CS/ECE 528: Embedded Systems and Machine Learning
Fall 2024

Homework Lab 2: Network Architecture Search, Hyperparameter Tuning, and Transfer Learning

Assigned: 12 September 2024
Due: 19 September 2024

Instructions:

- Submit your solutions via Canvas.
- Submissions should include your jupyter notebooks in a zip file, with notebooks names q1.ipynb, q2.ipynb, etc. in a single folder. You can include comments in your notebooks to explain your design choices.
- **"Save and checkpoint" your notebook after running your notebook, so that cell outputs are preserved.** If you are using Colab, make sure to 'Save' your notebook after running it, before downloading it and submitting.

Q1. (NAS; 75 points) In this question you will use Autokeras for neural architecture search (NAS). Autokeras uses Bayesian optimization to perform NAS, to find the best models for a given problem. See the paper here: https://arxiv.org/pdf/1806.10282.pdf and access tutorials and guidelines at https://autokeras.com/.

(a) [25 points] First, hand-design a model for the Fashion MNIST dataset classification problem. Fashion MNIST can be downloaded and used from the keras library, as shown here: https://keras.io/datasets/. More information on the dataset can be found here: https://github.com/zalandoresearch/fashion-mnist. Aim to get as high an accuracy on the test data as you can (at least 88%). Include your Jupyter notebook (call it q1a.ipyb).

(b) [50 points] Now use Autokeras to find the best model for the Fashion MNIST dataset classification problem. Use the Image Classifier variant of AutoKeras for the exploration with max_trials=2 (this will take at least 2 hours to execute on Colab with a GPU; if you are feeling adventurous you can increase the trials to 3 or more, for more comprehensive, exploration which is also more time consuming). Include your Jupyter notebook with the autokeras code for exploration (call it q1b.ipynb). Compare the accuracy of your model from Q1(a) with that of the one obtained from Autokeras. Also include a word/pdf file (call it q1b.docx/pdf) describing your best hand-designed model, the Autokeras derived model, images of the two models (obtained using the keras function plot_model()), along with the model accuracy on the test data. What can you say about the capabilities of Autokeras, based on your accuracy result comparison?

Note: The Autokeras exploration trials can take a long time, so you may want to save/checkpoint your model periodically, in case the simulation gets interrupted. Examples of how to checkpoint your Keras code can be found on the TensorFlow website: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/ModelCheckpoint Loading the checkpointed weights from a checkpoint file is as easy as calling: model.load_weights(<<specify the checkpoint file path/location here>>)

Q2. (Hyperparameter tuning; 150 points) The TensorFlow "hparams" plugin allows you to explore hyperparameters (such as the optimizer to use, dropout rates, number of neurons etc) for your model, to find the best ones. Unlike Autokeras which helps you find the best model, hparams helps you fine tune an existing model. Take a look at the example here: https://www.tensorflow.org/tensorboard/hyperparameter_tuning_with_hparams.

(a) [50 points] Starting with any of the models from Q1 (preferably the best model), use hparams for exploring the following optimizers: 'adam', 'nadam', 'rmsprop' and 'ftrl'. All optimizers supported in TensorFlow are outlined here: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers. Plot your accuracy results with a bar plot (one bar for each optimizer accuracy result). Include your Jupyter notebook (call it q2a.ipynb) and a word/pdf file with the plots (call it q2a.docx/pdf)

(b) [50 points] Starting with your best model from Q2(a), now use hparams to explore the following three activation functions for at least 2 (or the maximum number of layers with activation functions if it is less than 2) of your dense and/or convolutional layers: 'relu', 'leakyrelu', and 'gelu'. Ignore the last layer of your model with the softmax activation function. Describe the model which gives the best performance in a table with details of the type/act-function of each layer. Note that for the 2 selected layers in your model with activations functions, you need to explore $3^2 = 9$ possible combinations of the activation functions, to find the best configuration. Include your Jupyter notebook (call it q2b.ipynb) and a word/pdf file with the requested information (call it q2b.docx/pdf)

(c) [50 points] Starting with your best model from Q2(b), now use hparams for exploring the number of units (in dense layers) or filters (in CONV layers) in at least 2 of your dense and convolutional layers (or the maximum number of layers if it is less than 2). Use at least 3 different values for num_units or kernels per layer. Describe the model which gives the best performance in a table with details of each layer. Note that for the 2 selected layers in your model, you need to explore $3^2 = 9$ possible combinations of the number of units, to find the best configuration. Include your Jupyter notebook (call it q2c.ipynb) and a word/pdf file with the requested information (call it q2c.docx/pdf)

Q3. (Transfer learning; 100 points) Transfer learning can help reduce the time to train sophisticated deep learning models. The intuition behind transfer learning for image classification (note that it can be used in other problem domains as well) is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. You can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. The transfer learning tutorial on the TensorFlow site: https://www.tensorflow.org/tutorials/images/transfer_learning uses a pretrained MobileNetV2, which after fine tuning is able to perform a classification between images of cats and dogs with good accuracy. Take a look at the available pre-trained models for image classification available via Keras: https://keras.io/applications/. Also look at the catalog of datasets available via TensorFlow Datasets at: https://www.tensorflow.org/datasets/catalog/overview

(a) [50 points] Use transfer learning to tune a pretrained InceptionV3 model for the classification of images into various types of weather conditions, using the weather dataset provided. You must unzip the provided dataset and use it in your Colab environment or on your local machine, similar to how you did it for HW1. Note that unlike the cats and dogs example which is a binary classification problem, this is a multi-class classification problem where you must classify an image as belonging to one of the multiple possible weather conditions. Due to the complexity of the model, it is recommended that you work in Google Colab, starting from the Weather_InceptionV3-Q3a.ipynb skeleton file (that shows how to import and adapt InceptionV3 for the problem and achieves a 7.5% accuracy without transfer learning) and also refer to the TensorFlow transfer learning tutorial. Your goal is to perform transfer learning and achieve an accuracy of at least 82%. Include your Jupyter notebook (call it q3a.ipynb) and clearly indicate the accuracy achieved (on the test dataset).

(b) [50 points] Pick another pre-trained deep learning model and use transfer learning to improve upon the accuracy for the Weather condition classification problem, achieved by the modified InceptionV3 model. Include your Jupyter notebook (call it q3b.ipynb) and clearly indicate the accuracy achieved (on the test dataset).