C++ & Apache MXNet – CNN Implementation for MNIST Dataset

Sanjay Singh

san.singhsanjay@gmail.com

C++ & Apache MXNet

Introduction

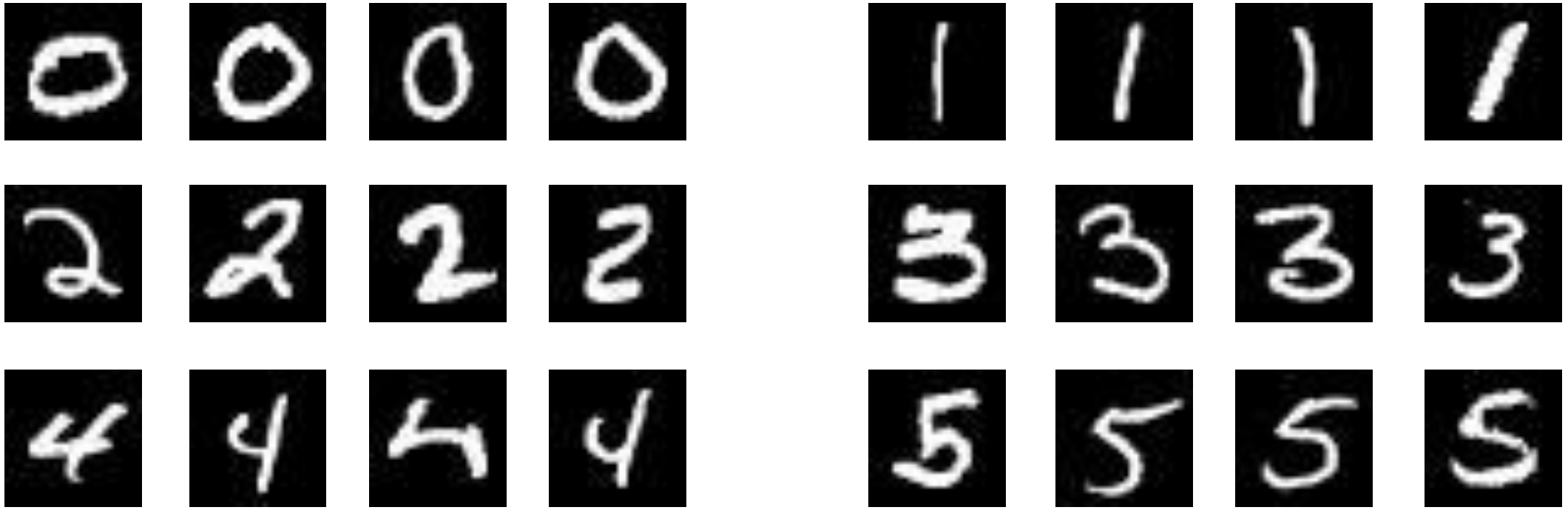
- C++:
  - C++ is a general purpose programming language and one of the world's most popular programming language.
  - It was created by Bjarne Stroustrup in year 1979 as an extension of C programming language which was created by Dennis Ritchie in year 1972.
  - C++ was initially standardized in year 1988 which was then amended by the C++03, C++11, C++14, C++17 and C++20. C++23 is the next planned standard.
  - C++ gives programmers a high level of control over system resources and memory.
  - C++ can be found in today's operating system, graphical user interface and embedded devices.
  - C++ is an object oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development cost.
  - C++ is portable and can be used to develop applications that can be adapted to multiple platforms.
  - C++ is extremely fast because it is a compiled language. Non-compiled languages have to be interpret at runtime which is comprised of 2 steps.
  - C++ is unsafe. This nature allows programmers to do more but also forces them to do more. For instance, C++ has no boundary checks on arrays and allows for improper type conversion. These things causes corrupted memory issues which is very hard to debug.

Introduction

- Apache MXNet:
  - Apache MXNet is an open-source deep learning framework used to train and deploy deep neural networks.
  - It is scalable to multiple GPUs (i.e., distributed training) and allows for fast model training.
  - It is available for eight programming languages. It has deep integration into Python and support for Scala, Julia, Clojure, Java, C++, R and Perl.
  - MXNet is supported by AWS (Amazon Web Services) and Microsoft Azure.
  - Amazon has chosen MXNet as deep learning framework of choice at AWS.
  - MXNet is also supported by several research institutions, such as MIT, Carnegie Mellon, etc.
- Why Machine Learning (ML) in C++:
  - C++ is the most efficient programming language.
  - C++ allows you to control and manage all resources, such as memory and each core of CPU & GPU.
  - All high level and the so called “flexible” deep learning frameworks, such as Tensorflow, Caffe, etc., are written in C++.
- Use Python for ML:
  - If speed is not a critical performance metric and / or you want to work interactively with data.

MNIST Dataset

- The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The database is also widely used for training and testing in the field of machine learning.
- Following are some of its sample images:

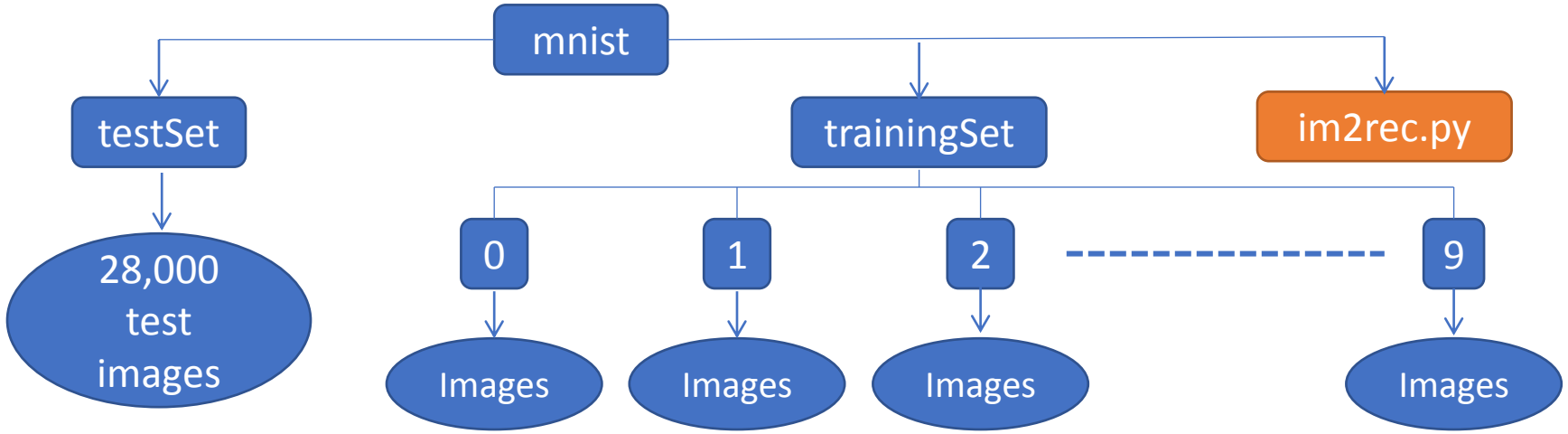


MNIST Dataset

- This dataset is downloaded from:
  https://www.kaggle.com/scolianni/mnistasjpg
- The size of this dataset is 38 MB.
- This dataset has total 70,010 images, out of which 28,000 images are given test set and 42,010 images are given in train set.
- Dimension of each given image is (28 x 28 x 3).

Execution Flow

1. “src/im2rec.py”:
   - This is a Python script provided officially by Apache MXNet.
   - This script creates a “.rec” file of the entire dataset. This “.rec” file can be passed to data iterators which automatically passes data in batches of fixed size for training to the neural network. Due to this “.rec” file, the process of passing data to the neural network becomes fast.
   - Following is the directory structure of MNIST dataset which is used in this project:



   - Like above, keep “im2rec.py” file under “mnist” directory, i.e., at the same level of trainingSet directory.

Execution Flow…

- Now, run the script from the terminal. Firstly, it will create a ".lst" file and then it will create the ".rec" file. Following are the commands:

```
$ python3 im2rec.py --list mnistData /home/sansingh/Downloads/DATASET/mnist/trainingSet --recursive --train-ratio 0.99
```

Above command will run im2rec.py script to create a ".lst" file (as –list option is used). The name of generated ".lst" file will be "mnistData". The entire data is kept under the "trainingSet" directory. Since data of each class is within its respective sub-directory having class name as directory name, thus "—recursive" option is used here. It is instructed to take 99% of this data set for training and 1% for validation. This command will create files: mnistData_train.lst and mnistData_val.lst.

```
$ python3 im2rec.py mnistData /home/sansingh/Downloads/DATASET/mnist/trainingSet --recursive
```

Above command will create ".rec" file. For this, we have run im2rec.py script and passed the name of .lst file, i.e., mnistData. Script will recursively visit each sub-directory under trainingSet directory and create ".rec" train and validation file having images and labels in an optimized form.

Execution Flow…

2. “src/train.cpp”:
   - This file has the C++ code and it is using Apache MXNet for CNN.
   - This program will make use of “.rec” files created in the last step and train the CNN model.
   - After training, it will save the model in “trained_model” directory.
3. “src/inference.cpp”:
   - This C++ program will load the trained CNN model saved in the last step to make inference on the test data.

Training

- CNN Architecture
- Epochs, Learning Rate, Batch Size, Optimizer, Loss Function
- Class-wise instances for training and validation

Result

- Training Accuracy epoch-wise
- Validation Accuracy epoch-wise
- Test Accuracy

Conclusion

- Template

Future Work

- Template

Template

- Template

Thank You