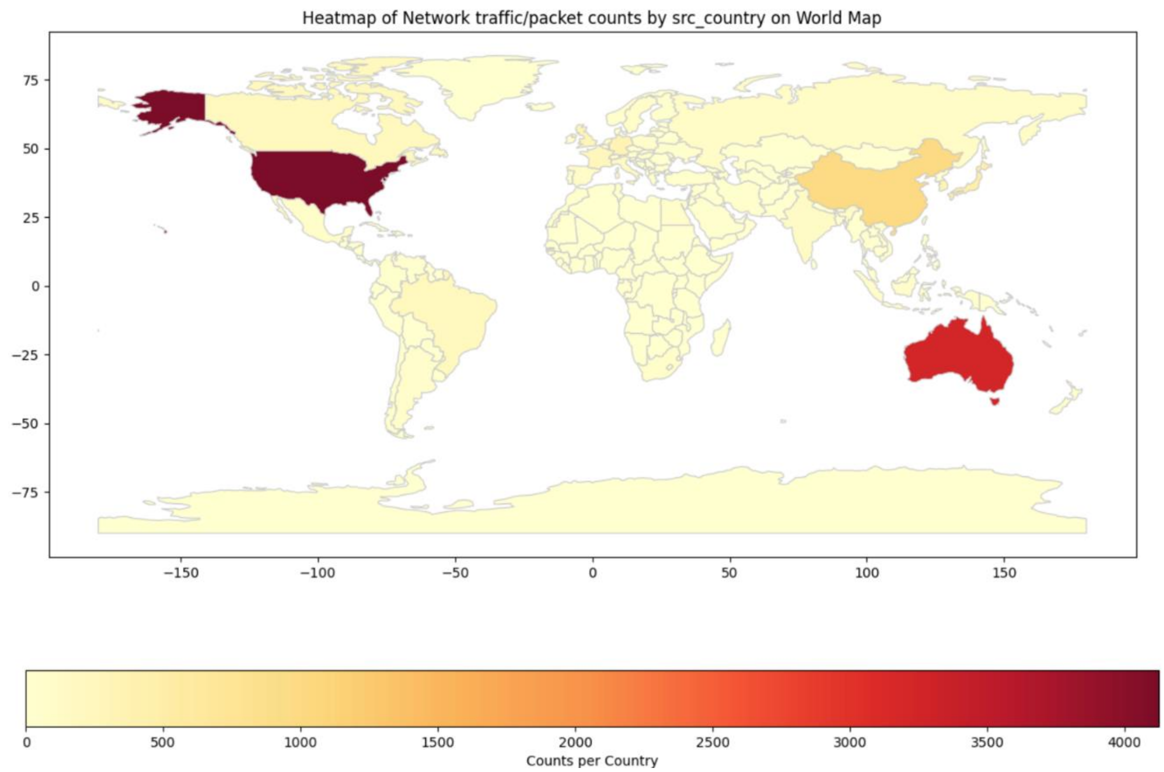
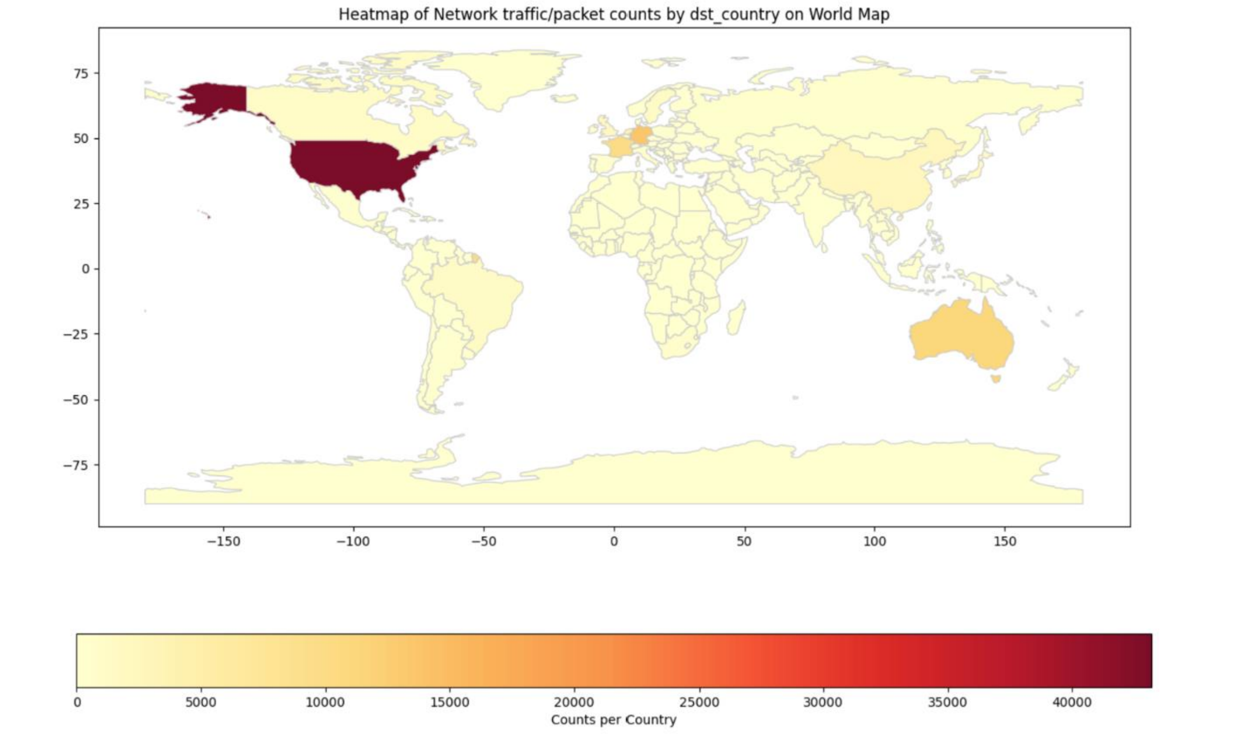
Detailed results summary

(Part I) Cyber security binary classification (threat “1” or normal “0”) and **(Part II)** multiclass classification or binning type of threats. (9 types of threats)

Part I – Binary Classification. (Cyber threat “1” or normal “0”)

We feed in data from network traffic packet sniffers. We perform EDA (Exploratory Data Analysis) including data sanitization, eliminating unnecessary columns or features, feature reduction to reduce/eliminate multi-collinearity, eliminating null values, and scaling the data using standard scalar. In addition to the required EDA data wrangling, we also generate cyber threat origination and destination location mapping using open-source IP to geolocation mapping and producing.





This is important to gauge the seriousness of cyber threats, preliminary analysis. This map with various heat maps corresponding to the number of traffic packets was produced using opensource tools and databases.

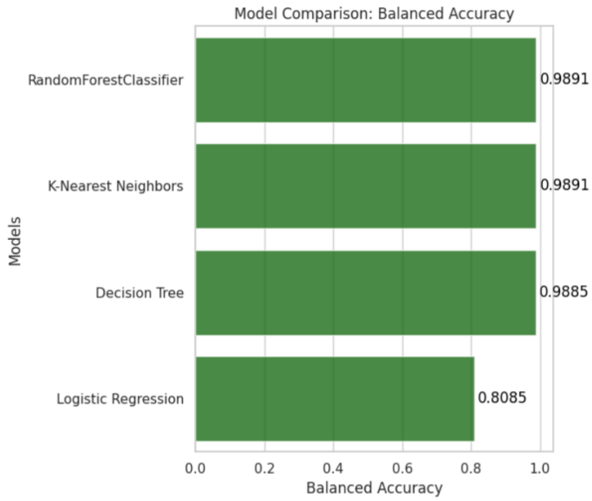
Binary Classification:

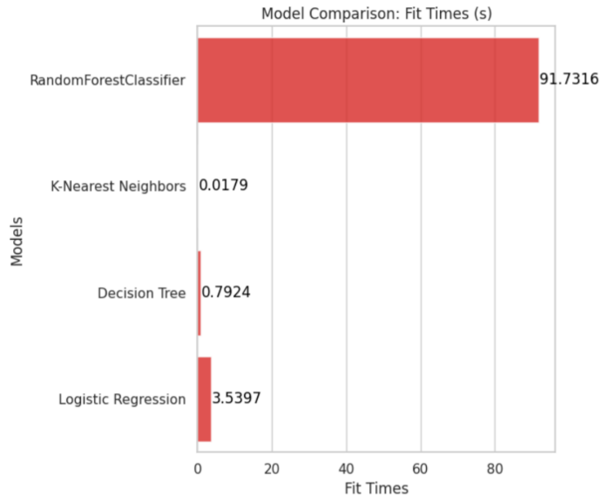
We evaluate several binary classification models to access the model performance. Among the models considered were

Models (Binary Classification)	First Pass Best Accuracy in %
LogisticRegression (L1 regularization)	85.98
LogisticRegression (L2 regularization)	85.96
DecisionTreeClassifier()	98.84
KNeighboursClassifier()	98.90
RandomForestClassifier()	98.91

We now short list four models from the first pass to optimize using GridSearchCV and gather results to further fine tune based on our needs.

Benchmarked results:





Based on the results we shortlist the top two models RandomForestClassifier and K-Nearest Neighbours to deploy.

Of the two which model should be deploy? What should the selection criteria be?

Both the models RandomForestClassifier & K-NearestNeighbours are identical in terms of performance. Since K-Nearest Neighbours has advantageous fit times. KNN is recommended.

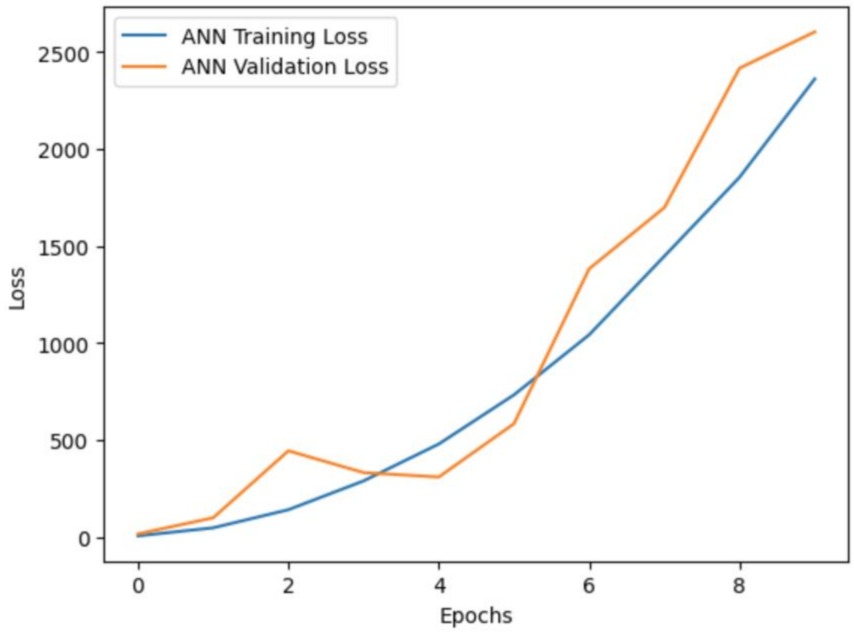
Part II – Multi Class Classification. (Bin the type of 9 threats to identify threats)

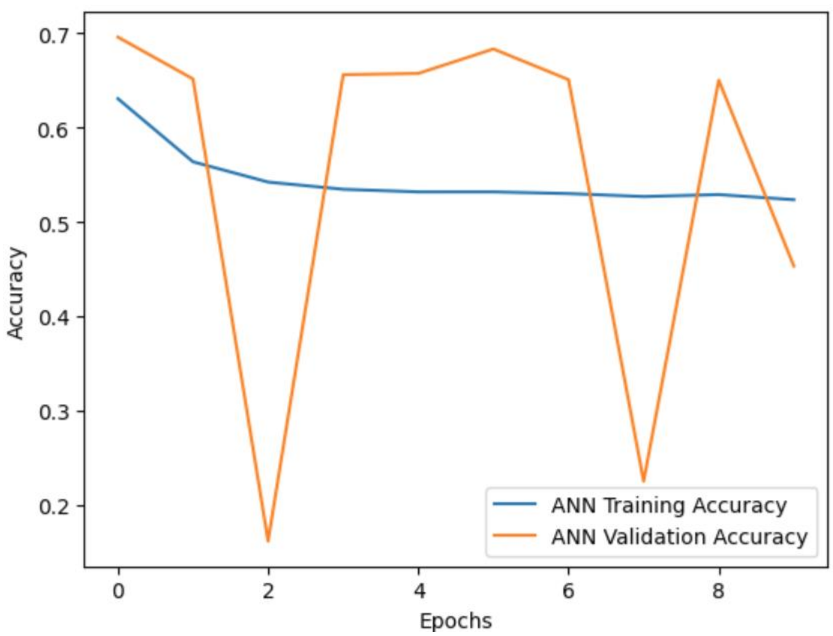
Next phase of this project involves neural networks. We will evaluate ANN, CNN, RNN and versions of RNN (GRU) for this phase. Decide on the ideal neural network, optimize/fine tune it and then deploy it for multiclass classification.

Some of the recorded results are listed below.

ANN (Artificial Neural Network):

```
Epoch 10/10
11527/11527 [==============================] - 34s 3ms/step - loss: 2361.5674
- accuracy: 0.5237 - val_loss: 2604.1521 - val_accuracy: 0.4533
2882/2882 [==============================] - 7s 2ms/step - loss: 2604.1521 -
accuracy: 0.4533
Test Accuracy ANN : 45.33%
```



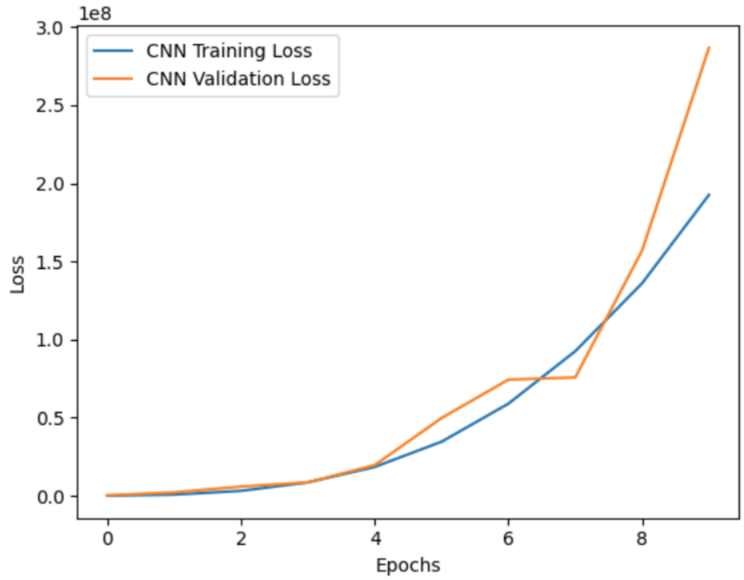


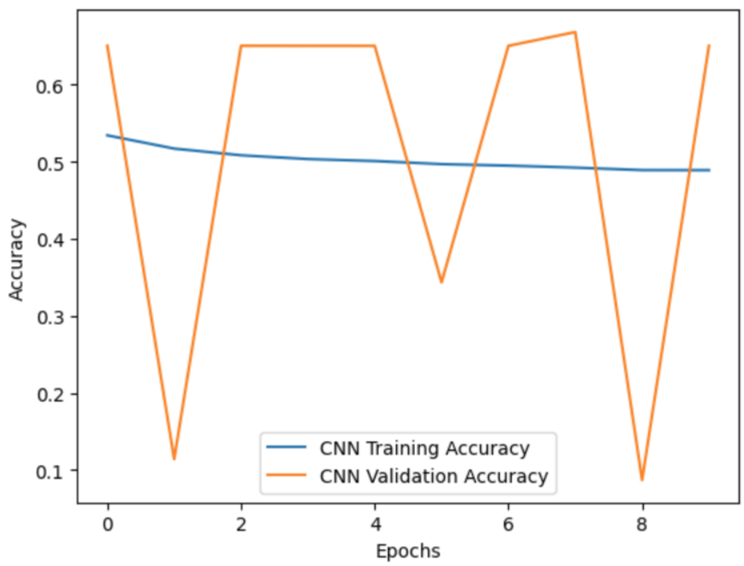
Observation: ANN, Loss seems to be increasing as epochs increase and Validation accuracy hovers around ~45%.

CNN (Convolutional Neural Network):

```
Epoch 10/10
11527/11527 [==============================] - 97s 8ms/step - loss: 192574160
.0000 - accuracy: 0.4890 - val_loss: 286683392.0000 - val_accuracy: 0.6504
2882/2882 [==============================] - 5s 2ms/step - loss: 2604.1521 -
accuracy: 0.4533
Test Accuracy CNN : 45.33%
```

—

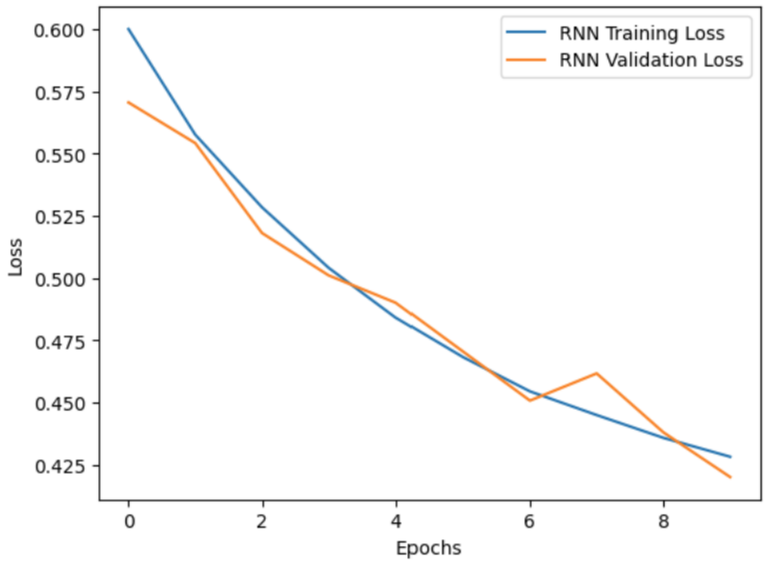


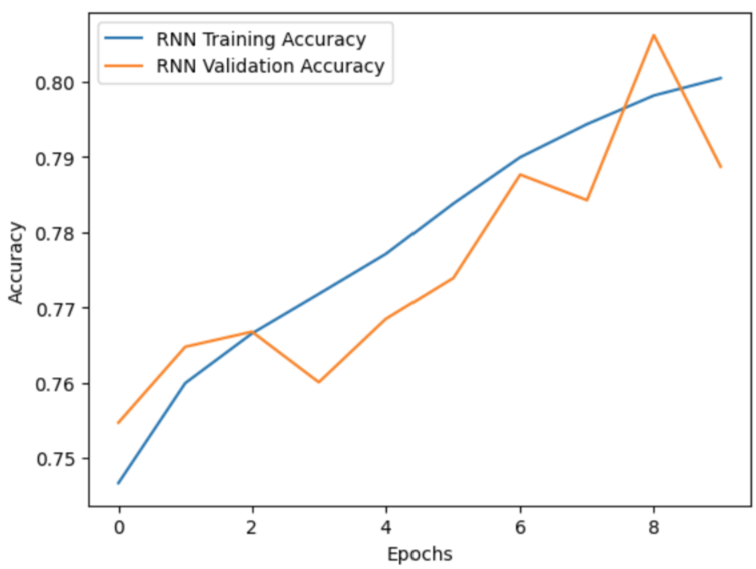


Observation: CNN, Loss seems to be increasing as epochs increase and Validation accuracy hovers around ~45%.

RNN (Recurrent Neural Network):

```
Epoch 10/10
11527/11527 [==============================] - 60s 5ms/step - loss: 0.4281 -
accuracy: 0.8005 - val_loss: 0.4200 - val_accuracy: 0.7887
2882/2882 [==============================] - 8s 3ms/step - loss: 0.4200 - acc
uracy: 0.7887
Test Accuracy RNN : 78.87%
```





Observation: **RNN,** Loss seems to be decreasing as epochs increase and Validation accuracy hovers around ~78%.

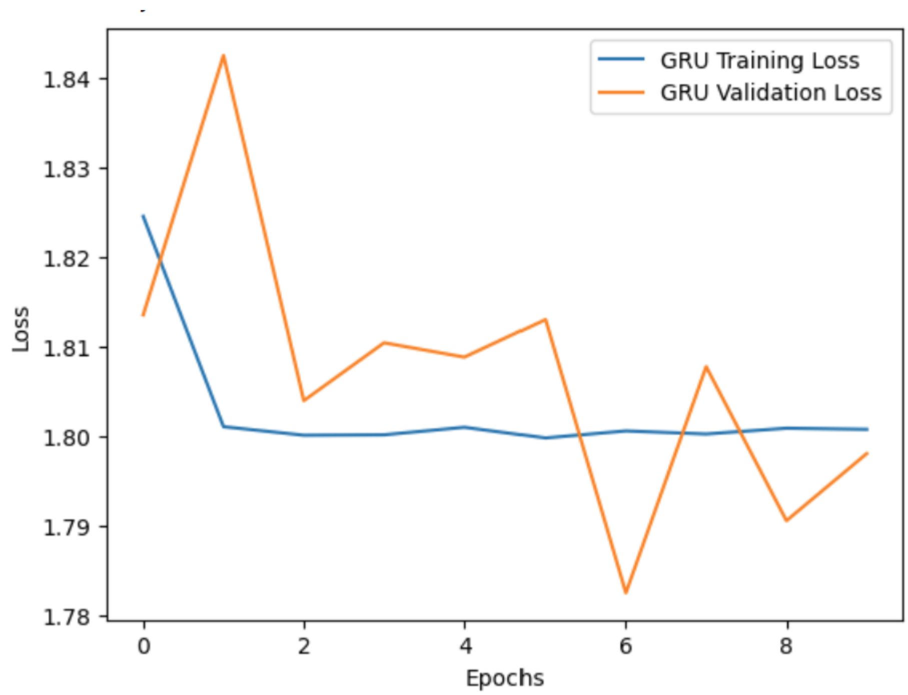
Inference: This suggests feedback mechanism for the RNN seems to work better for network threat multiclass classification. It makes sense, often network security breaches happens over time so its synchronous, starts with vulnerability at the edge, then a breach like an account compromise, password captures etc. happens to gain access and once access is gained then hackers methodically go after deeper and deeper layers of the network to gain access to data and critical or sensitive information. This happens over time and in sequence. With this new information we will further work on fine tuning RNN architecture, number of neurons, layers etc. to improve the performance of the model. We will focus on GRU (Gated Recurrent Unit) to improve on RNN.

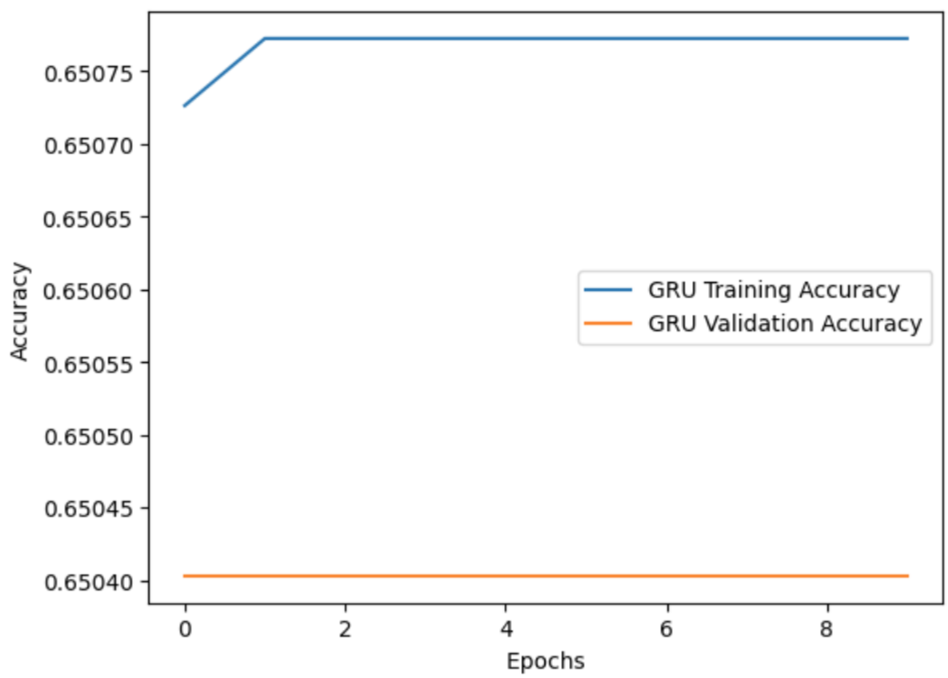
On further modeling increasing the number of Neurons or nodes and layers does not seem to help with the model performance in fact the model performance deteriorates.

Complex GRU model:

```
# Creating a GRU model
model_gru = Sequential()
model_gru.add(GRU(128, input_shape=(X_train_rnn.shape[1],
X_train_rnn.shape[2]), return_sequences=True,
kernel_regularizer=regularizers.l1(0.01)))
model_gru.add(Dropout(0.5))  # Adding dropout layer with a dropout rate of
0.5
model_gru.add(GRU(64, return_sequences=True,
kernel_regularizer=regularizers.l1(0.01)))  # Additional GRU layer with
return_sequences=True
model_gru.add(Dropout(0.5))  # Adding dropout layer with a dropout rate of
0.5
model_gru.add(GRU(32, kernel_regularizer=regularizers.l1(0.01)))  # Adding
another GRU layer
model_gru.add(Dense(9, activation='softmax'))
opt = Adam(learning_rate=0.01)
model_gru.compile(optimizer=opt, loss='categorical_crossentropy',
metrics=['accuracy'])
```

```
Epoch 10/10
11527/11527 [==============================] - 139s 12ms/step - loss: 1.8008
- accuracy: 0.6508 - val_loss: 1.7981 - val_accuracy: 0.6504
2882/2882 [==============================] - 9s 3ms/step - loss: 1.7981 - acc
uracy: 0.6504
Test Accuracy GRU : 65.04%
```



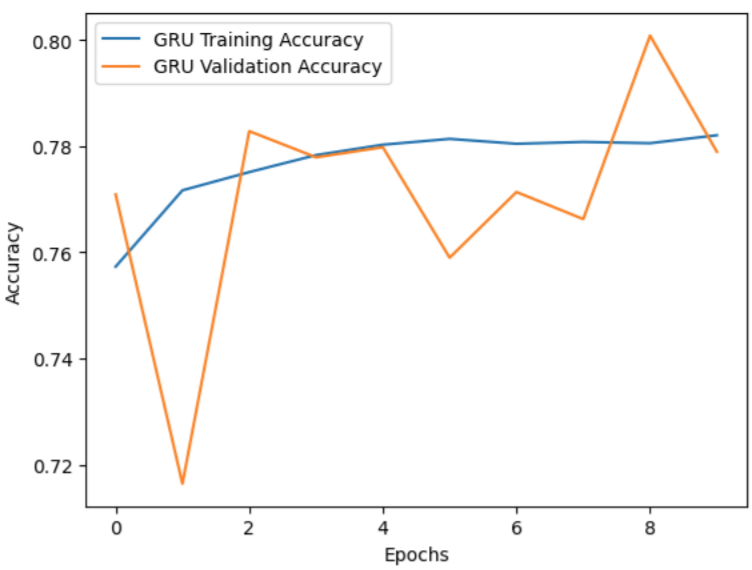


Observation: Sometimes simple models seem to work better. With this new knowledge we will fine tune the simple GRU model to improve the model's performance.

We will focus on simple RNN GRU model to improve the multiclass classification.

```
# Creating a GRU model
model_gru = Sequential()
model_gru.add(GRU(64, input_shape=(X_train_rnn.shape[1],
X_train_rnn.shape[2]), return_sequences=True))
model_gru.add(GRU(32))
model_gru.add(Dense(9, activation='softmax'))  # Softmax for 9-class
classification for each output
```

```
Epoch 10/10
11527/11527 [==============================] - 98s 9ms/step - loss: 0.6804 -
accuracy: 0.7821 - val_loss: 0.6875 - val_accuracy: 0.7790
2882/2882 [==============================] - 10s 4ms/step - loss: 0.6875 - ac
curacy: 0.7790
```



GRU Summary Report.

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru_19 (GRU)                (None, 1, 64)             21312

 gru_20 (GRU)                (None, 32)                9408

 dense_7 (Dense)             (None, 9)                 297

=================================================================
Total params: 31,017
Trainable params: 31,017
Non-trainable params: 0
```

Result and Next steps:

Cyber Security network traffic data can be efficiently and accurately classified as threat or normal. Once classified we can further classify into the type of threats using neural networks.

This output can be valuable to alert necessary cybersecurity personnel to protect the network and further develop dynamic firewall rules as a generative AI defense mechanism. In this world of constant cybersecurity breaches it is imperative to develop automated AI driven defense mechanisms and firewall rules. Ideal testing can be done in production network using the techniques discussed in the project or directly applying the models we trained. It is also best practice to develop KPI (Key Performance Index) to constantly evaluate the models to keep it up to date on new threats and accuracy upkeep.