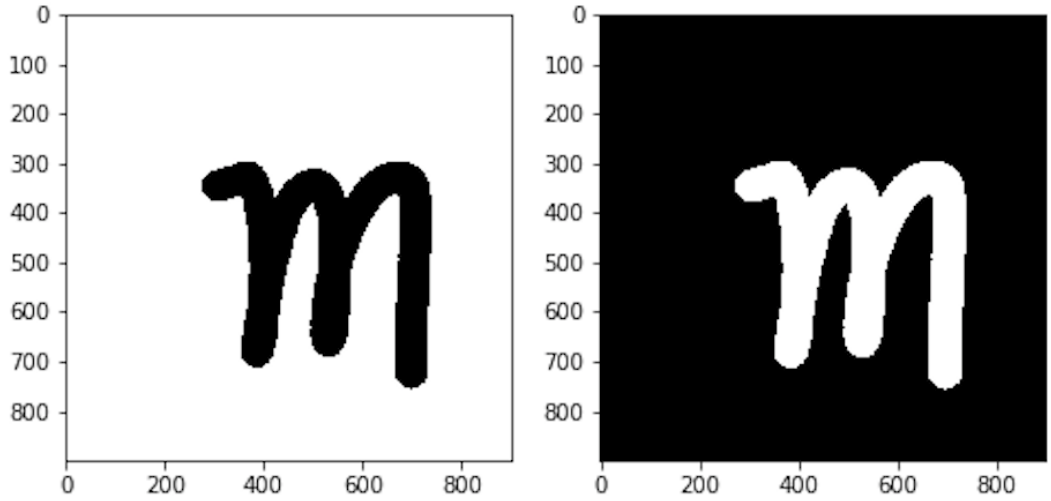

Fig. 3 A comparison between the original image and inversion image.

3. **Grayscale Conversion:** To maintain the same data distribution as MNIST, Grayscale was applied to convert 3-channel images to 1-

channel images.

4. **Resize Images:** To train the model efficiently. The images were resized to H=28 and W=28.

5. **Normalize Images:** The training images are then converted to tensors and normalized by the mean (0.1307) and standard deviation (0.3081) to help the network get data within a range and reduce the skewness which helps learn faster and better.

Network and Training Details

- We divide the given dataset into training and validation set in 80:20 split to cross-validate the network.
- The network consists of 2 convolutions layers with input channels [1,32] and output channels [32,*64*] respectively. Each followed by a relu activation function. The maxpool pooling layer was applied after the output of the second convolution layer. Dropout was applied with a value of 0.25. The output is then flattened and a fully connected layer was applied with input features 9216 and output features 128. Dropout was applied with a value of 0.5. The fully connected layer with input features 128 and output features 62 as the final layer of the model.
- We trained the network for 30 epochs with a learning rate of 1 and optimizer adadelta.
- We choose this network as it's faster+efficent for datasets like MNIST because training with deeper networks will saturate the performance.

Motivation for using Scheduler: Learning Rate (LR) Scheduler helps to adjust the LR during training. Models often benefit from this technique once learning stagnates, and helps to get better results.

- We used CosineAnnealingLR Scheduler: Min LR 0.8, Max LR 1, and warm-up 5 epochs.
- CosineAnnealingLR Scheduler increased the performance of the network as reported in Table1.

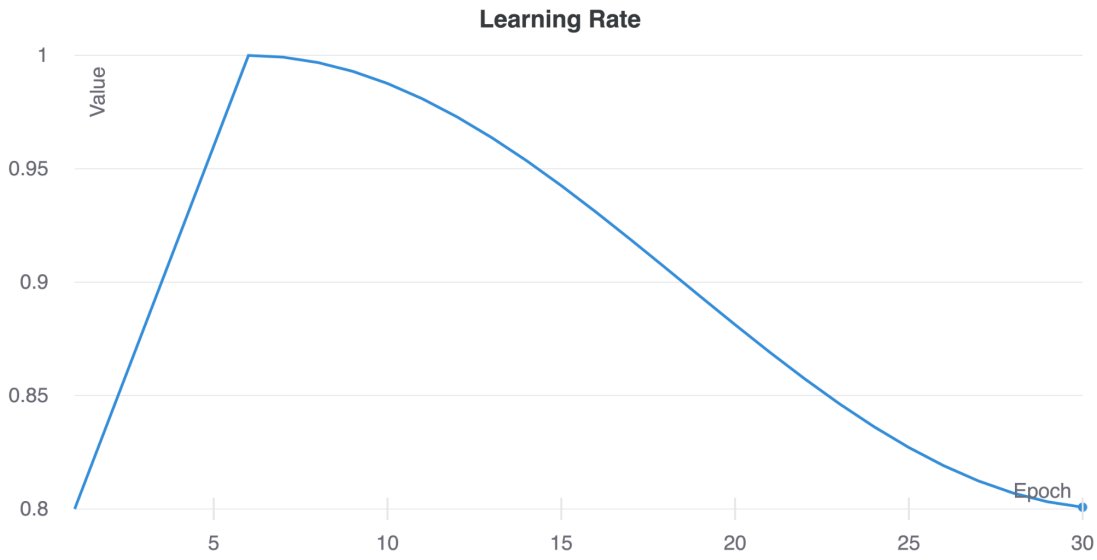

Fig. 4 Learning Rate using CosineAnnealingLR

- The trained network is cross-validated on the validation set (20% of the full dataset) to select the best model and hyperparameters.

Results

Method	Epochs	Accuracy
CNN without a Scheduler	30	67.74
CNN with CosineAnnealingLR Scheduler	30	68.75

Table 1 Quantitive result comparing the scheduler

As seen in Table 1, the performance is increased by 1.49% with the use of cosineannealing LR scheduler.

Subtask 2

Implementation details

Make a subset of the above dataset containing only digits labels.

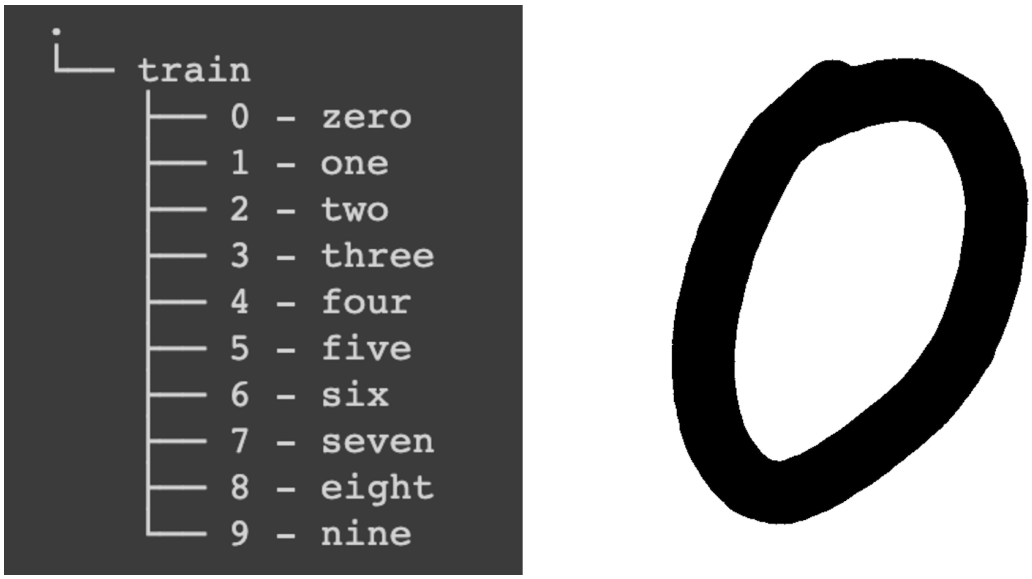

Fig. 5 Overview of the directory and image inside the dataset.

The following transformations were applied to the dataset before using it as an input to train the network.

1. **Center Crop:** In the given dataset, the images are of size H=900 and W=1200, and most of the images contain objects in the center, Centercrop (900,900) was applied to scale the image uniformly thus maintaining the aspect ratio.

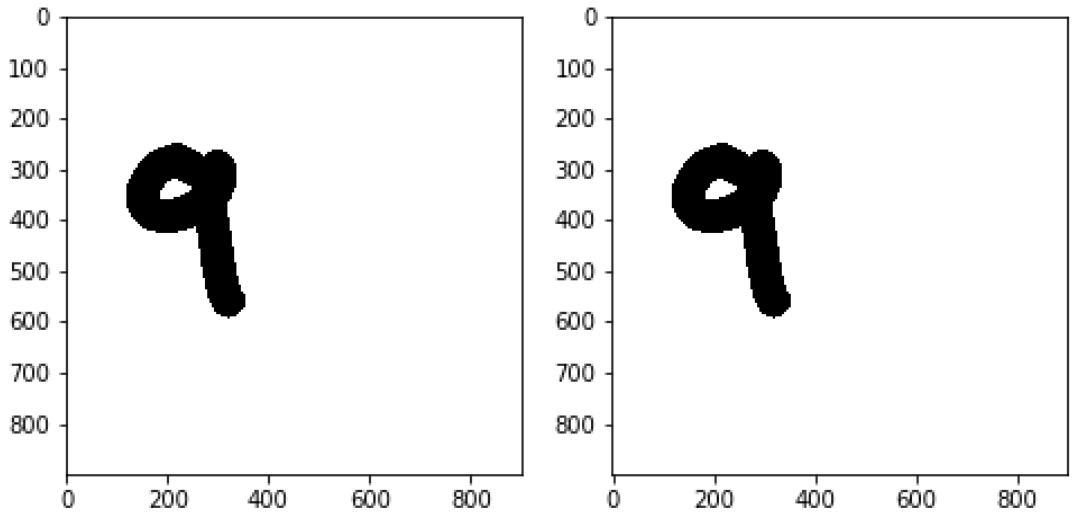

Fig.6 A comparison between the original image and centercrop image.

2. **Image Inversion:** To maintain the same data distribution as MNIST, Invert was applied to invert the colours of the images.

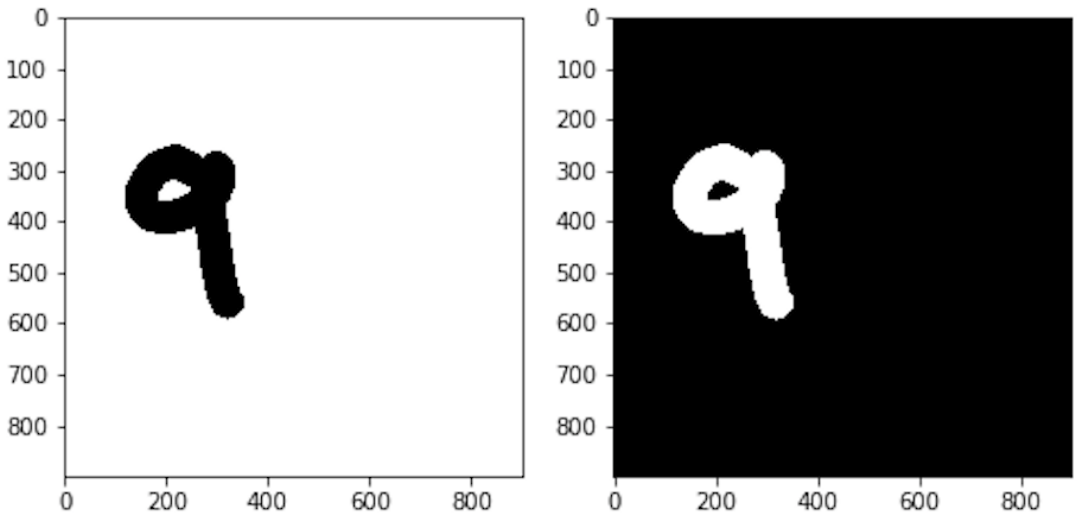

Fig. 7 A comparison between the original image and inversion image.

3. **Grayscale Conversion:** To maintain the same data distribution as MNIST, Grayscale was applied to convert 3-channel images to 1-channel images.

4. **Resize Images:** To train the model efficiently. The images were resized to H=28 and W=28.

5. **Normalize Images:** The training images are then converted to tensors and normalized by the mean (0.1307) and standard deviation (0.3081) to help the network get data within a range and reduce the skewness which helps learn faster and better.

Network and Training details

- The network consists of 2 convolutions layers with input channels [1,32] and output channels [32,*6*4] respectively. Each followed by a relu activation function. The maxpool pooling layer was applied after the output of the second convolution layer. Dropout was applied with a value of 0.25. The output is then flattened and a fully connected layer was applied with input features 9216 and output features 128. Dropout was applied with a value of 0.5. The fully connected layer with input features 128 and output features 10 as the final layer of the model.
- The network is trained for 30 epochs with a learning rate of 1 and optimizer adadelta.

- We used CosineAnnealingLR Scheduler: Min LR 0.8, Max LR 1, and warm-up 5 epochs.
- CosineAnnealingLR Scheduler increased the performance of the network as reported in Table2.
- The trained network is tested on the MNIST test set to select the best model and hyperparameters

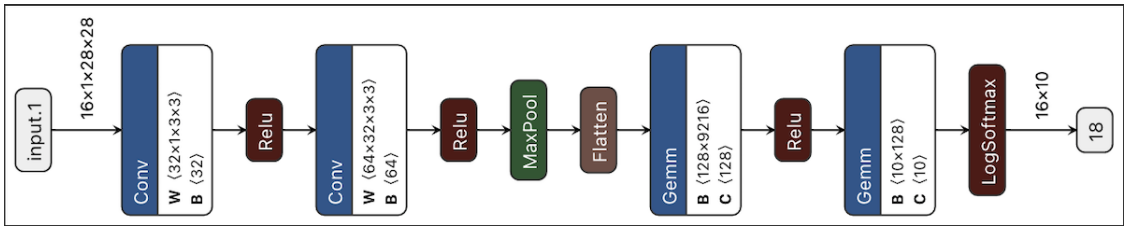

Fig. 8: An overview of the model architecture.

Training Strategy

- Making a subset of the above dataset containing only images of digits.
- Training the model on the processed dataset for 30 epochs with a learning rate of 1.

Results

Training on the MNIST dataset with random weights

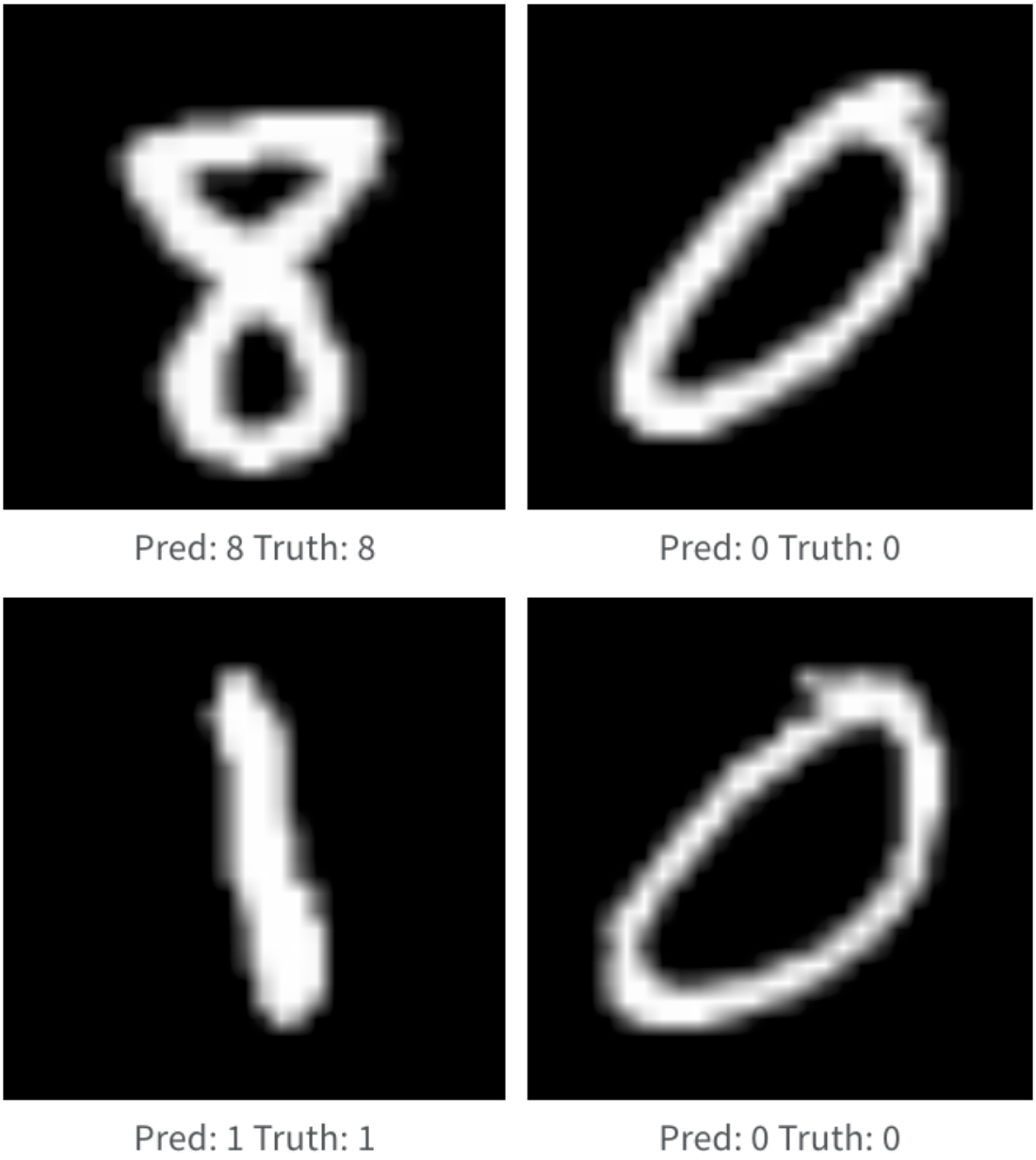

Fig. 9 Prediction and Ground truth by training the model on the subset of the dataset.

Train on the MNIST dataset with pretrained weights

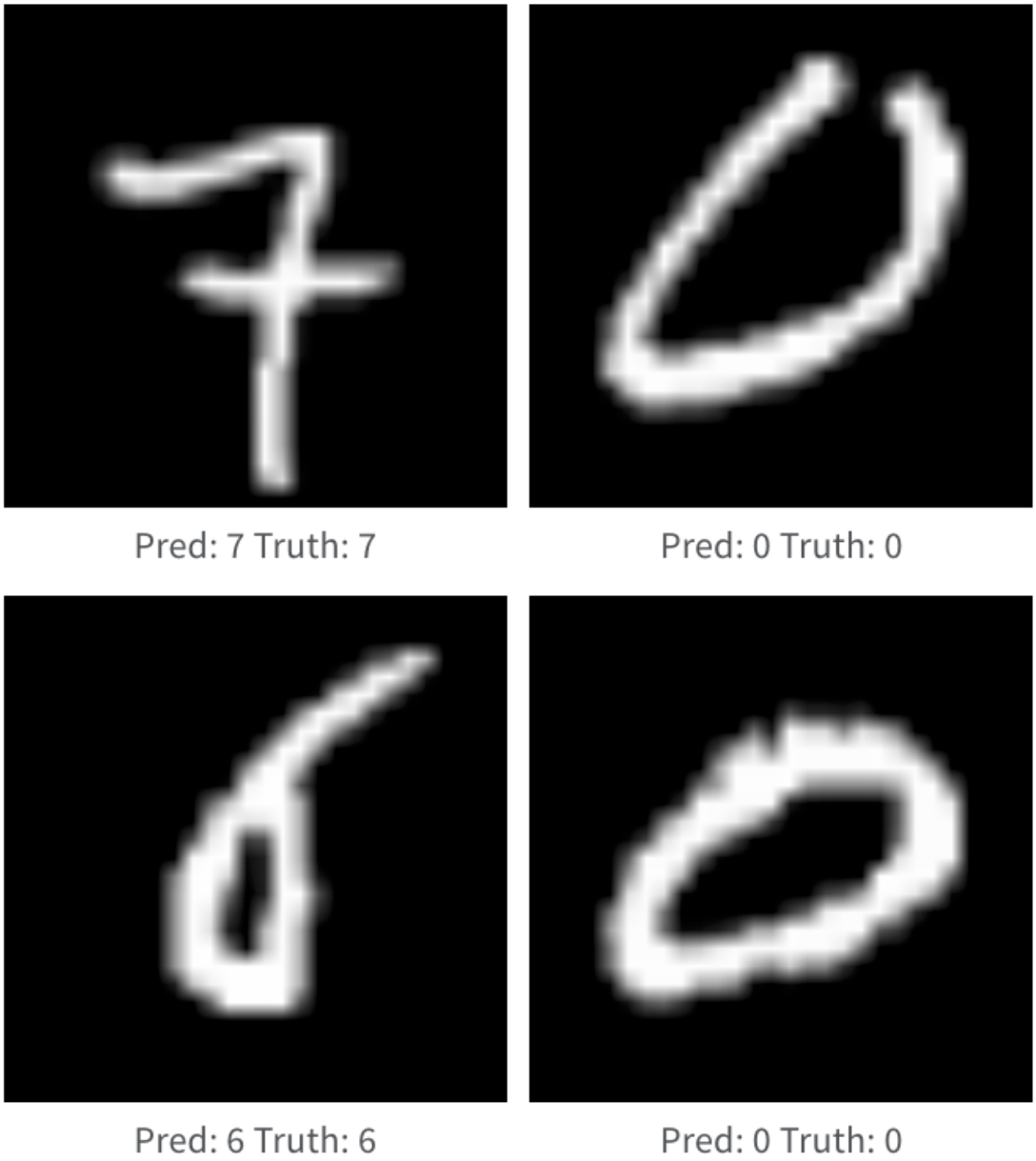

Fig. 10 Prediction and Ground truth by training the model with random weights.

Results

Method	Epochs	Accuracy
CNN on MIDAS dataset containing only digits, with a CosineAnnealingLR scheduler.	30	66.56
CNN on MNIST dataset with random weights, with a CosineAnnealingLR scheduler.	30	99.39
CNN on MNIST dataset with pretrained weights, with a CosineAnnealingLR scheduler.	30	99.34

Table 2 Results after using the CosineAnnealingLR scheduler.

Subtask 3

Implementation details

Given a dataset with images of digits.

Fig. 11 The directory structure of the folder.

Fig. 12 Images inside the folder of "0-zero".

The following transformations were applied to the dataset before using it as an input to train the network.

1. **Grayscale Conversion:** To maintain the same data distribution as MNIST, Grayscale was applied to convert 3-channel images to 1-channel images.

2. **Resize Images:** To train the model efficiently. The images were resized to H=28 and W=28.

3. **Normalize Images:** The training images are then converted to tensors and normalized by the mean (0.1307) and standard deviation (0.3081) to help the network get data within a range and reduce the skewness which helps learn faster and better.

Network details

- The network consists of 2 convolutions layers with input channels [1,32] and output channels [32,*64*] respectively. Each followed by a relu activation function. The maxpool pooling layer was applied after the output of the second convolution layer. Dropout was applied with a value of 0.25. The output is then flattened and a fully connected layer was applied with input features 9216 and output features 128. Dropout was applied with a value of 0.5. The fully connected layer with input features 128 and output features 10 as the final layer of the model.
- The network is trained for 30 epochs with a learning rate of 1 and optimizer adadelta.
- We used CosineAnnealingLR Scheduler: Min LR 0.8, Max LR 1, and warm-up 5.
- CosineAnnealingLR Scheduler increased the performance of the network as reported in Table2.
- The trained network is tested on the MNIST test set to select the best model and hyperparameters.

Training Strategy

- Training the model on the given dataset.
- Training the model on the processed dataset for 30 epochs with a learning rate of 1

Results

Method	Accuracy
CNN on MIDAS Dataset with random weights.	1.74
CNN on MIDAS Dataset with pretrained weights of MIDAS dataset containing only digits.	10.32

Table 3 Results of the model after training on the dataset.

Analysis

After visualizing the images inside the dataset, I found that images inside the folder are present in a particular manner i.e images of "0-zero" are very less present inside the folder "0". This made me clear

that the labels of the images are shuffled which results in poor accuracy on the MNIST test after training the model on the dataset.

Future Goals

1. Load the best-saved network weights from subtask 1 to then train on the MNIST train set and evaluate on the MNIST test.
2. Train a CNN with 224 by 224 as an input image for subtask 1 because the dataset provided for subtask1 is H=900 W=1200.
3. Explore a solution of subtask 3 as the performance is significantly affected by the change in images inside the folders.
4. Explore other possible network solutions for subtask 1 to get improvement in accuracy.

References

I want to give shoutouts to these works which enabled me to compile this report with various code examples and experimental results:

- Training a Classifier by Pytorch.
- Transfer Learning For Computer Vision Tutorial by Pytorch.
- Pytorch MNIST example by Pytorch.
- Weights & Biases Documentation by Weights & Biases.
- CosineAnnealingLR

Created with ❤️ on Weights & Biases.

https://wandb.ai/animesh-007/midas-task/reports/Report-for-MIDAS-Internship-Task-2--Vmlldzo1ODY1NjA?accessToken=jdcacr0ldkzegfg5swi1hyu5syabmjjadxa8p1wi859uwoughe1vwkjsz7g1tw7l