CSCI433/933: Machine Learning Algorithms and Applications
Assignment Problem Set III (Individual)

Instructors:Professor [redacted]
School of Computing and Information Technology
University of Wollongong

Due date: Saturday 29th May, at 18:00 Hrs

Deep Neural Network Design, Implementation and Report (60 Marks worth 20 % of total subject marks)

Aims and Objectives

This assignment aims at evaluating basic familiarity with fundamental concepts and implementation of deep neural networks and statistical machine learning. On completion of this assignment, you should be able to demonstrate basic mastery of:

- concepts of deep autoencoder, feature extraction, SVM, convolutional autoencoder;
- implementation of machine learning algorithms using Tensorflow, Keras, SkLearn.

Introduction

Autoencoder is a popular unsupervised learning technique used to learn data representations. Specifically, a neural network architecture is designed such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input. To build an autoencoder, we need three components: an encoder to compress the data, a decoder to decompress the data, and a loss function to measure the data reconstruction error. There are different types of autoencoder models with the variation of the encoder/decoder architectures. This assignment will focus mainly on two popular autoencoder models, which are deep fully-connected autoencoder and deep convolutional autoencoder.

What needs to be done

1. Implement a multi-layer fully-connected autoencoder using Tensorflow and Keras (**Use Tensorflow 2.0 or higher**):

- (**15 Marks:**) Load fashion_mnist using keras. Fashion_mnist is a dataset of article images-consisting of a training set of 60,000 examples and a test of 10,000 examples. Each image is a 28×28 grayscale image, associated with a label from 10 classes. Change the data type of each image as float32 and normalize the pixel values to $[0, 1]$. Hint: x_train, x_test = x_train.astype('float32')/255.0, x_test.astype('float32')/255.0.
- Use Tensorflow and keras to implement a **six-layer fully-connected** autoencoder based on fashion_mnist dataset. The encoder and decoder of the autoencoder both consists of three

layers. Each input image is flattened into dimensionality of 784. The three encoder layers are with output dimensionality of 128, 64, 32. The three decoder layers are with output dimensionality of 64, 128, 784. After each of the six layers, nonlinear function **ReLU** is applied.

- Train the network using **mean_squared_error** as loss function and **Adam** as optimizer. The **batch_size** should be set to 256. Train the network for 30 epochs. Hint: for each epoch, the training data should be randomly shuffled.
- Report the training error and test error for each epoch. Randomly choose two test images, display and compare the original images and reconstructed images. Show your images in your report.
- (**5 Marks:**) Experiment with varying the depth (more layers) and width (higher dimensional hidden layers) of the autoencoder and monitor the change of training and testing losses. Report your finding.

2. Train SVM classifier based on the image representations extracted from the above autoencoder (**15 Marks**):

- Once the above fully-connected autoencoder is trained, for each image, extract the 32-dimensional hidden vector (the output of the ReLU after the third encoder layer) as the image representation.
- Train a linear SVM classifier on the training images of fashion_mnist based on the 32-dimensional features. Tune the hyper-parameter 'C' using cross-validation; report your finding. Report the training accuracy. SkLearn is recommended.
- Test the trained SVM on the test images of fashion_mnist. Report the testing accuracy.
- Experiment by using a kernel-based SVM and compare the performance with linear SVM. Report your finding.

3. Train deep convolutional autoencoder using Tensorflow and Keras (**15 Marks**):

- Load and pre-process fashion_mnist as in fully-connected autoencoder.
- Train a **six-layer convolutional** autoencoder. Different from fully-connected autoencoder, each encoder and decoder layer is now a convolutional layer instead of fully-connected layer. Again, the autoencoder consists of three encoder layers and three decoder layers. Each input image to the autoencoder is now 3D ($28 \times 28 \times 1$). For each convolutional layer, use convolutional kernel of 3×3**, stride = 1, padding='same'**. After the first and second encoder convolutional layers, use $(2, 2)$ **max pooling** to downsample the feature maps and after the first and second decoder convolutional layers, use $(2, 2)$ **upsampling2d** operation to upsample the feature maps. Choose proper number of neurons for each convolutional layer, e.g. 16, 24. Each convolutional layer uses activation function **ReLu**.
- Train the network using **mean_squared_error** as loss function and **Adam** as optimizer. The **batch_size** should be set to 256. Train the network for 15 epochs. Hint: for each epoch, the training data should be randomly shuffled.
- Report the training error and test error for each epoch. Randomly choose two test images, display and compare the original images and reconstructed images. Show the images in your report.

4. Write a report of no more than four (4) pages to document the experiments, results and your conclusions. Your report **MUST** include an introduction about autoencoders and support vector machines **(10 Marks)**. Note that the quality of your conclusions is important if you wish you score a good mark. Your report must be structured and written to show each experiment you carried out (Hint: the template used in Assignment 2 is useful here). Furthermore, the writing must intelligible, thus allowing others to reproduce your experiments and results.

Submission instructions

Submission:
Due date: Saturday 29th May 2021, 18:00 Hrs.

1. Prepare three executable python files, i.e. deep_autoencoder.py, svm.py, conv_autoencoder.py. All the implementations should be based on python3.
2. Prepare a PDF file containing your 3-page report. Use a format similar to that given in the template for Assignment 2.
3. These files should be compresssed as a ZIP (or RAR) file and submitted on Moodle on or before the due date and time.

Late submission
This assignment does not have opportunity for late submission or extension because it is the last assignment before examination. You need to submit what you have accomplished by the due date and time.

Plagiarism:
A plagiarised assignment will receive a zero mark and be penalised according to the university rules. Plagiarism detection software may be used for this assignment.