CS/ECE 528: Embedded Systems and Machine Learning

Fall 2023

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

Assigned: 14 September 2023
Due: 21 September 2023

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) 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%).

(b) Now use Autokeras to find the best model for the Fashion MNIST dataset classification problem. Use the ImageClassifier 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 python notebook with the autokeras code for exploration. Compare the accuracy of your model from Q1(a) with that of the one obtained from Autokeras. Include a word/pdf file 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; 100 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) Starting with any of the models from Q1 (preferably the best model), use hparams for exploring the following optimizers: 'adam', 'adagrad', 'rmsprop', and 'sgd'. 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).

(b) 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', 'elu', 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.

(c) 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.

Q3. (Transfer learning; 75 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) Use transfer learning to tune a pretrained MobileNetV2 model for the classification of images in the Stanford dogs dataset (https://www.tensorflow.org/datasets/catalog/stanford_dogs). You can either use tfds or download the raw images from the link and access them for training. 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 120 dog breeds. Due to the complexity of the model, it is recommended that you work in Google Colab, starting from the tensorflow transfer learning tutorial notebook and making modifications to it. Share your notebook and clearly indicate your model details and accuracy achieved (on the test dataset).

(b) Pick another pre-trained deep learning model and use transfer learning to improve upon the accuracy for the Stanford dogs classification application, achieved by the modified MobileNetV2 model. Share your notebook and clearly indicate your model details and accuracy achieved (on the test dataset).