School of Computer Science
The University of Adelaide
Artificial Intelligence
Assignment 2

Semester 1, 2019
Due 11:55pm, Friday 17th May 2019

Introduction

In this assignment, you will develop classification models to classify noisy input images into two classes: square or circle. Examples are shown in Fig. 1.

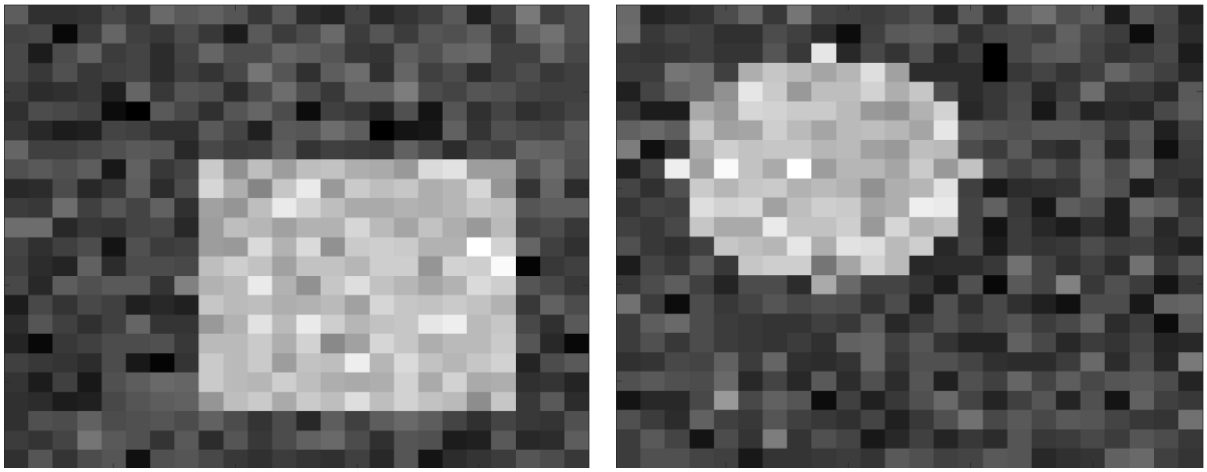

Figure 1: Samples of noisy images labelled as square (left) and circle (right).

Your classification models will use the training and testing sets (that are available with this assignment at https://myuni.adelaide.edu.au/courses/45386/assignments) containing many image samples labelled as square or circle.

Assignment

Your task is to write Python code which will train and validate the following two classification models:

1) **K-Nearest neighbour (KNN) classifier [30 marks].**
   For the KNN classifier, you can only use standard Python libraries (e.g., `numpy`) in order to implement all aspects of the training and testing algorithms.

   Using `matplotlib`, plot a graph of the evolution of classification accuracy for the training and testing sets as a function of K, where K = 1 to 10.
   Clearly identify the value of K for which generalisation is best.

   Undergraduates can use whatever algorithm they see fit (including exhaustive search) and all 30 marks will be available.

   Post-graduates who implement exhaustive search will be eligible for only 15/30 marks. To be eligible for 30/30 you must implement a K-d tree to store and search the database.

2) **Convolutional neural network (CNN) classifier [30 marks].**
   For the convolutional neural network, you should use Tensorflow within Jupyter Notebook by modifying the Multilayer Perceptron program supplied with this assignment. Instructions for installation of Python 3.7, Jupyter and TensorFlow (via a package called miniconda) are in a separate sheet supplied with this assignment.

   You should modify the code so that it implements the LeNet CNN structure to that was presented in lectures. In particular, the LeNet architecture should comprise two convolutional layers (5x5 convolutions), and two hidden full-connected (dense) layers in addition to the output layer. After each convolutional layer the architecture should use max pooling to reduce the size by a factor of 2 in each axis. After each pooling operation you should use a RELU (Rectified Linear Unit) activation function. The LeNet will also have three dense layers forming a Multilayer Perceptron (MLP) classifier (you can use the ones already in the sample implementation. The size of the two hidden-layers in the MLP must be 2x and x (where you will need to test different values of x by changing the code or writing a suitable function). Of course the output layer will have a single neuron.

Undergraduates should experiment training two LeNets, with 'x' = 20, 'x' = 50 and ensure the results are written up carefully, comparing the two networks.

Postgraduates should experiment training three LeNets, with 'x' = 20, 'x' = 50. and 'x' = 100 and report the results for all three variants, commenting carefully on each.

Using `matplotlib`, plot a graph of the evolution of accuracy for the training and testing sets as a function of the number of epochs, for each of the CNNs you train (up to a maximum of 200 epochs).

Sample python code that trains and tests a multi-layer perceptron classifier (and can run in a Jupyter Notebook session) is provided with the assignment specification at https://myuni.adelaide.edu.au/courses/45386/assignments. You should modify this code to produce your own program.

Submission

You must submit via MyUni, by the due date, two files:

1. A zip file or Jupyter notebook file (.ipynb) containing your code with the two classifiers and all implementations described above.
2. A pdf file with a short (no more than 2 pages) written report detailing your implementation, your results, and an interpretation of the results. The results you should include are:
    a. The training and testing accuracies at for KNN, K=1 to 10
    b. The training and testing accuracies for CNN, x=20, x=50 (and x=100 for postgrads)

    This should take the form of a table:

Classifier	Training Accuracy	Testing Accuracy
K-NN (k = 1)		
...		
K-NN (k = 10)		
CNN (x = 20)		
CNN (x = 50)		
CNN (x = 100)		

The implementations are worth 30 marks each (see above) and the report is worth 40 marks. For full marks your report should include a careful and critical analysis of your observations about the performance of the different algorithms and algorithm settings.

This assignment is due **11.55pm on Friday 17th May, 2019**. If your submission Is late, the maximum mark you can obtain will be reduced by 25% per day (or part thereof) past the due date or any extension you are granted.