Hong Kong University of Science and Technology
COMP 5212: Machine Learning
Spring 2018

Project 2
Due: 6 April 2018, Friday, 11:59pm

1 Objective

The objective of this project is twofold:

1. To acquire a better understanding of deep learning by using a public-domain software package called TensorFlow.
2. To evaluate the performance of convolutional neural network (CNN) and convolutional autoencoder (CAE) models by conducting empirical study on simple image data.

2 Major Tasks

The project consists of the following tasks:

1. To install and learn to use TensorFlow.
2. To train a CNN model for image classification using the data set provided.
3. To train a CAE model for feature learning using the data set provided.
4. To write up a project report.

Each of these tasks will be elaborated in the following subsections.

2.1 TensorFlow Installation

Detailed guides on installing TensorFlow for different computing platforms can be found at the following website:

`http://www.tensorflow.org/install/`

Installing TensorFlow using Anaconda or PIP is the recommended way since it is simple and straightforward.

For this project, it suffices to use the CPU-only version (higher than 1.0) of TensorFlow with Python 3.x. You may also use TensorFlow with GPU support if you have a powerful GPU, but please note that the installation will be slightly more complicated.

You may also need to install `matplotlib` for plotting figures. However, other high-level machine learning frameworks such as Keras should not be used for this project.

2.2 Project Files

The image data and an initial Python program (`main.py`) are provided on the project page.

The images provided are selected from the EMNIST data set [1], which consists of 28×28 images of handwritten characters that belong to 47 classes.

The Python code provides an overall skeleton for the major project tasks. A simple neural network trained using the batch gradient descent algorithm is provided in the code. It will help you get familiar with data loading and some basic TensorFlow operations. Please feel free to add your own functions and classes or even create other files. However, you should not use extra data or other pretrained models for the project.

2.3 Task 1: CNN Model

You are asked to train a CNN classifier using the training set (`data_classifier_train.npz`). The training and test sets contain 40,000 and 4,000 images, respectively.

You are recommended to use the following CNN architecture which contains four convolutional layers and one fully connected (FC) layer:

Layer Name	Description	# Filters	Filter Size	Stride
Input	Input image	NA	NA	NA
Conv1	ConvLayer	32	3×3	1
Conv2	ConvLayer	32	5×5	2
Conv3	ConvLayer	64	3×3	1
Conv4	ConvLayer	64	5×5	2
FC	1024-unit fc	NA	NA	NA
Output	Prediction	NA	NA	NA

The rectified linear function should be used as the activation function for all the hidden layers. You need to choose a good initialization for the network weights, or else the model might fail to improve during training. One common choice is the Xavier initialization [2].

You are required to train the model using the gradient descent with momentum (SGD with momentum) algorithm [4] on the softmax cross-entropy loss. You should implement the algorithm using basic TensorFlow operations instead of directly calling `tf.train.MomentumOptimizer`. You will also need to choose a proper mini-batch size and the number of training steps. Remember to randomly shuffle the training set before splitting it into mini-batches in each epoch. The random seeds should be fixed so that the experiment results can be reproduced.

You may use cross validation to tune the parameters. However, as is always the case, the test set (`data_classifier_test.npz`) should not be used for parameter tuning. It should only be used for evaluating the performance of the model after training.

2.4 Task 2: CAE Model

An autoencoder can extract low-dimensional features from the original data and, as a result, can achieve noise reduction. You are asked to train a CAE model [3] using the training data in `data_autoencoder_train.npz` and then evaluate the model using a separate set of images in `data_autoencoder_eval.npz`. The training and evaluation sets contain 70,000 and 1,000 images, respectively.

The encoder part of the CAE should have the following architecture:

Layer Name	Description	# Filters	Filter Size	Stride
Input	Input image	NA	NA	NA
Enc1	ConvLayer	32	5×5	2
Enc2	ConvLayer	64	5×5	2
Enc3	ConvLayer	2	3×3	1

The decoder part consists of deconvolutional layers [5] and should have an architecture symmetric to the encoder. The output is the reconstructed image. The mean squared error (MSE) loss between the input image and the reconstructed image should be used for training. As for the activation functions, the rectified linear function should be used for all the hidden layers and the identity function (i.e., linear units) should be used for the output layer. Other parameters should be tuned like in Task 1.

After training, you should evaluate the model on the evaluation set and plot some feature maps as well as reconstructed images.

2.5 Report Writing

In your report, you are expected to present the parameter settings and the experiment results for the two tasks. Besides reporting the accuracy and loss (for both training and test/evaluation data) in numbers, graphical aids should also be used to show the performance over time for different parameter settings. Please also indicate the type of computing equipment you use (CPU or GPU) and report the model training time. For Task 2, you should also show a few feature maps and reconstructed images produced by the autoencoder.

3 Project Submission

Project submission should be done electronically using the Course Assignment Submission System (CASS):

`http://cssystem.cse.ust.hk/UGuides/cass/student.html`

There should be two files in your submission with the following naming convention required:

1. **Project report** (with filename `report`): preferably in PDF format.
2. **Source code and a README file** (with filename `code`): all necessary code for running

your program as well as a brief user guide for the TA to run the programs easily to verify your results, all compressed into a single ZIP or RAR file. The data should not be submitted to keep the file size small.

When multiple versions with the same filename are submitted, only the latest version according to the timestamp will be used for grading. Files not adhering to the naming convention above will be ignored.

4 Grading Scheme

This project will be counted towards 8% of your final course grade. The maximum scores for different tasks are as follows:

- Training the CNN model [35 points total]:
  - Build the CNN model [10 points]
  - Develop the mini-batch SGD with momentum algorithm [10 points]
  - Tune the parameters for CNN training using cross validation techniques [10 points]
  - Compute the accuracy and loss of the CNN model on the training and test sets [5 points]
- Training the CAE model [35 points total]:
  - Build the CAE model [15 points]
  - Tune the parameters for CAE training [10 points]
  - Compute the feature maps and reconstructed images for CAE training [5 points]
  - Compute the loss of the CAE model on the training and evaluation sets [5 points]
- Project report [30 points total]:
  - Present the parameter settings for CNN training [3 points]
  - Explain your mini-batch stochastic optimization algorithm [5 points]
  - Show the CNN model's performance over time for different parameter settings [5 points]
  - Report the accuracy and loss of the CNN model [2 points]
  - Present the parameter settings for CAE training [3 points]
  - Report the loss of the CAE model [2 points]
  - Show a few feature maps and reconstructed images produced by the CAE model [5 points]
  - Report the model training time for CNN and CAE [5 points]

Late submission will be accepted but with penalty. The late penalty is deduction of one point (out of a maximum of 100 points) for every minute late.

5 Academic Integrity

Please read carefully the relevant web pages linked from the course website.

While you may discuss with your classmates on general ideas about the project, your submission should be based on your own independent effort. In case you seek help from any person or reference source, you should state it clearly in your submission. Failure to do so is considered plagiarism which will lead to appropriate disciplinary actions.

References

[1] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[3] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 52–59, 2011.

[4] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[5] Matthew D. Zeiler, Dilip Krishnan, Graham W. Taylor, and Rob Fergus. Deconvolutional networks. In *Proceedings of the IEEE Computer Science Conference on Computer Vision and Pattern Recognition*, pages 2528–2535, 2010.