```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras import Model, Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from tensorflow.keras.losses import MeanSquaredLogarithmicError
from tensorflow.keras.losses import MeanSquaredError
```

```
# load the dataset
data = pd.read_csv('/content/SDN_DDoS_.csv')
data.head()
```

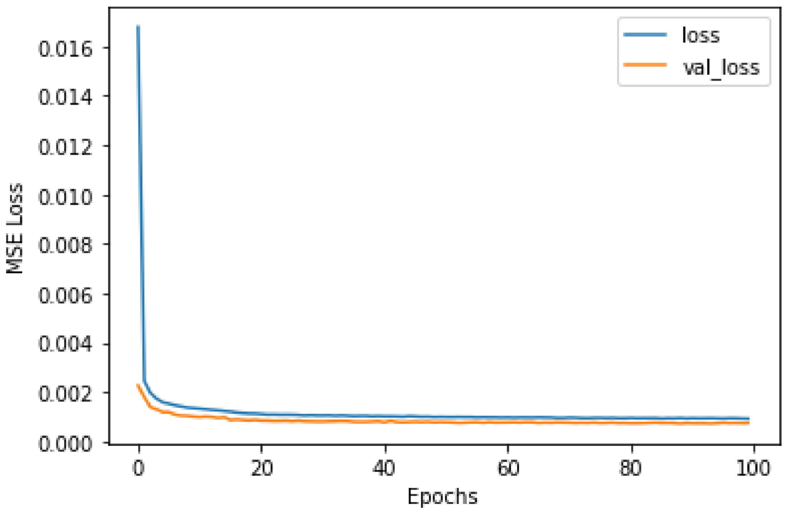
	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	Fwd Pkt Len Std	Bwd Pkt Len Max	Bwd Pkt Len Min
0	245230	44	40	124937	1071	9100	0	2839.477273	1839.508257	517	0
1	1605449	107	149	1071	439537	517	0	10.009346	67.496680	27300	0
2	53078	5	5	66	758	66	0	13.200000	29.516097	638	0
3	6975	1	1	0	0	0	0	0.000000	0.000000	0	0
4	190141	13	16	780	11085	427	0	60.000000	130.042942	2596	0

```
features = data.drop('Label', axis=1)
target = data['Label']

x_train, x_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, stratify=target
)

# use case is novelty detection so use only the normal data
# for training
train_index = y_train[y_train == 0].index
train_data = x_train.loc[train_index]

# min max scale the input data
min_max_scaler = MinMaxScaler(feature_range=(0, 1))
x_train_scaled = min_max_scaler.fit_transform(train_data.copy())
```

```
x_test_scaled = min_max_scaler.transform(x_test.copy())


class AutoEncoder(Model):
  def __init__(self, output_units, code_size=16):
    super().__init__()
    self.encoder = Sequential([
      Dense(66, activation='relu'),
      Dropout(0.1),
      Dense(32, activation='relu'),
      Dropout(0.1),
      Dense(16, activation='relu'),
      Dropout(0.1),
      Dense(code_size, activation='relu')
    ])
    self.decoder = Sequential([
      Dense(16, activation='relu'),
      Dropout(0.1),
      Dense(32, activation='relu'),
      Dropout(0.1),
      Dense(66, activation='relu'),
      Dropout(0.1),
      Dense(output_units, activation='sigmoid')
    ])

  def call(self, inputs):
    encoded = self.encoder(inputs)
    decoded = self.decoder(encoded)
    return decoded

model = AutoEncoder(output_units=x_train_scaled.shape[1])
# configurations of model
model.compile(loss='mse', metrics=['accuracy'], optimizer='adam')

history = model.fit(
    x_train_scaled,
    x_train_scaled,
    epochs=100,
    batch_size=64,
    validation_data=(x_test_scaled, x_test_scaled)
)
```

```
Epoch 73/100
856/856 [==============================] - 3s 4ms/step - loss: 9.3982e-04 - accuracy
Epoch 74/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3419e-04 - accuracy
Epoch 75/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4666e-04 - accuracy
Epoch 76/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4755e-04 - accuracy
Epoch 77/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3870e-04 - accuracy
Epoch 78/100
Epoch 78/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4721e-04 - accuracy
Epoch 79/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3328e-04 - accuracy
Epoch 80/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3171e-04 - accuracy
Epoch 81/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4375e-04 - accuracy
Epoch 82/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3237e-04 - accuracy
Epoch 83/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3574e-04 - accuracy
Epoch 84/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3613e-04 - accuracy
Epoch 85/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4016e-04 - accuracy
Epoch 86/100
856/856 [==============================] - 3s 3ms/step - loss: 9.1987e-04 - accuracy
Epoch 87/100
856/856 [==============================] - 3s 4ms/step - loss: 9.3433e-04 - accuracy
Epoch 88/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2752e-04 - accuracy
Epoch 89/100
856/856 [==============================] - 3s 3ms/step - loss: 9.4233e-04 - accuracy
Epoch 90/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2441e-04 - accuracy
Epoch 91/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3465e-04 - accuracy
Epoch 92/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2807e-04 - accuracy
Epoch 93/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3031e-04 - accuracy
Epoch 94/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3526e-04 - accuracy
Epoch 95/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2642e-04 - accuracy
Epoch 96/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2295e-04 - accuracy
Epoch 97/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3248e-04 - accuracy
Epoch 98/100
856/856 [==============================] - 3s 3ms/step - loss: 9.3180e-04 - accuracy
Epoch 99/100
856/856 [==============================] - 3s 3ms/step - loss: 9.2180e-04 - accuracy
Epoch 100/100
856/856 [==============================] - 3s 3ms/step - loss: 9.1545e-04 - accuracy
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.xlabel('Epochs')
plt.ylabel('MSE Loss')
plt.legend(['loss', 'val_loss'])
plt.show()
```



```
def find_threshold(model, x_train_scaled):
  reconstructions = model.predict(x_train_scaled)
  # provides losses of individual instances
  reconstruction_errors = tf.keras.losses.msle(reconstructions, x_train_scaled)
  # threshold for anomaly scores
  threshold = np.mean(reconstruction_errors.numpy()) + np.std(reconstruction_errors.numpy())
  return threshold

def get_predictions(model, x_test_scaled, threshold):
  predictions = model.predict(x_test_scaled)
  # provides losses of individual instances
  errors = tf.keras.losses.msle(predictions, x_test_scaled)

  anomaly_mask = pd.Series(errors) > threshold
  preds = anomaly_mask.map(lambda x: 1.0 if x == True else 0.0)
  return preds

threshold = find_threshold(model, x_train_scaled)
print(f"Threshold: {threshold}")
```

```
     Threshold: 0.0019437121899367092
```

```
predictions = get_predictions(model, x_test_scaled, threshold)
accuracy_score(y_test,predictions)
```

```
     0.8565040115743785
```

```
!pip install -U keras-tuner
```

```
     Collecting keras-tuner
       Downloading keras_tuner-1.1.0-py3-none-any.whl (98 kB)
          |████████████████████████████████| 98 kB 5.6 MB/s
     Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from ke
     Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-packages (f
     Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from ke
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-packages (from
Collecting kt-legacy
  Downloading kt_legacy-1.0.4-py3-none-any.whl (9.6 kB)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.7
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/li
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pytho
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dis
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-package
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.1.0 kt-legacy-1.0.4
```

```
import kerastuner as kt

class AutoEncoderTuner(Model):

  def __init__(self, hp, output_units, code_size=8):
    super().__init__()
    dense_1_units = hp.Int('dense_1_units', min_value=16, max_value=72, step=4)
    dense_2_units = hp.Int('dense_2_units', min_value=16, max_value=72, step=4)
    dense_3_units = hp.Int('dense_3_units', min_value=16, max_value=72, step=4)
    dense_4_units = hp.Int('dense_4_units', min_value=16, max_value=72, step=4)
    dense_5_units = hp.Int('dense_5_units', min_value=16, max_value=72, step=4)
    dense_6_units = hp.Int('dense_6_units', min_value=16, max_value=72, step=4)

    self.encoder = Sequential([
      Dense(dense_1_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_2_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_3_units, activation='relu'),
      Dropout(0.1),
      Dense(code_size, activation='relu')
    ])
    self.decoder = Sequential([
      Dense(dense_4_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_5_units, activation='relu'),
      Dropout(0.1),
      Dense(dense_6_units, activation='relu'),
      Dropout(0.1),
      Dense(output_units, activation='sigmoid')
    ])

  def call(self, inputs):
    encoded = self.encoder(inputs)
    decoded = self.decoder(encoded)
    return decoded


def build_model(hp):
  model = AutoEncoderTuner(hp, 66)
  hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
  model.compile(
      loss='msle',
      optimizer=Adam(learning_rate=hp_learning_rate),
  )
  return model
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: `im
  """Entry point for launching an IPython kernel.
```

```
tuner = kt.Hyperband(
    build_model,
    objective='val_loss',
    max_epochs=20,
    factor=3,
    directory='autoencoder',
    project_name='tuning_autoencoder6'
)

tuner.search(
    x_train_scaled,
    x_train_scaled,
    epochs=20,
    batch_size=64,
    validation_data=(x_test_scaled, x_test_scaled)

)
```

```
Trial 31 Complete [00h 01m 23s]
val_loss: 0.008662069216370583

Best val_loss So Far: 0.00047473839367739856
Total elapsed time: 00h 16m 10s
INFO:tensorflow:Oracle triggered exit
```

```
hparams = [f'dense_{i}_units' for i in range(1,7)] + ['learning_rate']
best_hyperparams = tuner.get_best_hyperparameters()
for hps in hparams:
  print(f"{hps}: {best_hyperparams[0][hps]}")
```

```
dense_1_units: 68
dense_2_units: 24
dense_3_units: 36
dense_4_units: 28
dense_5_units: 36
dense_6_units: 48
learning_rate: 0.001
```

```
best_model = tuner.get_best_models()[0]
best_model.compile(loss='mse',metrics=['accuracy'], optimizer=Adam(0.0001))

best_model.fit(
    x_train_scaled,
    x_train_scaled,
    epochs=100,
    batch_size=64,
    validation_data=(x_test_scaled, x_test_scaled)
)
```

```
Epoch 73/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9223e-04 - accuracy
Epoch 74/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9737e-04 - accuracy
Epoch 75/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9731e-04 - accuracy
Epoch 76/100
856/856 [==============================] - 3s 3ms/step - loss: 0.0010 - accuracy: 0.
Epoch 77/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9506e-04 - accuracy
Epoch 78/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8954e-04 - accuracy
Epoch 79/100
856/856 [==============================] - 3s 3ms/step - loss: 9.9706e-04 - accuracy
Epoch 80/100
```

```
856/856 [==============================] - 3s 3ms/step - loss: 9.9426e-04 - accuracy
Epoch 81/100
856/856 [==============================] - 3s 3ms/step - loss: 9.9205e-04 - accuracy
Epoch 82/100
856/856 [==============================] - 3s 3ms/step - loss: 9.9819e-04 - accuracy
Epoch 83/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9292e-04 - accuracy
Epoch 84/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8940e-04 - accuracy
Epoch 85/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9203e-04 - accuracy
Epoch 86/100
856/856 [==============================] - 3s 4ms/step - loss: 9.9264e-04 - accuracy
Epoch 87/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8757e-04 - accuracy
Epoch 88/100
856/856 [==============================] - 3s 3ms/step - loss: 9.8893e-04 - accuracy
Epoch 89/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8704e-04 - accuracy
Epoch 90/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8996e-04 - accuracy
Epoch 91/100
856/856 [==============================] - 3s 3ms/step - loss: 9.7606e-04 - accuracy
Epoch 92/100
856/856 [==============================] - 3s 3ms/step - loss: 9.8898e-04 - accuracy
Epoch 93/100
856/856 [==============================] - 3s 3ms/step - loss: 9.8941e-04 - accuracy
Epoch 94/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8237e-04 - accuracy
Epoch 95/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8939e-04 - accuracy
Epoch 96/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8299e-04 - accuracy
Epoch 97/100
856/856 [==============================] - 3s 3ms/step - loss: 9.8565e-04 - accuracy
Epoch 98/100
856/856 [==============================] - 3s 4ms/step - loss: 9.7922e-04 - accuracy
Epoch 99/100
856/856 [==============================] - 3s 4ms/step - loss: 9.8607e-04 - accuracy
Epoch 100/100
856/856 [==============================] - 3s 3ms/step - loss: 9.8713e-04 - accuracy
<keras.callbacks.History at 0x7f9fe09dc650>
```

```
threshold_ = find_threshold(best_model, x_train_scaled)
preds_ = get_predictions(best_model, x_test_scaled, threshold_)
accuracy_score(preds_, y_test)
```

```
0.8567670656319873
```

✓ 3s completed at 7:54 PM ● ✕