Critical Thinking Module 5 Option 1

Benjamin Gutierrez

Colorado State University Global

CSC 580: Applying Machine Learning and Neural Networks - Capstone

Dr. Lori Farr

February 5, 2023

My code uses a multi-layer feedforward neural network architecture with a sigmoid activation function in the output layer to train a neural network on the Tox21 dataset. My code attempts to discover the optimum neural network hyperparameters among batch size, learning rate, and number of epochs.

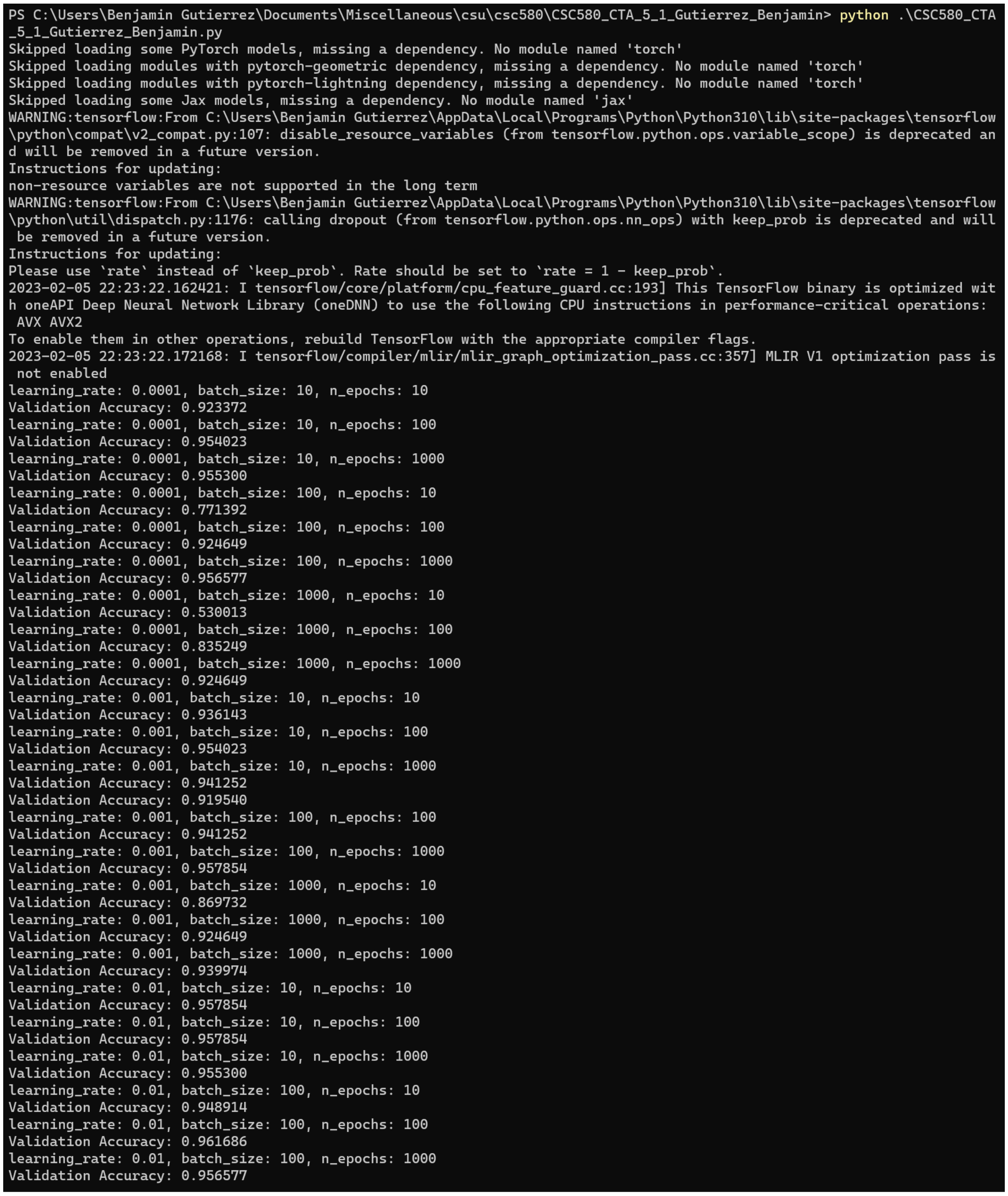
The Tox21 dataset is loaded as the initial step in the code (U.S. EPA, 2018). The deepchem package, which offers molecular machine learning capability, is used to import the dataset. Several global variables are defined in the code, which will be used to hold the optimal hyperparameters determined throughout the training process.

The tensorflow graph that depicts the neural network architecture is then defined. Tensorflow placeholders, variables, and operations are used to build the graph. A hidden layer of 50 neurons is used, followed by an output layer with a single sigmoid activation unit. The network's cost is specified as the sum of sigmoid cross-entropy losses and is optimized using the Adam optimizer. Dropout regularization is also implemented in the code by inserting a dropout layer between the hidden and output layers (Brownlee, 2018).

My code's last step is to train the neural network using mini-batching. The train function takes as parameters the learning rate, batch size, and number of epochs. Tensorflow operations create the computation graph and perform optimization throughout the training phase. The training process runs for the number of epochs supplied, with the batch size and learning rate chosen. After each epoch, the validation accuracy is recorded, and the best hyperparameters are saved in the global variables set at the start of the algorithm.

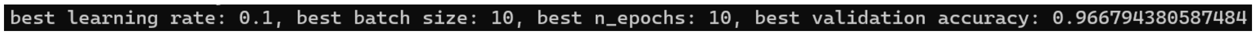
In conclusion, I trained a feedforward neural network on the Tox21 dataset and determined the optimum hyperparameters among batch size, learning rate, and epoch number. When the learning rate was 0.1, the batch size was 10, and the number of epochs was 10, the best validation accuracy was 96.67% as seen in Figure 2.

Figures



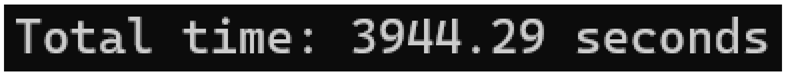
```
PS C:\Users\Benjamin Gutierrez\Documents\Miscellaneous\csu\csc580\CSC580_CTA_5_1_Gutierrez_Benjamin> python .\CSC580_CTA
_5_1_Gutierrez_Benjamin.py
Skipped loading some PyTorch models, missing a dependency. No module named 'torch'
Skipped loading modules with pytorch-geometric dependency, missing a dependency. No module named 'torch'
Skipped loading modules with pytorch-lightning dependency, missing a dependency. No module named 'torch'
Skipped loading some Jax models, missing a dependency. No module named 'jax'
WARNING:tensorflow:From C:\Users\Benjamin Gutierrez\AppData\Local\Programs\Python\Python310\lib\site-packages\tensorflow
\python\compat\v2_compat.py:107: disable_resource_variables (from tensorflow.python.ops.variable_scope) is deprecated an
d will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
WARNING:tensorflow:From C:\Users\Benjamin Gutierrez\AppData\Local\Programs\Python\Python310\lib\site-packages\tensorflow
\python\util\dispatch.py:1176: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will
 be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
2023-02-05 22:23:22.162421: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-02-05 22:23:22.172168: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:357] MLIR V1 optimization pass is
 not enabled
learning_rate: 0.0001, batch_size: 10, n_epochs: 10
Validation Accuracy: 0.923372
learning_rate: 0.0001, batch_size: 10, n_epochs: 100
Validation Accuracy: 0.954023
learning_rate: 0.0001, batch_size: 10, n_epochs: 1000
Validation Accuracy: 0.955300
learning_rate: 0.0001, batch_size: 100, n_epochs: 10
Validation Accuracy: 0.771392
learning_rate: 0.0001, batch_size: 100, n_epochs: 100
Validation Accuracy: 0.924649
learning_rate: 0.0001, batch_size: 100, n_epochs: 1000
Validation Accuracy: 0.956577
learning_rate: 0.0001, batch_size: 1000, n_epochs: 10
Validation Accuracy: 0.530013
learning_rate: 0.0001, batch_size: 1000, n_epochs: 100
Validation Accuracy: 0.835249
learning_rate: 0.0001, batch_size: 1000, n_epochs: 1000
Validation Accuracy: 0.924649
learning_rate: 0.001, batch_size: 10, n_epochs: 10
Validation Accuracy: 0.936143
learning_rate: 0.001, batch_size: 10, n_epochs: 100
Validation Accuracy: 0.954023
learning_rate: 0.001, batch_size: 10, n_epochs: 1000
Validation Accuracy: 0.941252
Validation Accuracy: 0.919540
learning_rate: 0.001, batch_size: 100, n_epochs: 100
Validation Accuracy: 0.941252
learning_rate: 0.001, batch_size: 100, n_epochs: 1000
Validation Accuracy: 0.957854
learning_rate: 0.001, batch_size: 1000, n_epochs: 10
Validation Accuracy: 0.869732
learning_rate: 0.001, batch_size: 1000, n_epochs: 100
Validation Accuracy: 0.924649
learning_rate: 0.001, batch_size: 1000, n_epochs: 1000
Validation Accuracy: 0.939974
learning_rate: 0.01, batch_size: 10, n_epochs: 10
Validation Accuracy: 0.957854
learning_rate: 0.01, batch_size: 10, n_epochs: 100
Validation Accuracy: 0.957854
learning_rate: 0.01, batch_size: 10, n_epochs: 1000
Validation Accuracy: 0.955300
learning_rate: 0.01, batch_size: 100, n_epochs: 10
Validation Accuracy: 0.948914
learning_rate: 0.01, batch_size: 100, n_epochs: 100
Validation Accuracy: 0.961686
learning_rate: 0.01, batch_size: 100, n_epochs: 1000
Validation Accuracy: 0.956577
```

Figure 1: Running my program comparing different hyperparameters



```
best learning rate: 0.1, best batch size: 10, best n_epochs: 10, best validation accuracy: 0.966794380587484
```

Figure 2: best validation accuracy corresponding to the best learning rate, best batch size, and best number of epochs



```
Total time: 3944.29 seconds
```

Figure 3: Total runtime of program

References

Brownlee, J. (2018, December 3). *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved February 5, 2023, from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

U.S. EPA. (2018, October 28). *Exploring ToxCast Data* | *US EPA*. Environmental Protection Agency. Retrieved February 5, 2023, from https://www.epa.gov/chemical-research/toxicity-forecaster-toxcasttm-data