Investigating popular datasets and DNN architectures

Table of Contents

Introduction

This paper is a continuation of "A Survey on Deep Neural Network Security in an Embedded Context". In this paper, we will be investigating some of the datasets and models described in survey. This paper will cover environment setup, dataset preparation, model training, and model evaluation.

Environment Setup

Hardware

The following table describes the hardware used for this paper.

Component	Specification
CPU	Ryzen 5 5600
GPU	NVIDIA GeForce RTX 3070
RAM	32 GB

Software

The following table describes the relevant software used for this paper.

Component	Specification
OS	Windows 11
Python	3.8.5
PyTorch	1.8.1
CUDA	11.1

Setting up the Environment

Python was installed from the official Python website. PyTorch was installed using the following command (found on the PyTorch website):

```
pip3 install torch torchvision torchaudio --extra-index-url
https://download.pytorch.org/whl/cu116
```

To confirm successful installation, the following commands were run:

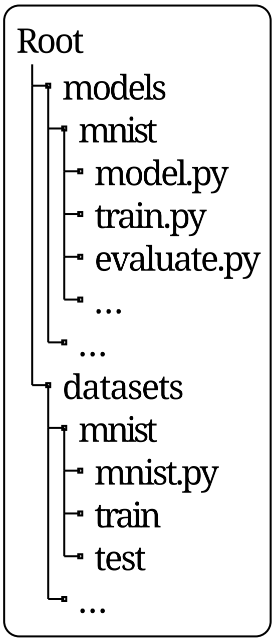
```
python3
>>> import torch
>>> torch.cuda.is_available()
```

```
True
```

Project Structure

The source code for this paper can be found on Github. The directory structure is split into two parts: the 'models' directory and the 'datasets' directory. The 'models' directory contains the source code for the models that will be trained and evaluated. The 'datasets' directory contains the different datasets used.

For example, the following shows files relevant to MNIST and the CNN-2-2 model.



```
Root
├ models
│ ├ mnist
│ │ ├ model.py
│ │ ├ train.py
│ │ ├ evaluate.py
│ │ └ ...
│ └ ...
└ datasets
  ├ mnist
  │ ├ mnist.py
  │ ├ train
  │ └ test
  └ ...
```

Datasets and Models

The following table describes the datasets and models that will be investigated in this paper.

Dataset	Model
MNIST	CNN with 2 convolutional layers and 2 fully connected layers (CNN-2-2)
CIFAR-10	-
ImageNet	-

These datasets and models were chosen because they were described in the survey paper and were the most common amongst the works discussed in the survey.

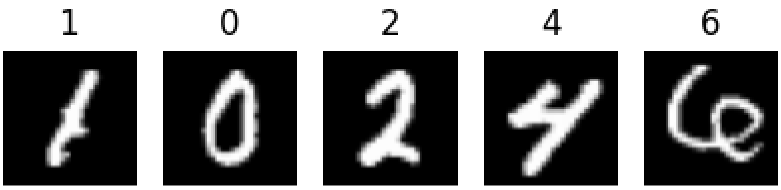
MNIST

The relevant files for MNIST can be found in the 'models/mnist/' and 'datasets/mnist/' directory.

Dataset

The MNIST dataset is a dataset of 60,000 28x28 grayscale images of the 10 digits, along with a test set of 10,000 images. The dataset is available from the MNIST section of the PyTorch documentation. The MNIST dataset is a popular dataset for testing image classification models and is often used as a "Hello World" example for image classification.

The following is a random sample of 5 images from the MNIST training dataset (these can be obtained by running the 'datasets/mnist/mnist.py' script).



Model

The model used for MNIST is a CNN with 2 convolutional layers and 2 fully connected layers (CNN-2-2). The model is defined in the 'models/mnist/model.py' file. A summary of the model is shown below (this can be obtained by running the 'models/mnist/model.py' script).

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 32, 24, 24]             832
            Conv2d-2             [-1, 64, 8, 8]          51,264
            Linear-3                 [-1, 1024]       1,049,600
            Linear-4                   [-1, 10]          10,250
================================================================
Total params: 1,111,946
Trainable params: 1,111,946
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.18
Params size (MB): 4.24
Estimated Total Size (MB): 4.42
----------------------------------------------------------------
```

Training

The model was trained using the 'models/mnist/train.py' script. The following table describes the training parameters used.

Parameter	**Value**
Batch Size	100
Learning Rate	0.001
Epochs	50
Optimizer	Adam
Loss Function	Cross Entropy