Using a Generative Modeling Approach to a Variational Circuit Problem

A brief introduction to the task given to me was to train a variational circuit to give a 4-qubit output state from a random 4-qubit input state. I completed this task using Qiskit's Circuit Library to create and test the variational circuit as a quantum neural network. Through executing the job on a QASM Simulator, it allowed me to get results that should be expected on NISQ devices.

There were many possible approaches I could have taken this problem. After iterative testing with many quantum neural networks, I came to the conclusion that my selection performed the best.

OpflowQNN from Qiskit

My several circuits were trained on Qiskit's already created quantum neural networks with classification. It would be disrespectful for me to say that any of the classifiers created by the Qiskit Team and the Qiskit community aren't effective, but for my problem, they did not seem to produce the results I was hoping for. One realization I came to when training this classifier is that I realized that it could not solve classification problems with multiple classes. I came to this after help from a newly started community: *EntangledQuery.com*. This led me to my next set of tests.

CircuitQNN from Qiskit

The added benefit from this quantum neural network circuit was that I could specify the number of classes through the additional parameter: *output_shape*. I was very happy to see that I could train a variational circuit to have an output supporting multiple classes because that was the requirement of the task. However, after several days of testing, I noticed that the training time was taking too long, several hours in fact. Even though I tuned the hyperparameters to as low as I thought should be allowed, this was still causing a major problem. Some tests were done through switching between different feature maps like RealAmplitudes and TwoLocal. Other tuning was through changing the maxiter and other parameters of the feature map and the ansatz function. I had to find a solution to decrease training time and have an acceptable overall accuracy.

Custom Quantum Neural Network

My final testing process was through creating a custom quantum neural network. It allowed for much faster training times because I could see all the different parts of the QNN: feature map, variational circuit, loss/cost function, gradient, etc. More details are in the Jupyter Notebook. Through more testing of creating a custom quantum neural network, I believe a variational circuit could be trained the have significantly better accuracy than the one I created.