EEN1001 Computer Vision Assignment 24/25

Assignment Due Week 10[1]

All assignment work **must use** the **Python/Keras (Tensorflow)[2]** development environment (as outlined in the course notes).

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 **25%** 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 text code listings in the appendix**
- **Plus** a **link within your PDF report to an online drive** containing all assignment related material (code / data / test results).

Please use the naming convention **<your Student Name>_<your Student ID>.pdf** for your submission. A completed ***Coursework Assignment Cover Sheet*** (an example coversheet can be found in the shared drive, i.e. *CV_coversheet.pdf*) must accompany each assignment submission for it to be **considered valid** (**this should be the first page of your report**).

[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.**
[4] The format is as a technical report, there is no page quantity requirement. See Appendix.
[5] All computer accounts should use the usernames and passwords issued by ISS at registration.

The assignment (including all code submitted) must be your **own original work**. **Selected students will be subjected to interview and/or demonstration.** The use of **third-party source code** or **automation/AI software** to directly address the task outlined in this assignment **is not permitted** and **will result in at least a zero mark** and **possible additional disciplinary action**. If in doubt, ask your module tutor/coordinator.

Note:

- **Screenshots of code will not be accepted** (this may result in a zero mark for your CA submission). All code must be in text form in your final report.
- **No use of automation, AI based code / text generation** or **reuse of 3rd party code is permitted** – all coding solutions presented for grading **must be your own original work.**
- **Submission of compressed files (e.g. zip, rar) will not be accepted** (this may result in a zero mark for your CA submission). Please use PDF submissions.
- **High URKUND (text matching scores)** may result in a zero mark for your CA submission.
- **All third party material must be referenced correctly and explicitly.** All ideas, paraphrasing of other people's words must be correctly attributed in the body of the report and in the references

Getting Started

Read and understand the *Python/Keras* requirements/installation instructions as indicated in the module notes and on the relevant websites.

Programme development[6] should use the **training**[7] 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 is not to be used for programme development**[8]). **Do not edit the datasets; they must be used as presented.**

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. In addition to the **training and validation accuracy and loss diagrams** your solution must also include your **final validation accuracy and loss & final test accuracy and loss**. All models developed should be saved in Hierarchical Data Format (HDF5) (.h5) format.

[6] Using a CPU only PC will result in long run times for these tasks. If you have access to GPU [Dettmers, AI-Benchmark] 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 style notebook environment,** but time limited) and AWS (Amazon paid Web Services). For details on using Keras with GPU on Amazon EC2 see [CS231n, Brownlee]**.**

[7] **Training Dataset**: data samples used to fit the model. **Validation Dataset**: data samples of data to evaluate model fitting on training data as we tune model hyperparameters. **Test Dataset**: data samples that are set aside until we wish to generate an unbiased evaluation of a final model fit on training data.

[8] "If you're using Colab and you feel like training your model on GPU is slow, switch to the TPU runtime and tune the "steps_per_execution" parameter in *compile().* Seeing a 5-10x speedup is pretty common". François Chollet

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

Dataset #1 (ImageNette): Imagenette from **fast.ai** is a subset of 10 easily classified classes from Imagenet (*tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute*). We will use the **160 px** ImageNette dataset as our starting point. This data has a train/validation split of 1000/500 (approx.) images, although this will be changed as part of the assignment. In this task you are to select **any 4 classes from the original ImageNette data** as your working dataset.

Layer	Number of output filters
Input Image (64x64x3)	
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]	**4**

Figure 1: Baseline CNN architecture *vgglite* - this will accept images with spatial dimensions 64x64x3.

a) The original data has a train/validation split of 1000/500 (approx.) images. This will need to be **reorganized into appropriate train/validation/test split** before you train your network models. The details of the splitting[9] is left to you, but you must fully justify any final split used in your evaluation.

[Marks: 3/25]

b) Design and develop a simple baseline convolutional neural network architecture **as per the CNN structure illustrated in Figure 1** (and as detailed below) to classify your selected ImageNette data. Illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file.

Network details:

- The model requires a colour input image of size **64x64.** Note the first layer will need to account for the input shape of the data.
- **Convolution layers** should use Xavier (glorot) uniform kernel initialization, a dilation rate of 1, strides of (height,width)= (1,1) and 'relu' activations **where appropriate**.
- **Pooling layers** will use (2x2) max pooling.

[Marks: 6/25]

[9] For example, you can use the **Keras** function ***train_test_split()*** from **sklearn** library to split your data into train, test and validation set. You can also use the ***ImageDataGenerator*** class in Keras to split the data. Another option is to use https://pypi.org/project/split-folders/ - Split folders with files (e.g. images) into train, validation and test (dataset) folders.

c) **Without adding any additional convolutional** or **fully-connected layers**, experiment with ways to improve the baseline networks performance. Each change to the *vgglite* network outlined in Figure 1 must be clearly justified in terms of the theoretical concept and supported by appropriate experiential results. Illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file.

[Marks: 4/25]

2: Dog Breed Classification using Fine-Tuning based Transfer Learning[10]

Dataset #2 (ImageWoof): In this section we will use fast.ai's ImageWoof dataset, which contains 10 dog classes [*Australian terrier, Border terrier, Samoyed, beagle, Shih-Tzu, English foxhound, Rhodesian ridgeback, dingo, golden retriever, Old English sheepdog*] based on the original ImageNet . This data has a train/validation split of 1300/50 images, although this will be changed as part of the assignment. In this task you are to select **any 3 classes from the original ImageWoof data** as your working dataset.

The original data has a train/validation split of 1300/50 images. This will need to be **reorganized into appropriate train/validation/test split** before you train your network models. The details of the splitting is left to you, but you must fully justify any final split used in your evaluation.

a) Implement an **InceptionV3** based **fine-tuning based transfer learning** **CNN architecture** to optimise the classification task. During fine-tuning, only train the top 2 Inception blocks (mixed 8 and 9: each block = 31 layers)[11]. InceptionV3 requires input image size of (299,299). Illustrate the networks performance using the required diagrams and metrics listed previously. Save your final model as a HDF5 file.

[Marks: 6/25]

b) Illustrate the performance of your network (as developed in part 2(a)) by applying it to **previously unseen images** (i.e., **not part** of the of the ImageWoof or ImageNet datasets) for all your selected classes. Comment on the performance of the network when applied to these previously unseen "in the wild" images.

[Marks: 3/25]

3: Report Structure

Overall structure and quality of the technical report (as per guidelines).

[Marks: 3/25]

[10] Pretrained on ImageNet

[11] Freeze the first (311-2*31) = 249 layers and unfreeze the rest

References

- Jeremy Howard (2019), "Fast.ai Imagenette", https://github.com/fastai/imagenette. Please refer to usage LICENSE.
- Francois Chollet (2021), Deep Learning with Python (2nd Edition)
- Keras.io
- Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd Edition)

Useful Links

- How to prevent Google Colab from disconnecting ? | by Shivam Rawat | Medium
- The Best GPUs for Deep Learning in 2020 — An In-depth Analysis (timdettmers.com)
- A Full Hardware Guide to Deep Learning — Tim Dettmers
- deep-learning-benchmark

Appendix

Assignment Report Details

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 question given in your assignment sheet** and **provide the appropriate evidence of any conclusion through experimentation / testing**.

GENERAL REQUIREMENTS ● Best practice is to use the third person passive voice ● Audience: Write the report so that your peers will understand it ● Font Use Times New Roman 12 point or similar ● There is no specific page quantity requirement.

Example headings (use as appropriate) for each section:

- **Introduction** (To the specific task. Context - briefly describe the overall aims and objectives of the task.)
- **Techniques / Rational** (Explain the approach adopted and why)
- **Concepts / Design / Pseudo Code** (Detailed outline for solution - see course notes for details on using pseudo code descriptions.)
- **Code Implementation / Procedures used** (Key aspects, example of your implementation details – reference relevant sections of your full code listing in the appendix as appropriate.)
- **Testing, Results & Analysis** (Include sample data and use tables and/or sample images / diagrams to illustrate your points as appropriate.)
- **Discussion & Conclusion** (Not a summary - refer back to the problem and give insight into the impact of your solution on addressing this task. Discuss key outcomes, eg is it what is expected clearly detailing your reasoning, issues that arose as part of your implementation)
- **References** (use a standard scientific/engineering reference style)
- **Appendix** (**Full** code listing for each section e.g. your Jupyter notebook (please comment your code appropriately). Large amount of numerical data if generated. Include all relevant material that would otherwise interrupt the flow of the main report.)

However, you decide to structure your report **you must address ALL the questions / issues raised in the assignment sheet** as these form the basis of your mark. Key to any technical report is the **precision** of your answer. As well as the engineering content, the report is 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 **students must produce their own report**.
Selected students **are subject 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 use of automation/AI based code/text generation** or **reuse of 3rd party code is permitted –** all coding solutions presented for grading **must be your own original work.**
- Reference material in the course notes when explaining concepts, e.g.: … Xavier Initialization (1) …

1. Paul F. Whelan (2025), "**Computer Vision Course Notes – EEN1001**", **Section: DL - Training,** Dublin City University, 2025.