Convolutional Networks

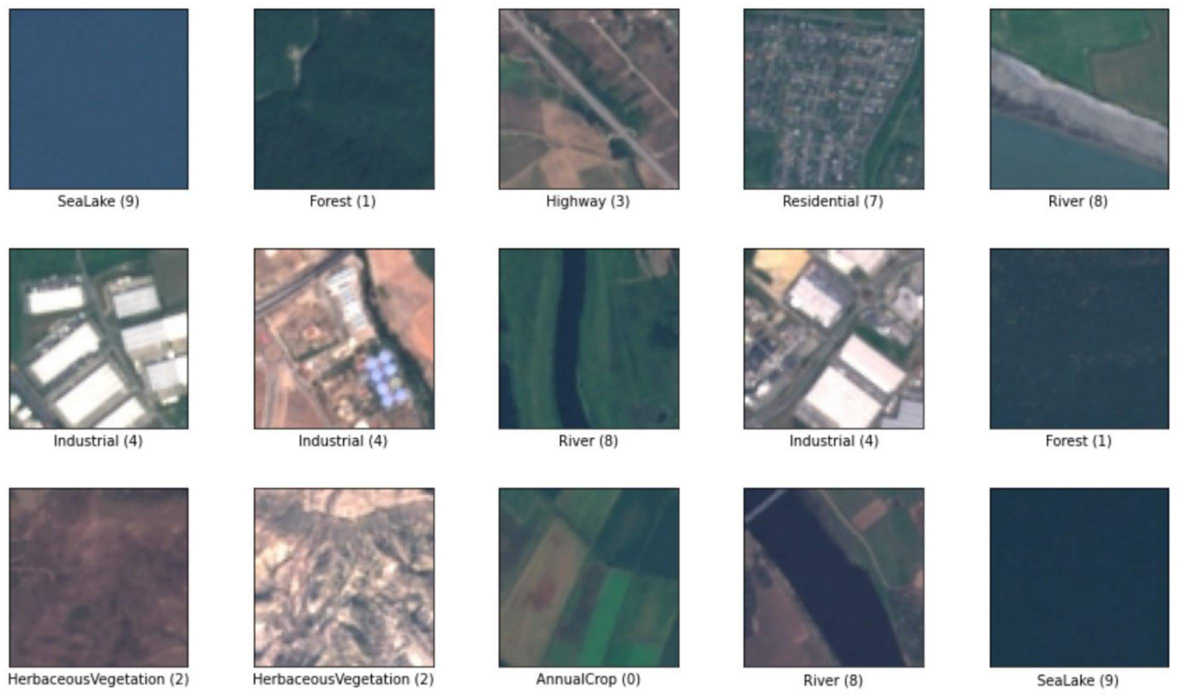
Guidelines and Submission Instructions

- Please upload your final submission (as a single zip file) to Canvas before 23:59 on **Sunday May 1st**.
- Your submitted zip file should contain your python files (.py files or notebooks for part A and B) and a report.
- Please **do not** include the dataset file in your uploaded zip file.
- It is your responsibility to make sure you upload the correct files.
- Please make sure you fully comment your code. You should clearly explain the operation of important lines of code.
- Please note that marks are awarded for code that is efficient, well structured and with minimum duplication.
- Late submissions will be penalized.
    - If you submit the assignment after the deadline but within 7 days, **a 10% late penalty will be applied**.
    - If you submit the assignment more than 7 days after the deadline but within 14 days, a **20% penalty** will be deducted.
    - A **grade of 0%** will be given to any assignment submitted more than 14 days after the assignment deadline.
- Please clearly reference any sources you use in your code or in your report. You should adhere to referencing and citation standards.
- Please note that analysis tools will be used to identify plagiarism in both your written documentation and your source code. Any plagiarism identified will be submitted to the examination infringement board, which could result in serious penalties. If you have any doubt about plagiarism please contact me via email to ask.
- I recommend that you use Google Colab for completing the implementation objectives described below. Please make sure that you select a GPU runtime instance for the VM.

Problem Specification

The objective of this assessment is to build a range of deep learning models using convolutional neural networks for the task of land cover classification using satellite images. The images are RGB images from the Sentinel-2 satellite. This is a multi-class classification problem consisting of 9 separate classes with a total of 24000 images. Please note that there is a minor amount of imbalance between the different classes but you are not expected to address this issue in the assignment.

Below you can see a selection of images from the dataset.



You will find the data file (**earth_data.zip**) in the assignment unit on Canvas. The zip file stores the data in a HDF5 file called **earth_data.h5**. Instructions for extracting the contents using Google Colab (along with a link to a Colab Notebook) are contained in Appendix A at the end of the assignment specification.

For the purposes of this assignment we will just work with training and validation data (no test data split). Therefore, you can report your accuracy on the validation data.

Please note that I have made some modifications to the dataset and performed some pre-processing on the dataset, all images are now of an equal size (64*64*3). The image data has not been normalized.

The shape of the feature training data is (19200, 64, 64, 3), while the shape of the validation data is (4800, 64, 64, 3). Therefore, there are 19200 images for training and 4800 for validation data.

The assignment consists of the following three sections:

- **Part A**: Explores the application of convolutional networks, data augmentation and ensemble techniques.

  [40 Marks]

- **Part B**: Focuses on transfer learning (specifically the use of CNNs as feature extractors and the application of fine tuning with pre-trained CNNs).

  [30 Marks]

- **Part C**: Focuses on the research component of the assignment.

  [30 Marks]

PART A: Convolutional Neural Networks: **[40 Marks]**

Part A requires you to build a range of convolutional networks for tackling the earth observational dataset problem. It also requires you to explore the impact of data augmentation and investigate an ensemble technique.

For each model you train you should plot the training and validation accuracy as well and the training and validation loss. You should also offer your interpretations of each of plots produced. Please include the graphic of the loss and accuracy as well as your interpretation in your report.

(i) ✓ Implement a baseline CNN, which contains just a single convolutional layer, single pooling layer, fully connected layer and Softmax layer.

✓ Gradually increase the number of layers in your CNN (the number of convolutional and pooling layers). You should implement at least three different basic CNN configurations (not including the baseline).

In your report show the impact on the validation and training accuracy/loss values (inclusive of the baseline case). **Compare and contrast the performance of your models in your report.**

Investigate the implementation of data augmentation techniques for the best performing model. In your report **describe the impact** (if any), of applying data augmentation on these models. How do you explain the impact of data augmentation and under what scenarios do you think it would be most useful?

Reporting after implementation

Does the selection of methods used as part of your data augmentation (such as cropping, flipping etc) have an influence on accuracy?

Research

Research and describe two more recent data augmentation techniques (not currently offered by the ImageDataGenerator in Keras). Please note there is no need to implement these. ✓

Research

A range of other solutions have been developed to facilitate building deep learning models for environments with limited data. Research and clearly describe one-shot and zero-shot learning. Given an example network and application for each category. ✓

(20 marks)

(ii) Build a CNN ensemble containing a <u>maximum</u> of 10 base learners.

In your report describe:

- The methodology you used for implementing the ensemble.
- The source of variability in your ensemble and why variability is an important factor when building an ensemble.
- Compare the performance of the ensemble with each of individual the base learners.
- Assuming you have additional time what extensions/advancements of your ensemble would you explore and what alternative ensemble methodologies might you implement (as usual this answer should be informed by research). You should clearly describe the rationale for your suggested future directions.

During the training process of each base learner you should use **<u>checkpointing</u>** so that you can capture the base learner model with the lowest validation loss.

Clearly there are many types of ensembles that you could build and explore. The following are important factors to keep in mind when building your ensemble:

- Colab has a limited amount of disk space and RAM. This can fill up very quickly depending on how you implement your ensemble and your checkpointing. A couple of points to help you minimize you RAM and disk storage space. (i) **Train a specific base model, load the best checkpointed model and use it to predict the classes for the validation data.** In other words, **do not** save each base model in a data structure and then do the predictions when you have collected all your base models (as this approach will consume significant amount of memory). Instead do the prediction for the current base model directly after you have trained the base model. (ii) When checkpointing during the model training process I suggest that you continually **save the weights to the same file** (not separate files). (iii) Also, if you using Colab you could consider saving your predictions for each base model to your **Google drive**. This means that if the training process for each model takes time then you don't have to rerun everything from scratch if you change one base learner or if the session drops.
- Please note that the **maximum number of base learners is 10**. However, you may end up using just 3 or 4 due to time constraints or computational resources. You will not be penalized for this. You may see better results if you have a larger number of base learners.
- Also, you will **not be penalized** if your ensemble doesn't outperform the individual base learners.
- There are a wide range of possible neural network ensembles that you could investigate. Appendix B provides a high level view of creating a very basic neural network ensemble (initializing a fixed architecture with new randomized weights). Successful implementation of this basic model is an acceptable approach. However, to achieve a very high grade for the ensemble you should aim to implement a more sophisticated and technically challenging ensemble (supported with evidence of research).

(20 marks)

PART B: Transfer Learning [30 Marks]

Part B focuses on transfer learning. You will be required to investigate and explore the impact of feature extraction and fine-tuning techniques.

(i) Use a pre-trained CNN model (such as VGG) as a **feature extractor** and pair its output with a secondary (standard) machine learning algorithm. For example, you could use a pretrained VGG16 network as a feature extractor and feed the extracted feature data into a logistic regression model. What is the impact on the performance?

To grade well in this question, you should explore and examine appropriate variants of the above structure. For example, one appropriate variant would be to examine a selection of different secondary machine learning algorithms in order to improve the overall level of validation accuracy (for example would a Random Forest provide any performance advantage over a logistic regression unit). In your report you should include a description of the different variants you examined, a rationale for examining each variant and its impact on model performance. You should examine three variants and also list any other variants you think might be important to consider.

(15 marks)

(ii) Explore the application of **fine tuning as a method of transfer** learning for the earth observational dataset.

Again, to grade well in this question you should include appropriate exploratory work. Your objective is to identify the best validation accuracy that you can achieve for the dataset. Therefore, you should consider the variables involved when performing fine tuning as well as additional techniques you could use to improve performance. As in the previous question, in your report include a description of the different variants you examined, a rationale for examining each variant and its impact on model performance. You should examine three variants and also list any other variants you think might be important to consider.

(15 marks)

PART C: Research [30 Marks]

Deep convolutional networks have achieved exceptional levels of accuracy when applied to image related machine learning problems.

However, convolutional networks do have some significant limitations and vulnerabilities, some of which may undermine their long-term success and viability. Geoffrey Hinton, one of the leading figures in deep learning research, has argued that despite the success of CNNs they have some have significant disadvantages that may be difficult to solve.

The objective of this section is to write a research report. Please select one of the following topics for your research report.

- Adversarial Machine Learning: The goal of adversarial techniques is to fool a machine learning model through the provision of malicious input. (Please note this is not the same thing as generative adversarial models).

- Self-supervised Learning. A category of machine learning that can learn from unlabelled data.

Your research report should not exceed 3 pages (guideline for max word count 1500 words). Please include any references. To grade well for this question you should demonstrate that you have researched the topic from a range of sources, your explanation should convey an exemplary depth and understanding of your selected topic, expressed in your own words, along with a good grasp of the underpinning technical knowledge.

Appendix A: Accessing Training and Validation for the Earth Observation Dataset (Using Google Colab).

The following instructions describe an efficient way of accessing the data using Colab. Please note you can follow the same process if working with the Google Cloud deep learning VM (without mounting your Google Drive of course).

1. The dataset is stored in a zip file called earth_data.zip, which you can find on Canvas in the assignment unit.

2. Copy the compressed data file (earth_data.zip) to a folder in your Google Drive (wait for the upload to complete). For the purposes of this example I have placed the zip in a Google Drive folder called *datasets*.

3. Once the compressed data file has been transferred to the *datasets* folder in your Google Drive open a new Google Colab notebook. You can find the full code available in the following Colab Notebook. You can easily make a copy of the code by going to File -> Save a copy in Drive

   The following is a summary of the steps needed to access the data:

   a. Step 1 requires you to mount your Google Drive. In your Colab notebook execute the following code. This will ask you to enter follow a link and enter an authorisation code. Once complete you should see the message "Mounted at /content/gdrive"

   ```
   from google.colab import drive

   drive.mount('/content/gdrive')
   ```

   b. Next extract the contents from the file earth_data.zip by entering the following in a Colab notebook. Remember my zip file is stored on my Goole Drive in the datasets folder.

   ```
   !unzip "/content/gdrive/My Drive/datasets/earth_data.zip" -d "./"
   ```

c. This should extract the file **earth_data.h5** into your current working directory. To confirm run the following in a Colab cell and you should see the file earth_data.h5.

```
!ls
```

d. You can now use the code below to open the HDf5 file and extract the contents and store in a NumPy array. Upon executing this code you should see the size of each NumPy array printed as follows:

(19200, 64, 64, 3) (19200,)
(4800, 64, 64, 3) (4800,)

```
import numpy as np
import h5py


def loadDataH5():


    with h5py.File('earth_data.h5','r') as hf:
        trainX = np.array(hf.get('trainX'))
        trainY = np.array(hf.get('trainY'))
        valX = np.array(hf.get('valX'))
        valY = np.array(hf.get('valY'))
        print (trainX.shape,trainY.shape)
        print (valX.shape,valY.shape)
    return trainX, trainY, valX, valY

trainX, trainY, valX, valY = loadDataH5()
```

Appendix B: High level Guide to Create a Basic Neural Network Ensemble.

1. Train a neural network model for a fixed number of epochs (in this simple case we just use the same model architecture - each time you create a new instance of the same model it will initialize the model with different random weights). For example, you could use the ShallowVGGNet architecture that we looked at in the lectures (please note this is a simple model and not the fully training VGG model from imagenet). Each time you create a new instance of this model it will be initialized with different randomized weights (this is as basic an NN ensemble as possible and will likely not be very effective).

2. After the model is trained push your validation data through the model and store the results. (Use the predict function to input all validation images into the model and retrieve all associated probabilities. Remember if you have 100 input images and 10 classes in your classification problem then a single neural network model will output an array 100*10).

3. Repeat steps 1 and 2 for each base learner in your ensemble.

4. Average the probabilities obtained for each of the base learner models. For example, if you have 4 models, all the probabilities for the first image should be averaged across the four models (so that you end up with 10 probabilities for the first image, as opposed to 40 (4*10)).

5. Finally get the class predicted by the ensemble for each image by selecting the index with the highest probability.