SYDE 671 Advanced Image Processing

Zahra Gharaee University of Waterloo `Fall 2022`

Assignment 4 - Coding Part

Instructions

- You need to perform modifications to 3 files:
  (a) *preprocess.py*
  (b) *vgg_model.py*
  (c) *your_model.py*
  (d) possibly *hyperparameters.py*

  The locations in these files that need editing are marked by TODO comments.

- Your submission should be a zip file that includes:
  - the 6 .py source code files.
  - your best performing weights for Your_Model
  - Do not include weights for the VGGModel in your submission.
  - a PDF or Jupyter Notebook writeup report discussing your findings.

Your report should include the following information

- Describe your process and algorithm, show your results, describe any extra credit, and share any other information you feel is relevant.
- Report your classification performance for each step in Task 1, and for Task 2.
- Include graphs of your loss function over time during training. You can use Tensorboard to view these graphs.
- Include screenshots of the model summaries. Make sure to capture both the architecture and the number of parameters used (trainable and non-trainable).

Overview

We will design and train convolutional neural networks (CNNs) for scene recognition using TensorFlow. Remember scene recognition with bag of words (assignment 3), which achieved 50 to 70% accuracy on 15-way scene classification? We're going to complete the same task on the 15 scenes database with deep learning and obtain a higher accuracy.
Two very common approaches to recognition problems in computer vision today are: either train a deep network from scratch—if you have enough data—or fine tune a pre-trained network.

Task1 Design a CNN architecture with less than 15 million parameters, and train it on a small dataset of 1,500 training examples. This isn't really enough data, so we will use:

- Standardization (a type of normalization)
- Data augmentation
- Regularization via dropout

You will be implementing standardization and data augmentation in *preprocess.py*. Regularization via dropout layers will be in *YourModel.py*. It's a good idea to have an (at least) preliminary preprocessing routine set up before building your model that you can fine-tune later. You can see some of the results of your preprocessing function visualized after/during training under the ”IMAGES” tab in Tensorboard.
Your goal will be to achieve at least 65% **test accuracy** (for any training epoch) on the 15 scene database. No points will be awarded for architectures with more than 15 million parameters.

Task 2 Write and train a classification head for the VGG-F pre-trained CNN to recognize scenes, where the CNN was pre-trained on ImageNet. With the weights of the pre-trained network frozen, there should be no more than 15 million trainable parameters in this model.
Your goal will be to achieve at least 85% test accuracy (for any training epoch) on the 15 scene database. No points will be awarded for architectures with more than 15 million trainable parameters.
Each time the program is run, a summary of the network will be printed, including the number of trainable and non-trainable parameters. Make sure to pay attention to this so that you don't exceed the limit enforced on each network.

Extra Credit up to 10 points total!!

Be sure to analyze in your reports whether your extra credit has improved classification accuracy. Each item is ”up to” some amount of points because trivial implementations may not be worthy of full extra credit. Some ideas:

- (up to 10 pts): Gather additional scene training data (e.g., from the SUN database or the Places database) and train a network from scratch. Report performance on those datasets and then use the learned networks for the 15 scene database with fine tuning.
- (up to 10 pts): 1 point for every percent accuracy over 70% when training from scratch on the 15 scene database (good luck with this one).
- (up to 10 pts): 1 point for every percent accuracy over 90% when fine-tuning from VGG-F. You don't get extra credit for switching to another network (like one trained on the Places database). The challenge here is to adapt a very big network to a relatively small training set.

Setting up Google Colab

If you have good hardware you should be able to solve the assignment on your local machine; if you don't have access to good hardware, there are many tutorials online to setup google colab

and use it for training. Its free for a 12 hours session at a time, you should be able to save checkpoints and resume training in a new session.
This is a tutorial that may help you get started with google Colab.

Credit

Project description and code written by Isa Milefchik, Aaron Gokaslan, James Tompkin, and James Hays. Originally solely by James Hays, but translated to TensorFlow from MatConvNet by Aaron, then translated to Tensorflow 2.0 by Isa. Colab guide by Ruizhao Zhu, Zhoutao Lu and Jiawei Zhang.