Dr Muhammad Rana and Dr Anjan Dutta
TAs: Aaron, Ahmed, Anindya, Jiantao, Sergio, Silpa, Srinivasa
Institute for People-Centred AI
University of Surrey

Spring Term 2024

EEEM068: Applied Machine Learning
Project: Convolutional Neural Networks for Scene Recognition
Project TA Lead: Anindya Mondal (a.mondal@surrey.ac.uk)

- Submission deadline: **Monday, 13 May 2024**

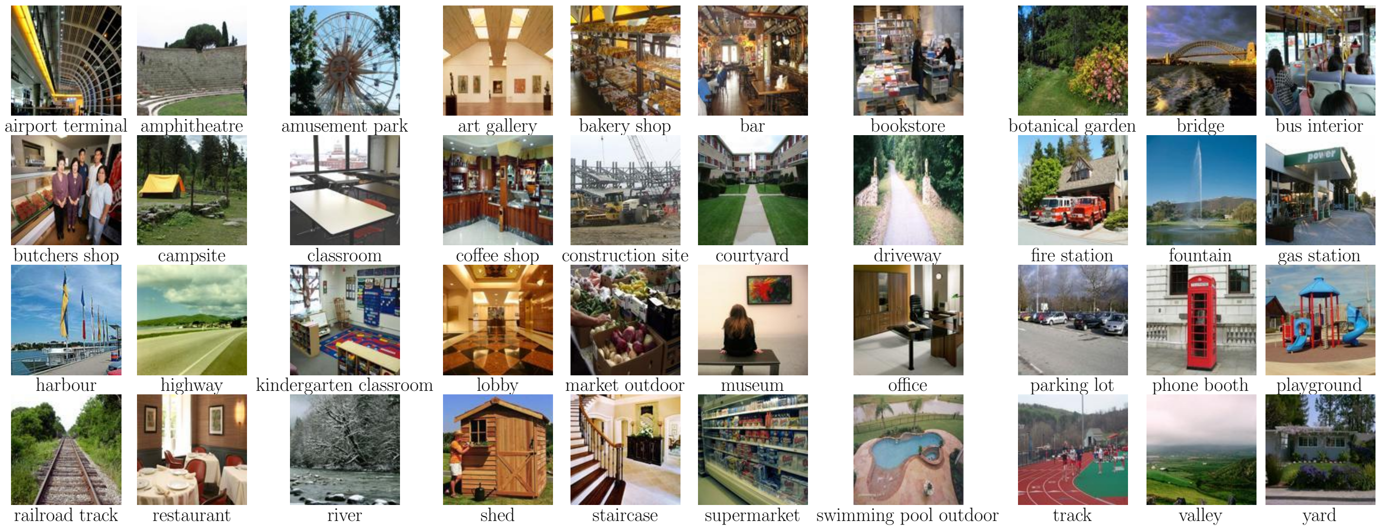

Figure 1: Instances of 40 classes from the `Places2_simp` scene dataset.

Project specification

Overview

This project will require you to apply the learnt knowledge that you are supposed to acquire in the tutorials of the EEEM068: Applied Machine Learning modules. This project is one of the options which you can opt in for undertaking summative assessment carrying 25% of the total module marks. The aim of this project is for you to use Convolutional Neural Networks (CNNs) and Transfer Learning for scene recognition.

Submission

You should submit a zip file containing the following:

- your code;
- a 4-pages report explaining your code, visualising your results you obtained and discussing your observations. If you want you can include additional pages only with visualisations (this is optional and you won't lose any marks if you do not include additional pages). However, the main text of your report should fit in the first 4 pages and the additional pages (if any) should only include visualisations with short captions.

Please note that for all the visualisations and tables that you include in your report, it is important to include a reference in the main text (typically using a Figure or Table number).

Background

For the main background, please see the lectures slides that are related to CNNs. You have to train a CNN that will learn to recognise 40 different types of scenes. For that you will use data augmentation and transfer learning, based on the popular ResNet-34 model (He *et al.*, 2015)[1], which was discussed in the lectures.

You will have to implement a similar procedure as of Workshop 4, where you implemented CNN for image classification and for transfer learning. You can use the exampled in Workshop 4 as your starting point. However, please note that you would need to modify and expand it significantly, so that it fits your needs.

[1]Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition.* In arXiv preprint arXiv:1512.03385, 2015.

Image dataset and ResNet-34 network

You are provided with an image dataset ("`Places2_simp`"), which includes images of 40 different categories of scenes/-places, such as "airport terminal", "bar", "museum" and "playground" etc as shown in Figure 1. This is a simplified version of the "`Places2`" database[2]. The dataset you will use is available through SurreyLearn, as a compressed zip file: `Places2_simp.zip`. As in the case of many other datasets, every subfolder of the "`Places2_simp`" dataset corresponds to a different category, therefore the folder names serve as the labels of the corresponding classes. All images are colour images with 128×128 pixels. Every category includes 1,000 images, therefore there are total 40,000 images in the "`Places2_simp`" dataset.

You will have to use ResNet-34 model available within the `torchvision` package. If this package is properly installed, then the ResNet-34 model can be loaded by calling `torchvision.models.resnet34()` command.

Train the CNN

- Similar to Workshop 4, load the "`Places2_simp`" dataset by implementing an appropriate dataset and dataloader.
- Since most of the standard CNN models (including ResNet-34) are trained on images with size 224 × 224 and given the fact the images of the "`Places2_simp`" dataset are of size 128 × 128, it is necessary to resize to resize the images from 128 × 128 to 224 × 224. Incorporate the necessary transformation functions which can resize the images as above.
- Randomly split the dataset into a training and validation set, using the typical 80% (training) versus 20% (validation) ratio.
- Use data augmentation of the training set, by including random X and Y translations, random rotations (for a range around -20 to 20 degrees, since very large rotations would not make sense), as well as random scalings.
- Create a CNN model by removing the last layers of ResNet-34 and adding new layers that are consistent with the classification problem that you need to solve. To learn faster in the new layers than the transferred layers from ResNet-34, try different and increased learning rate. Have a look on this discussion thread to know how to set different learning rate for a specific layer.
- Train the CNN model. Try to tune the hyperparameters, such as batch size, number of epochs, learning rate etc. Try to plot the training progress using the `tensorboard`.

Test the CNN

- Calculate the top-1 and top-5 classification accuracies and the confusion matrix for the validation set. Plot the latter in such a way that the confusing cases could be understood effectively.
- To obtain more intuition on the top-5 accuracy rate measure, randomly choose some of the correctly and wrongly classified images from the validation set and display the top-5 scores together with the corresponding actual and predicted class names. This will help you make interesting observations about how the CNN has learnt to classify the scenes.
- Create your own test set of scene images, using photographs from Guildford or the wider Surrey or London area. Include at least 5 to 10 images per category. You can use an image search engine on the Internet (e.g. Google) by typing Guildford or London and each category's name (or similar keywords), for example "guildford butchers shop". If you want, you can also use photos taken by you. To help you, we are providing a zip file with some images that we found online (you can find it on the SurreyLearn page): `testset.zip`. There is one folder for each category (even for categories for which we haven't included any image, in which case the folder is empty). You can start from this test set and expand it accordingly. Make sure that you place every image in the correct folder.
- As in the case of the validation set, estimate the accuracy rates (standard and top-5 rates) and confusion matrix for your test set. Also randomly choose test images and display again the top-5 scores and and corresponding class names.
- Try different values of hyperparameters to improve the training behaviour and the accuracy measures. Observe how the choice of hyperparameters affects the results.

Please note that on the validation set, the accuracy rate should be at least (if not much higher than) 45%, but the top-5 accuracy rate should be at least (if not much higher than) 75%.

[2]`http://places2.csail.mit.edu/explore.html`

Discuss your Observations

- In your report, you should discuss your observations with reference to the fine-tuning of hyper-parameters, training behaviour, accuracy measures, confusion matrices and the visualisations of the CNN that you have produced. Whenever appropriate, please refer to the corresponding figures / tables to make your observations more concrete.

Extra credit

Extra credit will be awarded if one could potentially perform additional tasks related to the main scene recognition task. These additional tasks might include but not limited to visualisation, interface design etc.

MARKING CRITERIA

The project will be assessed giving the 50% weight (12.5 marks) to technical report and 30% weight (7.5 marks) to functionality and 20% weight (5 marks) to code quality according to the following criteria:

REPORT QUALITY [50 marks]

Whether the results are well presented and discussed.
In particular:

- Is the report well written and clear?
- Is the report well structured?
- Are the figures clear and well explained?
- Does the report provide a clear explanation of what has been done to solve the problem?
- Is there a sufficient discussion regarding observations on the produced results?

The distribution of the marks within the report are as follows:

- Abstract: 10 marks,
- Introduction: 10 marks,
- Literature (minimum 5 papers): 15 marks,
- Methodology: 25 marks,
- Experiments: 30 marks,
- Conclusion and future work: 10 marks

FUNCTIONALITY [30 marks]

Whether the submitted program performs as specified.
In particular did the code implement all the steps specified in the previous sections?

CODE QUALITY [20 marks]

Quality and efficiency of the coding, including appropriate use of documentation.
In particular:

- Is the code efficient?
- Is the code extensible and maintainable?
- Is the code clear and well documented?