Assignment 07
Convolutional Neural Networks (CNNs)

EECE-6822 Machine Learning

Cris Ababei
Electrical and Computer Engr., Marquette University

1. Objective

The objectives of this activity include: (1) run several code examples that work with convolutional neural networks for image classification; implementations both in Keras/TF2 and Pytorch (2) use transfer learning to achieve short training time and high performance, (3) investigate different optimizers for CNNs.

2. Prerequisite Readings

Murphy
- Ch. 14: neural networks for images

Geron
- Ch.14: deep computer vision using CNNs

Raschka
- Ch.14: classifying images with deep CNNs

3. Code Examples

Example 1: CNNs in Keras

This is the example code from Ch.14 from Aurelian Geron's book.
[*B3-Geron] Aurelien Geron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, O'Reilly, 2022.
Open in your google colab and go through all the code; observe all details. Of course, you should read first the chapter itself from the book, before going through the code.
https://github.com/ageron/handson-ml3/blob/main/14_deep_computer_vision_with_cnns.ipynb

This chapter explores where CNNs came from, what their building blocks look like, and how to implement them using **Keras**. Then some of the best CNN architectures are discussed. Various visual tasks are discussed, including object detection (classifying multiple objects in an image and placing bounding boxes around them) and semantic segmentation (classifying each pixel according to the class of the object it belongs to).

Example 2: CNNs in Pytorch

This code example is from Ch.14 from:
[*B3-Raschka] Sebastian Raschka, Yuxi Liu, and Vahid Mirjalili, Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python, Packt Publishing, 2022.
The source code is located at the GitHub repository:
https://github.com/rasbt/machine-learning-book/tree/main/ch14
To execute the code, I found the easiest to actually run directly the Python code corresponding to this notebook in Anaconda Spyder.
At this time, you should run the notebook or run directly the Python code. You should also first read the chapter from the book itself, before running the code! Essentially, in this code example we see a discussion of basic building blocks of CNNs, using a bottom-up approach. Then, a discussion of how to implement CNNs in **PyTorch**. The following topics are covered:

- Convolution operations in one and two dimensions
- The building blocks of CNN architectures
- Implementing deep CNNs in PyTorch
- Data augmentation techniques for improving the generalization performance
- Implementing a facial CNN classifier for recognizing if someone is smiling or not

Example 3: CNNs in TF2 and Pytorch

These are examples suggested by K. Murphy on his github repository for the textbook:
https://github.com/probml/pyprobml/tree/master/notebooks/book1/14
Scroll to the bottom of the above page to find several notebook examples with **TF2 and Pytorch** implementation examples! Do at least two such notebook examples from each category.

Example 4: mlm tutorials and code examples on NNs

In this part, we will look at several tutorials from **machinelearningmastery (mlm)**. Read the following tutorials and run the code where applicable:

--Crash Course in Convolutional Neural Networks for Machine Learning

https://machinelearningmastery.com/crash-course-convolutional-neural-networks/

-- Stanford Convolutional Neural Networks for Visual Recognition Course (Review)

https://machinelearningmastery.com/stanford-convolutional-neural-networks-for-visual-recognition-course-review/

--Deep Learning Models for Human Activity Recognition

https://machinelearningmastery.com/deep-learning-models-for-human-activity-recognition/

--1D Convolutional Neural Network Models for Human Activity Recognition

https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/

--A Gentle Introduction to Padding and Stride for Convolutional Neural Networks

https://machinelearningmastery.com/padding-and-stride-for-convolutional-neural-networks/

--How to Visualize Filters and Feature Maps in Convolutional Neural Networks

https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

--Handwritten Digit Recognition Using Convolutional Neural Networks in Python with Keras

https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/

--Building a Convolutional Neural Network in PyTorch

https://machinelearningmastery.com/building-a-convolutional-neural-network-in-pytorch/

4. Assignment

Part 1: You must redo the example shown in class, which focused on detecting the presence of a cat or dog in an image.

Cat_Dog_Detection_using_Transfer_Learning.ipynb

That notebook was a **shorter** version of the original notebook from Google, located at the link below.

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/transfer_learning.ipynb

The shorter version provided in the .zip archive for this lecture did not include the second way to customize a pretrained model:

> *"2. Fine-Tuning: Unfreeze a few of the top layers of a frozen model base and jointly train both the newly-added classifier layers and the last layers of the base model. This allows us to "fine-tune" the higher-order feature representations in the base model in order to make them more relevant for the specific task."*

In this assignment, you are asked to change:

Cat_Dog_Detection_using_Transfer_Learning.ipynb

to include also the "*Fine-Tuning*" part. Read the original tutorial to see how to use MobileNet2 and to copy the text and code boxes for "*Fine-Tuning*" part.

Record the new accuracy vs. epoch and cross_entropy vs. epoch plots, as well as test accuracy. You will have to include these together with the initial ones (from before adding the changes) in your report for comparison purposes.

Part 2: In this part, using the updated notebook from Part 1, you must rerun the example to investigate the impact of using three other optimizers.
First, read Ch.11, section "Faster Optimizers" from A. Geron's book. Choose three other optimizers (different from Adam, currently used in our example code) and rerun the example to collect their final train and test accuracies. Create a Table or a Plot to show train and test accuracies vs. optimizer. The main goal of this exercise is to explore other optimizers as an attempt to search for a training setup that will improve the test accuracy. As such, you are free to implement other tricks towards that goal, in addition to changing the Adam optimizer – you get extra points if you can push performance higher.

5. Deliverables

You must write (typed) a report and upload it as a PDF file on D2L. The report should be named "**hw7_report_LastName.pdf**". You should also create a .zip archive with all your code and implementations of all parts of the assignment. Upload also this archive .zip file with the name "**hw7_implementation_code_LastName.zip**" to D2L. Hence, your D2L should contain two items: the report and the .zip file. **Do not include the report inside the .zip and upload only the .zip. They should be two separate items!**

The report should include the following sections and subsections. Make sure section titles are in bold font and pages are numbered.

1) **Title + course info + your name**
2) **Summary.** Describe in one paragraph what the objective of the assignment is.
3) **Description of Experiments and Discussion.** Describe the experiments you did. All tables and figures should be numbered and should have captions. All plots in all figures should have axes labels and titles. Present a meaningful discussion with the interpretation of the results you obtained. Explain if you expected the results or not; discuss the intuition behind it.
4) **Conclusion.** Present your conclusions; highlight what are your main takeaways that you learned from this assignment. Describe what issues you encountered and how you solved them.

5) **References.** Include all references that you used, as a numbered list. Cite them in the report itself; do not just list them! If your report has References that are not cited in the report, points will be deducted!