Image Classification (Multi-Class)

Assignment Due Week 11[1]

All assignment work must use the **Python/Keras**[2] development environment.

Deliverables

The main purpose of this assignment is to introduce students to the practical aspects of developing deep learning based computer vision systems within the **Python/Keras** environment. The assignment is worth **40%** of your overall module mark[3].

Implement the tasks outlined and record your observations. Answer all questions and develop appropriate code based solutions when requested. Submit a report detailing the **rational, design and testing** of the approach taken in developing your solution. This document[4] should include your report details (*introduction, rational, design, testing [including the test procedures used to evaluate the effectiveness of the method chosen], and conclusion*).

A key element of this assignment is your ability to design and implement your own test strategy. Please submit[5] a **single pdf document** via **Loop (Moodle)**.

- **Students should include full code listings in the appendix**
- **Plus** a **link to online drive** containing all assignment related material (code / data / test results).

[1] Please refer to the *Module Protocol* for the strict no late assignment policy. Please refer to Loop for the **exact submission deadline and the detailed marking scheme (rubric).**
[2] This is installed on all the PC's in the Schools own Computer Labs. These are available for use by ALL students registered on this module.
[3] While I will endeavour to get your coursework marks back to you as soon as possible, this is resource dependent and may not occur prior to the exam board.
[4] The format is as a technical report. While there is no page quantity requirement, the final report should normally be no more than 40 pages. See Appendix A.
[5] All computer accounts should use the usernames and passwords issued by ISS at registration (notify the Paul Wogan <paul.wogan@dcu.ie> if you have any problems with this).

Getting Started

Read and understand the *Python/Keras* requirements/installation instructions as indicated on the relevant websites. We will use two datasets.

Dataset #1 (ImageNette): Based on fast.ai's ImageNette [1] dataset, a subset of 10 easily classified objects from ImageNet [2]. In this assignment we will use a subset of 6 classes [*Church, English_springer, garbage_truck, gas_pump, parachute* and *tench*], see Figure 1. This dataset is on the EE544 module drive in the *assignment* folder (**imagenette_6class.zip**). Data is organised in three main sub-folders: **train, validation** and **test.** Our training data consist of 1200 images for each of the classes, we have 100 validation images per class and 50 test images per class.

Dataset #2 (food-101): Based on original food-101 [3] dataset, which has 101 food categories. All images are rescaled to have a maximum side length of 512 pixels. We will use a subset of 4 food categories [*chicken_curry, hamburger, omelette* and *waffles*] for this assignment, see Figure 2. This dataset is on the EE544 module drive in the *assignment* folder (**food101_4class.zip**). Data is organised in three main sub-folders: **train, validation** and **test.** Our training data consist of 1000 images for each of the classes, we have ~500 validation images per class and ~500 test images per class. The dataset was not cleaned (on purpose), and thus still contain some amount of noise. This comes mostly in the form of intense colours and sometimes the wrong labels. Note the dataset is not perfect which makes the problem more challenging, but we will work with the assigned labels).

Do not edit the datasets; they must be used as presented.

Programme development[6] should use the **training** and **validation** (pseudo test data to allow tuning of our hyperparameters) datasets**.** Note that final system accuracy measure should only apply to the data generated on the previously unused **test** data set when applied to your finalised system (**this data set should not be used for programme development**).

The assignment is in two sections. The first focuses on developing a basic VGG-lite CNN that is **trained from scratch**. The second part will focus on the development of a high accuracy and computationally efficient solutions using **fine-tuning based transfer learning** (pre-trained on ImageNet).

[6] Using a CPU only PC will result in long run times for these tasks. If you have access to GPU [Dettmers, servethehome] on your PC then this is a much better option. Alternatively, you can make use of cloud-based services such as **Colab** (Colaboratory: **free Jupyter notebook environment,** but time limited) and AWS (Amazon paid Web Services). For details on using Keras with GPU on Amazon EC2 see [CS231n, Brownlee]**.** Limited access to a high end **Nvidia Tesla K40C GPU** is available via the School of Electronic Engineering via **Linux** (Ubuntu). See https://sites.google.com/dcu.ie/ee-gpgpu/ for details. Contact conor.mcardle@dcu.ie for any additional information.

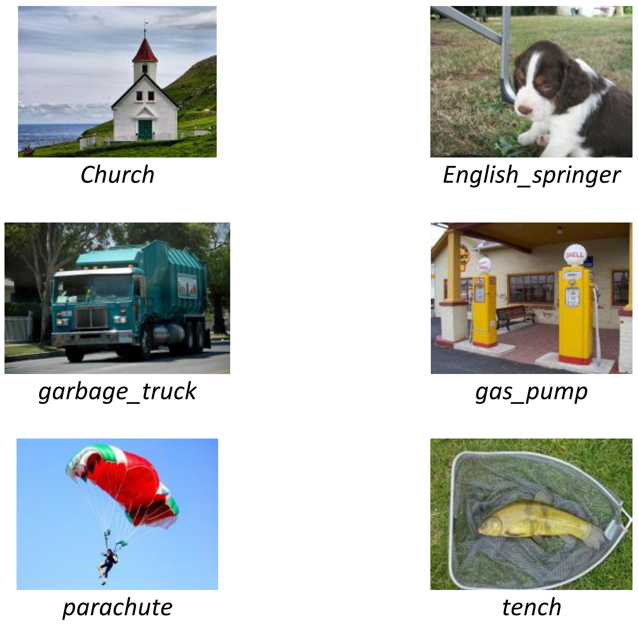

Figure 1. Dataset #1: 6 classes from ImageNette [1].

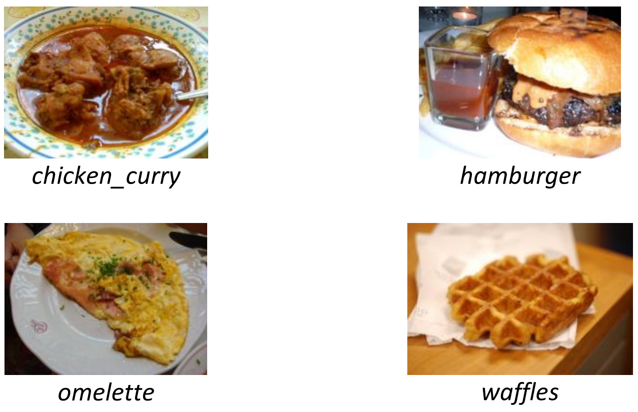

Figure 2. Dataset #2: 4 classes from Food-101 [3].

1: Multi-class Image Classification [Training from Scratch using ImageNette]

Complete all sections, **justifying all engineering design choices.** In particular, discuss the relevance of your network architecture, your choice of optimizer and all the hyper-parameters used. This section will use **Dataset #1.** In addition to the training and validation accuracy and loss diagrams your solution must also produce all the metrics listed below (see scikit-learn metrics for details) for each section.

Required Diagrams: Illustrate you answer with appropriate training and validation accuracy and loss diagrams.

Required Metrics: Final validation accuracy and loss, final test[7] accuracy and loss, Precision, Recall, F1 score, confusion matrix, the networks computational speed and environmental impact (CO_2 equivalent emissions).

Required Models: All models developed should be saved in Hierarchical Data Format (HDF5) (.h5) format.

Layer	Number of output filters
2D (3x3) convolution layer	32
2D (3x3) convolution layer	32
2D (2x2) Pooling	
2D (3x3) convolution layer	64
2D (3x3) convolution layer	64
2D (2x2) Pooling	
Flatten	
Fully-connected NN layer	512
Fully-connected NN layer (Prediction)	Number of classes
Convolution layers should use Xavier (glorot) uniform kernel initialization, a dilation rate of 1, 'same' padding, strides of (height,width)= (1,1) and 'relu' activation. Note the first layer will need to account for the input shape of the data been examined.	
Pooling layers will use (2x2) max pooling with (ie a pool size of (vertical, horizontal) = (2,2)).	
Fully-connected layers will also use Xavier (glorot) uniform kernel initialization and 'relu' activations.	

Figure 3: Baseline VGG-lite[8] CNN Structure

[7] Note that final system accuracy measure should only apply to the data generated on the previously unused **test** data set when applied to your finalised system (**this data set should not be used for programme development**).

a) Design and develop a simple **VGG-lite** (based on [4], which in turn is based on VGG-16D CNN [5]) baseline convolutional neural network architecture **as per the CNN structure illustrated in Figure 3** to classify the ImageNette data. Illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file.

[Marks: 5/40]

b) Apply **dropout** to the baseline network developed in (a) and illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file. What conclusions can you draw from using dropout?

[Marks: 3/40]

c) Apply **data augmentation** to the baseline network developed in (a) and illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file. What conclusions can you draw from using data augmentation?

[Marks: 3/40]

d) Apply **batch normalization** to the baseline network developed in (a) and illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file. What conclusions can you draw from using batch normalization?

[Marks: 3/40]

e) Using a combination of some (or all) of dropout, data augmentation and batch normalization in conjunction with the baseline network developed in (a) optimize the networks performance (i.e. best performing model) for the dataset provided. Save your final model as a HDF5 file. Illustrate the networks performance using the required diagrams and metrics listed previously. What conclusions can you draw from the final design configuration?

[Marks: 6/40]

[8] The default input size for this model is 224x224.

2: Food-101 Classification using Fine-Tuning based Transfer Learning[9]

Complete all sections, **justifying all engineering design choices.** In particular, discuss the relevance of your network architecture, your choice of optimizer and all the hyper-parameters used. This section will use **Dataset #2.** In addition to the training and validation accuracy and loss diagrams your solution must also produce all the metrics listed below (see scikit-learn metrics for details) for each section.

Required Diagrams: Illustrate you answer with appropriate training and validation accuracy and loss diagrams.

Required Metrics: final validation accuracy and loss, final test[10] accuracy and loss, Precision, Recall, F1 score, confusion matrix, the networks computational speed and environmental impact (CO_2 equivalent emissions).

Required Models: All models developed should be saved in Hierarchical Data Format (HDF5) (.h5) format.

a) Implement a **Resnet-50**[11] based **fine-tuning based transfer learning CNN architecture** to optimise the food classification task. During fine-tuning, only retrain the last main convolutional bloc: i.e. the **res5c block**. Illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file.
[Marks: 15/40]

b) Illustrate the performance of your network (as developed in part 2(a)) by applying it to **previously unseen images** of the food classes that you have acquired yourself. Comment on the performance of the network when applied to these "in the wild" images.
[Marks: 5/40]

[9] Pretrained on ImageNet

[10] Note that final system accuracy measure should only apply to the data generated on the previously unused **test** data set when applied to your finalised system (this data set should not be used for programme development).

[11] The default input size for this model is 224x224.

References

1. Jeremy Howard (2019), “Fast.ai Imagenette”, https://github.com/fastai/imagenette. Please refer to usage LICENSE
2. Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L. (2009), “ImageNet: A Large-Scale Hierarchical Image Database”, CVPR09
3. Lukas Bossard and Matthieu Guillaumin and Luc Van Gool (2014), “Food-101 - Mining Discriminative Components with Random Forests”, ECCV 2014
4. Adrian Rosebrock (2019), Deep Learning for Computer Vision with Python.
5. Karen Simonyan, Andrew Zisserman (2014), “Very Deep Convolutional Networks for Large-Scale Image Recognition”, arXiv:1409.1556
6. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna (2015), “Rethinking the Inception Architecture for Computer Vision” arXiv:1512.00567
7. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015), “Deep Residual Learning for Image Recognition”, arXiv:1512.03385
8. Francois Chollet (2017), Deep Learning with Python
9. Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems

Appendix A: Assignment Report

The technical report structure [i.e. introduction, rational, design, testing, and conclusion] is **just for guidance.** The key to your assignment report is that you **fully and clearly answer each questions given in your assignment sheet** and **provide the appropriate evidence of any conclusion through experimentation / testing**.

Key to any technical report is the **precision** of your answer. As well as the engineering content, the report will be evaluated based on the **clarity and communication** of the ideas involved. Please reference any work that is not your own (there are many bibliography formats, select one and stick to it).

All quantitative results, including the test procedures used to evaluate the effectiveness of the method chosen, should also be included in the report. A key element of this assignment is your ability to design and implement your own test strategy where required.

This is an **individual assignment** and **each student must produce their own report**.

Selected students **may be subjected to interview** and/or demonstration.

REMEMBER:

- Everything you write has to be in your own words
- All ideas, paraphrases of other people's words must be correctly attributed in the body of the report and in the references
- No reuse of 3rd party code – all coding solutions presented for grading must be your own work.