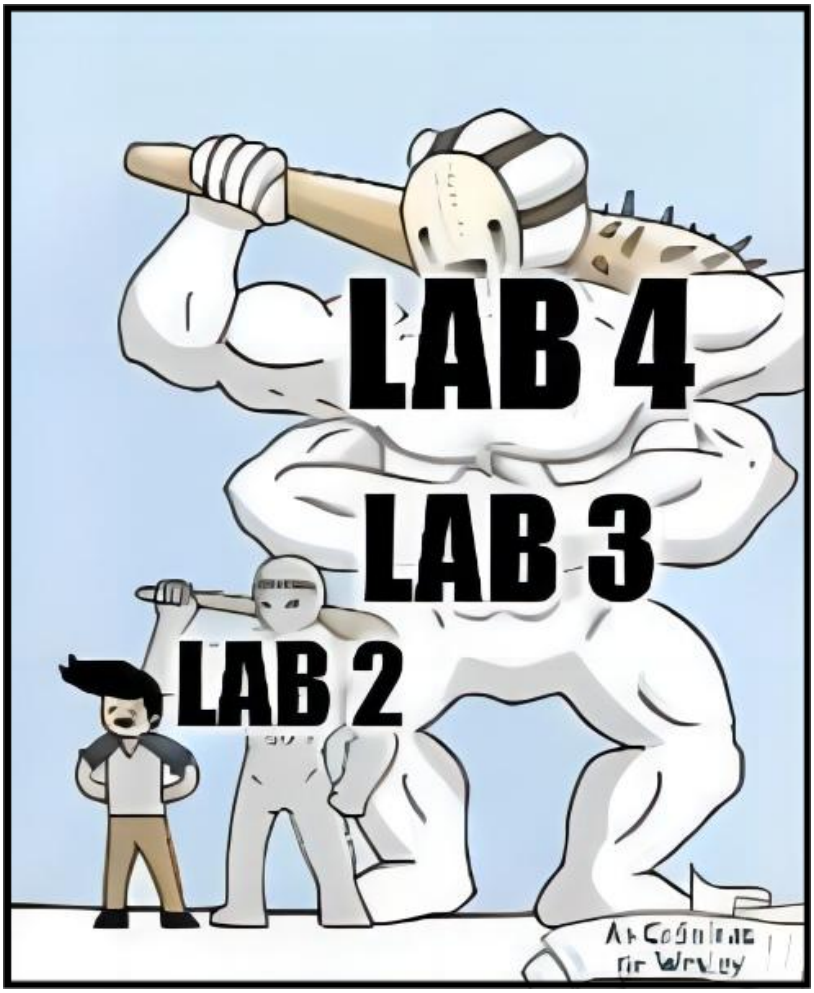
ECE4179 - Lab 4
Convolutional Neural Networks (CNN)



Contents

Academic integrity and the Use of Generative AI

Academic integrity

Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment and interviews with the plagiarism officers at Monash University.

Lab Instructions and the Use of Generative AI

- You should use Matplotlib to display images and any intermediate results.
- You may use a generative AI tool or coding assistance. We recommend understanding the concepts and coding as much as possible, as you may be performing hand calculations and write code by yourself during the final exam. If you use generative AI tools, please indicate the prompts you used and explain in detail any changes you made to modify the code to complete the assignment.

Late submission policy and grading guidelines

Late submissions

Late lab submission will incur a penalty of 5% of available marks for each day late. That is, with one day delay, we will (sadly) deduct 5% of this lab from the mark you get, *i.e.* if you get 75% and submitted one day late, the final marks you will recieve is 70%. Labs submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness). You can do this by applying through Monash University's special consideration application website.

Lab Grading

This lab is worth 5/100 (5 out of 100) marks of the entire unit. Lab attendance is not compulsory. The lab sessions are there for you to get help.

For the .ipynb submission, there are a number of tasks. Each task will have coding components and a discussion component. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

You can complete this lab with a partner. You need to register on Moodle under week 5. Only one of you need to submit the lab.

There are 3 tasks in this lab. The tasks are worth 30%, 30% and 40% respectively. Both the coding and discussion tasks are assessed, so please complete all of it! You can enter the discussion answers directly into the notebook.

This lab is about understanding and applying PyTorch-Lightning to simple Computer Vision tasks using Convolutional Neural Networks. By the end of the lab, you will have developed simple CNNs for multi-class classification tasks. You will also have explored and optimised your own model. The following are the three tasks that you should complete.

- Task 1: Perform multi-class classification using a CNN. This task will explore PyTorch data-loaders and other functionalities.
- Task 2: Analyse and understand the results of the CNN and how it performs on STL-10 test data. This will involve applying old and new analysis tools.
- Task 3: Design and train a CNN on the FER-4 dataset and analyse the results. You will be exploring your own CNN network architecture!

The learning outcomes for this lab are:

- Furthering your knowledge of PyTorch and PyTorch-Lightning
- Selecting data augmentation techniques to improve robustness of the model
- Implementing CNNs from scratch for image classification tasks
- Loading model weights to reuse a trained model
- Applying analysis tools to understand CNNs
- Exploring and optimising model performance

Similar to lab 3, most of the lab does not require you to use the Numpy library. You will use PyTorch/Lightning in-built methods to create your tensors instead of Numpy. Follow the PyTorch/-Lightning videos for more information.

Introduction.

The prominence of deep learning models erupted with the ability to train deep neural networks more efficiently. The ability to split data up to different GPUs and run simultaneous training has propelled the research of neural networks forward. With CNNs, we are not required to handcraft the kernels to detect features (such as edges), but rather allow these kernels to be learnt automatically. Being able to train and test CNNs will be your entrance to hands-on deep learning, and you will re-use a lot of past concepts in order to complete this lab.

The submission for lab 4 is due 20th of September (Friday) 4:30 PM AEST.

It is recommended that you go through the following videos/documents prior to attending your lab 4 sessions:

1. Begin by reading through this document. This document contains all the relevant information for lab 4.
2. Watch the introductory video for this lab.
3. Follow the lecture series by Luke Ditria on CNNs with PyTorch/PyTorch-Lightning.
4. You may choose to use Google Colab for this lab so that you can utilise the free GPU provided. [1]

In the lab and in your own time, you will be completing the lab 4 notebook by going through this document and the provided notebook.

Lab Warning

It is **highly** recommended that you start on this as soon as you can. If you leave it to the second week of the lab, you may find yourself not having time to ask questions[a]. **This lab also takes a while to complete running the actual code! You want to ensure your models are trained ahead of time, so you can add in any necessary discussions and have all the outputs ready for submission by the deadline.**

[a] With all the code completed, the two notebooks can take anything between approx. 30 min - 1 hour to run on Colab, so ensure to start early! Note: optimisation in task 3 may take awhile (ie. a few hours) to run.

[1] There is a video in Week 0's Moodle page detailing on how to use Google Colab. **Remember, Google Colab is NOT compulsory to use, but may help if you do not have your own GPU**. If you do use Google Colab, ensure the data is stored on Google Drive and accessed via mounting GDrive to your Google Colab.

Task 1: Design a CNN and train to classify STL-10 images [30%]

In this task, you will write the code to train a CNN model to perform multi-class classification. You will be developing a CNN model for the STL10 dataset. The dataset represents 10 object classes, namely {'airplane', 'bird', 'car', 'cat', 'deer', 'dog', 'horse', 'monkey', 'ship', 'truck'} in that order. The dataset has 13,000 color images of size 96×96 . We divide 5,000, 1,000, and 7,000 images for training, validation and testing, respectively.

Figure 1: Example images of the STL10 dataset

- 1.1 Create Pytorch datasets and dataloaders for STL-10
- 1.2 Visualize the dataset
- 1.3 Design the CNN and Lightning Modules
- 1.4 Train and Evaluate the CNN
- 1.5 Visualize training and validation loss/accuracy

This task follows similar steps to how you have applied the end-to-end training process for MLPs. The dataset will be downloaded from the *torchvision* library [2] and you will design a CNN

[2]For your own interest, you can see what other datasets are available as well. Most of these are common datasets that have been used to benchmark model performances in the past

for this image classification task. As usual, you will train and evaluate the model performance by analysing the loss and accuracy curves. For this task, we have provided you with the network structure and hyper parameters.[3]

If you have setup your *ModelCheckpoint* to save the best model during the training process, you can reuse the same model weights so that you do not need to re-train the model after restarting your python kernel. To do this, you can refer to the PyTorch-Lightning documentation for more information.

[3] Don't worry, you will have to explore your own network in the second notebook!

Task 2: Analyse the results on the STL-10 test images [30%]

In this section, you will be implementing additional ways to visualize performance of the model and how features have been extracted in a CNN architecture.

- 2.1 Visualize predictions
- 2.2 Visualize Top Classified/Misclassified
- 2.3 Confusion Matrix
- 2.4 Visualize Feature maps
- 2.5 Visualize Saliency maps

The first two subtasks requires you to visualzie the predictions made by the model, and to see which images have been classified with a high degree of certainty, and which images have been misclassified (but with a high degree of certainty that the model thought was correct). The next task will be to visualize the confusion matrix similar to lab 3.

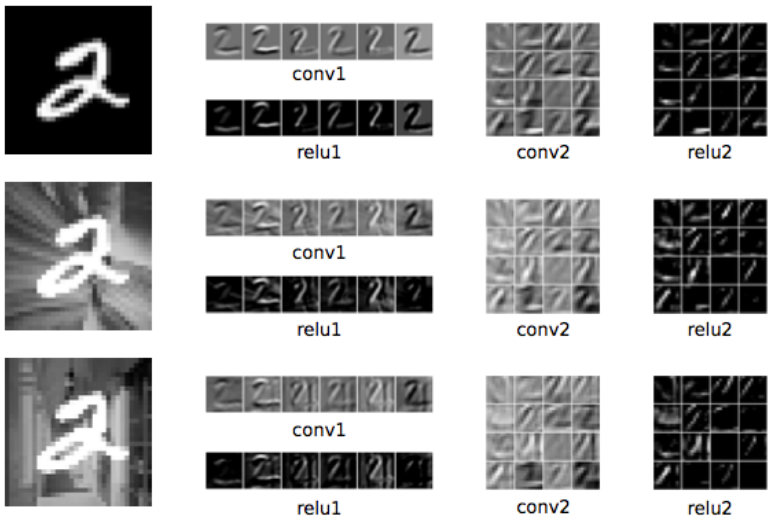


Figure 2: Example of intermediate feature maps for the MNIST dataset

Figure 3: Example of saliency maps at progressive layers within a CNN

The final two subtasks are new methods in visualizing the intermediate features within a CNN architecture. To visualize the feature map, you will forward pass a single image through the network, and extract the intermediate values after each pooling layer. Remember, the max pooling layers extracts the highest weighted input features (hence most prominent features).

To find the saliency maps, you will analyze the gradients of an image to identify regions with distinct visual characteristics, such as contrast, color, and texture, which are then used to prioritize and highlight the most attention-grabbing parts of the image. [4]

[4]Saliency maps find applications in various fields, including object detection, image segmentation, and even in improving the performance of deep learning models by focusing on relevant image regions during training and inference.

Task 3: Design and train a CNN on the FER-4 dataset and Analyse Results [40%]

In this task, you will be exploring your own network architecture and optimizing your CNN. We have provided places where parts of your code will go but you are welcome to add in additional analysis. You will be using the FER (Facial Expression Recognition) dataset. This dataset is a widely used collection of facial images for emotion recognition tasks. It typically consists of grayscale images with dimensions of 48x48 pixels, and the standard FER dataset contains around 35,000 images of human faces expressing various emotions, including happiness, sadness, anger, surprise, fear, disgust, and neutral expressions. We will only use four of these classes - namely (0) happy, (1) neural, (2) sad, and (3) surprise.

Figure 4: FER-4 dataset

The final deliverables for this task are the following:

1. Accuracy of more than 65% on the test dataset.
2. Convergence of train and validation loss/accuracy curves.
3. Confusion Matrix
4. Visualization of top 5 misclassified images for each of the four classes.
5. Visualization of the saliency map of a correctly predicted 'happy' image.
6. Try your own data! - Capture an image of yourself from the webcam/phone and test.

We have provided you with some guidelines within the notebook for designing your final network for this task.

Hopefully, you have enjoyed this lab as much as we did when we made it! This lab aimed to give you insight into training a simple CNN via the PyTorch-Lightning framework. Additional robustness and analysis techniques were also introduced in this lab and we hope that has been an interesting learning experience!