Sacramento State University

Project 2 - Report

CSC 180 Intelligent Systems

Due: March 12, 2021 @ 11:00am

Logan Hollmer (**ID:** 301559973)

Quinn Roemer (**ID:** 301323594)

Problem Statement:

In project 2 we were tasked with creating fully connected neural networks and convolutional neural networks to predict whether a network intrusion was from an authorized user or not. The dataset used for this project is called KDD CUP 99. This dataset includes the features necessary to predict whether a network signature is valid or invalid. This dataset contains a total of 22 different attack types, however, for the purpose of this project we focused on determining whether a connection was safe or not. Thus, turning this problem into binary classification. The first step in creating our models would be data pre-processing. In the case of this dataset, that meant encoding all categorical features with OHE (One Hot Encoding) and all numerical features with their z-score (standard score) or something similar. In addition, label encoding the *outcome* column to support binary classification. Once done, fully connected models with varying parameters (layer/neuron count, activation function, and optimizer) would be trained. In the same way, convolutional neural networks with varying parameters (CNN layer count, filter size, kernel size, etc) would be trained. All models being scored, graphed, and saved.

Methodology:

After discussion, we decided to tackle this problem statement similarly to the previous project (which we also teamed on). Each team member worked separately on their models, using different techniques to train their models in a friendly competition to see who could create the best model. As before, each member set up automatic or semi-automatic (in the case of CNN models) functions for building, training, and testing models. When collecting the data Logan decided to use the full dataset while Quinn used a 10% subset of the dataset. Each set was preprocessed similarly with categorical features being encoded using OHE, and the outcome being encoded as a binary label. However, some numerical features were encoded differently. Quinn used z-score normalization on all his numerical features while Logan used a log normalization function on the *srv_count. num_count,* and *duration* columns since they followed a logarithmic distribution and then performed a z-score on all the numeric features. After this, both performed a train/test split on their data.

For creating the standard fully connected neural networks Quinn decided to create a function that would build a model based on passed parameters. These parameters being the activation function, hidden layer count, and optimizer. Unlike in project 1 where such a function was completely random, the neuron count in

each layer was 2 raised to the hidden layer number, getting smaller as hidden layers were added. This was then topped off with a softmax layer for output. Logan used a similar approach in creating his own custom model builder. This also used powers of 2 to determine the neuron count in the hidden layers in descending amounts. In addition to this, he also had the ability to add kernel/activity regularizers in the last layer. With these functions, both members created scripts to automatically build models and train them, creating around 75 different models of various makeup.

For creating the convolutional neural network, Quinn also created a function to build a model based on passed parameters. This function allowed the activator, the number of convolution layers (convolution + MaxPooling), fully connected layers, kernel/pool size, and the neuron count to be specified. Each convolution layer increased in depth by powers of 2 as they were added, with the fully connected neuron count decreasing by half for each additional layer added. On top of this, dropout layers were added after the first and last convolution layers. In contrast, Logan created several CNN architectures inspired by proven models found online. This resulted in the best model trained between us. In total, we trained around 25 CNN models.

In addition to the above, Logan played around with training a fully connected model using Logistic Regression to pick out the top 30 most important features. However, this fared worse than our previous models.

Experimental Results and Analysis:

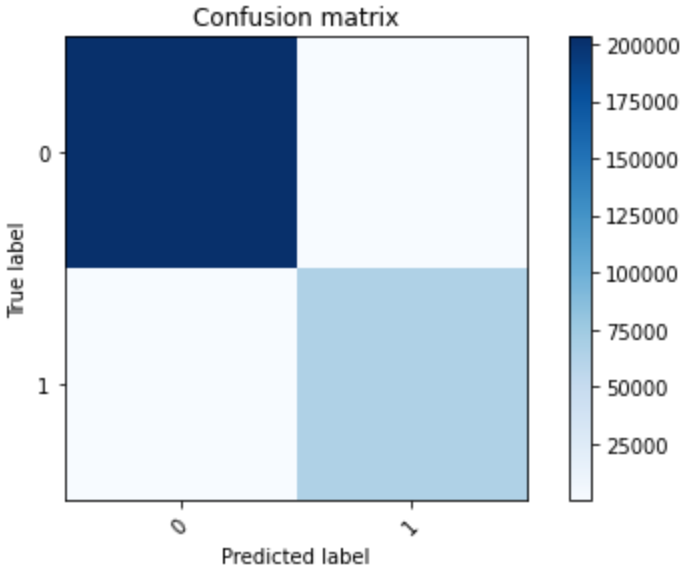
Logan's top models were CNN models and were closely followed by his fully connected models in proficiency. His best CNN model had an accuracy of 0.99961 and an average F1 score of 0.99961. This model is of a design that Logan came up with himself, inspired by the VGG-16 model. This model had the unique feature of having 2 convolution layers directly connected without a max-pooling layer between them. The architecture started with a convolution layer with 32 features where kernel_size=(1, 10) and strides=(1, 1). Next was a convolution layer with 128 features where kernel_size=(1, 5) and strides=(1, 1). After this, a max-pooling layer with pool_size=(1,3). Finally, the features were flattened and fed into a dense layer with 64 neurons. Followed by a dropout layer with a probability of 0.1, and another dense layer with 4 neurons. This was finally fed into the output layer. All layers shared the Tanh activation function with an SGD optimizer.

Quinn's top models were fully connected. However, the CNN models that he created did come close in score. His best model was a fully connected model that had an accuracy of 0.99871 and an average F1 score of

0.99871. This model had 6 layers whose neuron count decreased by a power of 2, so the first layer had 64 neurons and the last layer had 2 neurons. For all layers the activation function used was Sigmoid and the optimizer was Adam.

The best model created between us was the CNN model that Logan created with his custom architecture. Overall, we were very happy with how it performed. Given the time constraints and our entry-level knowledge of machine learning, this model exceeded our expectations. Below you can find the confusion matrix and the confusion matrix values for this model. Note that 0 represents normal connections and 1 represents malicious connections.

Confusion Matrix From our Best Model



Confusion Matrix Values

[[203255 33]

[73 65387]]

Upon looking at all the models created we found that the CNN models generally performed very similarly to the fully connected networks. Logan's CNN models slightly outperformed his fully connected models and Quinn's fully connected models slightly outperformed his CNN models. However, all the scores of our top models were within a tenth of a percent of accuracy with each other. So for this type of data, we believe that CNN and fully connected models are essentially equivalent in power.

Another thing we noticed was that attempting to do any form of regularization seemed to decrease the F1 score. In Table 1, you can see the top model from each architecture type that Logan trained. Observe that when performing regularization the accuracy and F1 score decreased by at least 0.1%.

When attempting to train a model based on the highest features from logistic regression we also observed a decrease in the accuracy and the F1 score. These models were interesting because their accuracy started out fairly high even after 1 epoch (higher than the fully connected networks after 1 epoch) however, the accuracy only climbed slightly after that and did not achieve anywhere near the accuracy of our other models.

Upon examining all the models that we created we noticed that Logan's models had a strong trend toward the Relu and Tanh activation function. On average, this function led to higher accuracy models and was present in many of our top models. However, Quinn's models seemed to favor the Sigmoid and Tanh activation functions. The optimizer function did not seem to make as much of a difference, both Adam and SGD led to models that performed well.

Table 1

Model type	Accuracy	Average F1 score	Activation function	Optimizer
CNN	0.99961	0.99961	Tanh	SGD
Fully connected network	0.999416	0.999416	Relu	SGD
Fully connected network using regularization	0.99868	0.99868	Relu	Adam
Fully connected network using features from logistic regression	0.98894	0.98886	Tanh	SGD

Table 1 displays Logan's top model from each network-style that he tried.

Task division and project reflection:

As previously stated, both group members did the entire project. Each member processed their own data and trained their own models. The writing of the report was split between each member with Quinn writing the first two and last sections, and Logan writing the section on *Experimental Results and Analysis*. The project notebook was compiled by Quinn and the project presentation was scripted and recorded by Logan.

In reflection on this project, we felt that using the entire dataset (instead of the 10% subset) resulted in better models. This can be seen as many of Logan's models were well within the 0.999 range for their F1 score while Quinn's models topped out at 0.998. We also saw that you don't necessarily have to normalize all numerical data the same way. Sometimes, depending on the type of data it is better to choose a different style of normalization. Likewise, if a column only contains values in the 0 and 1 range (binary) there is no need for normalization as it would be redundant.

Another thing of note is that because the dataset is heavily skewed toward the "good" connections the false positive rate was much higher than the false negative rate. If this system was going to be implemented in real life we would want the lowest false positive we could get as we would rather have false negatives than false positives. To deal with this we would have wanted to remove some of the "good" connections (downsampling) until the dataset was at least even. We might possibly even want to make sure that there were more "malicious" connections so that the model was able to properly categorize them. This would ensure a lower false positive rate which is the most important part of this model.

We found it interesting that the models trained using the features from LogisticRegression performed the worse. This seems to suggest that the result relies on most of the data points and that many of them are significant.

The biggest challenge we faced was training time. While our computers could easily handle the data size, we quickly became aware of the extended amount of time CNN models took to train. This was mostly resolved by automating our training through scripts and running them overnight.