AMME4701 Computer Vision and Image Processing
Tutorial Activity Week 7, 2022

THE UNIVERSITY OF SYDNEY

Machine Learning and Computer Vision

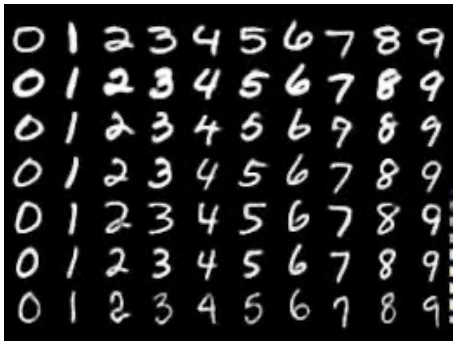
- This tutorial activity is to be completed during the Week 7 tutorial (Thursday 16th September 2022)
- When you have completed the activity, you will need to show your results to the tutor who will check your work and check that you have sufficiently completed the task

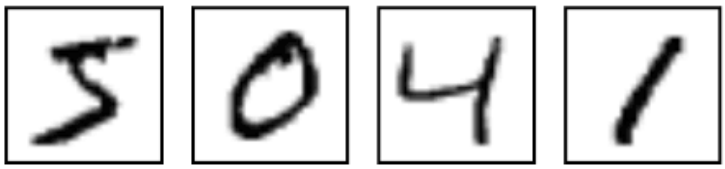
Objectives

- Character recognition is a machine learning application that is used every day, whether it be when you post a letter in a mailbox, enter a ticketless carpark, or drive too fast down a street in your red Corolla and get booked for speeding.
- This tutorial aims to give you an introduction into the processes undertaken in order to develop such high-level machine learning algorithms. In this tutorial, we focus on feature extraction, model training and validation for the task of image classification for recognising handwritten digits.

MNIST Handwritten Character Dataset

Download the MNIST dataset [1] from Canvas, which contains a large number of 28-by-28 pixel images of handwritten digits and their associated labels.





The MATLAB file contains several variables corresponding to the dataset:

- **data_train** - 28x28xK matrix containing the images used during training of the machine learning algorithm for classification. Each layer in the matrix represents an individual sample.
- **labels_train –** K dimensional vector containing the label of the corresponding image in **data_train.**
- **data_test** – 28x28xP matrix containing the images used during the final testing of the algorithm. It is NOT to be used during training as it will be indicative of deploying the model in the field.
- **labels_test –** P dimensional vector containing the label of the corresponding image in **data_test.**

Visualise some of the examples in the dataset using `imshow` and get used to the image format and labels. Subsample **data_train** (since it contains a large number of examples) down to a manageable size. Each class should be balanced. Don't forget to select the corresponding instances from **labels_train** as well.

Feature Extraction

After observing the images, think about what kind of features would be suitable for input into your model. For example:

- Raw data – think about the effect of image resolution and whether blurring the data could help.
- Spatial histograms along the columns and rows.
- Image statistics such the mean of the image or the position of the modes of the spatial histograms.
- What other features might help?

Model Training and Validation

The aim of the classification system you are trying to develop is to be able to classify an input image into 1 of 10 classes that represent the digits [0-9].

Metrics

- Implement the function **ML_AnalyseModel** that assesses the predictive capabilities of a model.

```
[confmat, acc, prec, rec, f1score] = ML_AnalyseModel(pred_, gt_)
```

- Inputs
  - **pred_** - vector of integers indicating the predicted label.
  - **gt_** - vector of integers indicating the ground truth label.
- Outputs
  - **confmat** – confusion matrix. 10x10 matrix of predicted vs ground truth classes.
  - **acc** – accuracy (a scalar value)
  - **prec** – precision (a 10x1 vector of values per class)
  - **rec** – recall (a 10x1 vector of values per class)
  - **f1score** – f1score. (a 10x1 vector of values per class)

Features

- Given your observations and intuitions behind what kind of features would be useful for your classifier, select a method and generate the matrix **data_features** which contains the extracted features. The matrix should contain:
  - N rows, where N is the number of images
  - M columns, where M is the dimensionality of the extracted feature
- Think about concatenating several features together and whether this may help in discriminating different classes.
- Make sure that the order of the features has the same ordering as the ground truth labels.

Model Validation

- Recall the process of k-fold cross validation. Write the function **ML_CrossVal_KFold** that generates the indices of the data used for training and validation.

```
[ indices_train, indices_val ] = ML_CrossVal_KFold( K_, N_ )
```

- Inputs
  - **K_** - integer dictating how many folds are to be used during validation.
  - **N_** - number of training samples in the dataset.
- Outputs
  - **indices_train** – K_ x V matrix, containing the indices of the data used during training the machine learning algorithms. V is the number of examples used during each training fold.
  - **indices_val** – K_ x V2 matrix, containing the indices of the data used during training the machine learning algorithms. V2 is the number of examples used during validating/testing for each fold.
- Helpful functions
  - **randperm** – generates a random permutation of a vector.

Model Training

- Select a machine learning approach such as KNN, SVMs or Decision Trees. Become familiar with how to use the functions by reading the documentation and take note of what variables can be tuned.
- Helpful functions:
  - **fitcknn** – k nearest neighbours
  - **fitctree** – decision tree
  - **fitcecoc** – multi-class support vector machine
  - **predict** – uses a trained model and applies it to new data to predict labels

- Write a loop that goes through each of the cross validation folds, trains a model on the relevant data, predicts labels on the validation set and assesses the model. Choose an appropriate and effective method by which you can visualise the results.

```
ML_CrossVal_KFold( K, N )
for i = 1:1:K
     Extract training data and labels for fold i
     Train model
     Extract validation data and labels for fold i
     Predict labels
     ML_AnalyseModel
     Store statistics for this fold
Average the statistics over the different folds
```

- Nest this loop inside another loop that changes one of the parameters of your chosen machine learning algorithm. For example; the number of nearest neighbours in KNN, number of leaves/depth in decision trees and kernel type in SVMs.

```
ML_CrossVal_KFold( K, N )
For param = 1:1:W
     for i = 1:1:K
          Extract training data and labels for fold i
          Train model
          Extract validation data and labels for fold i
          Predict labels
          ML_AnalyseModel
     Average the statistics over the various models
     Store statistics for chosen param
```

- Generate a plot that effectively shows how model performance changes as you vary the parameter. For example, a plot of f1 score vs K for KNN for each of the classes
- Using **imagesc** or any other method of visualisation, determine what are the most commonly misclassified classes in your dataset for your best model. Why do you think these errors are occurring and can you explain these in terms of the chosen features?
- Apply your model to the held out dataset and compare the results against those obtained from k-fold cross validation. Are they similar? Why or why not?
- Analyse the variances of the statistics across all the folds for the parameter you have deemed to be the best.
- Helpful functions for visualisation
    - **imagesc**
    - **boxplot**
    - **plot**
    - **bar**

References and Further Reading:

[1] J. Liang, T. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[2] MNIST Database: http://yann.lecun.com/exdb/mnist/