J COMPONENT REVIEW II

MALWARE ANALYSIS IN DATA SCIENCE

BOTNET DETECTION WITH AUTOENCODER

CSE4053 - E1 Slot

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Table of Content

BOTNET DETECTION WITH AUTOENCODER

- Introduction
- Problem Statement
- Architecture
- Proposed Methodology
- Architecture Diagram & Code
- Modules Explained

INTRODUCTION

The proliferation of IoT devices which can be more easily compromised than desktop computers has led to an increase in the occurrence of IoT-based botnet attacks. In order to mitigate this new threat, there is a need to develop new methods for detecting attacks launched from compromised IoT devices and differentiate between hour and millisecond long IoT-based attacks.

In this paper we propose and empirically evaluate

- A novel network-based anomaly detection method which extracts behavior snapshots of the network and
- Uses deep autoencoders to detect anomalous network traffic emanating from compromised IoT devices.
- Specifically, we extract statistical features which capture behavioral snapshots of benign IoT traffic and train a deep autoencoder (one for each device) to learn the IoT's normal behaviors.

To evaluate our method, we infected nine commercial IoT devices in our lab with two of the most widely known IoT-based botnets, Mirai and BASHLITE. Our evaluation results demonstrated our proposed method’s ability to accurately and instantly detect the attacks as they were being launched from the compromised IoT devices which were part of a botnet.

PROBLEM STATEMENT

We model botnet traffic identification as an anomaly detection task, aiming at establishing a baseline of benign traffic, in order to detect unusual behavior using NetFlow data.

ARCHITECTURE 1 - AUTOENCODERS

- We use deep autoencoders and maintain a model for each IoT device separately. An autoencoder is trained to reconstruct its inputs after some compression which ensures that, the network learns the relation among its input features.
- Each autoencoder had an input layer whose dimension is equal to the number of features in the dataset.
- Autoencoders effectively perform dimensionality reduction internally, such that the code layer between the encoders and decoder efficiently compress the input layer and reflects its essential characteristics.
- If an autoencoder is trained on benign instances only, then it will succeed at reconstructing normal observations, but fail at reconstructing abnormal observations. When a significant re-construction error is detected, then we classify the given observations as being an anomaly.

ARCHITECTURE 2 - ANN

ANNs can be used for IoT botnet detection by training the network on features that distinguish botnet traffic from normal traffic. This involves collecting and preprocessing network traffic data and building an ANN with a fully connected network and multiple hidden layers.

ANN's is trained using iterative weight and bias adjustments to minimize the prediction error, and tested using metrics like accuracy, precision, recall, and F1-score. Once the ANN is trained, it can be deployed in a real-time environment for botnet detection by integrating it into an application or network infrastructure. Overall, ANN can be a powerful tool for IoT botnet detection, but requires careful data selection, network tuning, and testing and evaluation to ensure its effectiveness.

ARCHITECTURE 3 - LSTM & CNN

To improve the accuracy of IoT botnet detection, both LSTM and CNN can be used together. The process involves collecting network traffic data and identifying features that are indicative of botnet traffic. The data is preprocessed, and the CNN is used to extract features from the data while the LSTM captures temporal dependencies.

Model is trained and tested using a testing set to evaluate its performance. Finally, the model is deployed in a real-time environment for botnet detection by integrating it into an application or network infrastructure. This approach requires careful data preparation, model tuning, and testing and evaluation to ensure its effectiveness. LSTM and CNN together can improve the accuracy of botnet detection.

ARCHITECTURE 4 - DEEP RESIDUCAL CNN

To improve botnet detection in IoT, Deep Residual CNN can be implemented by collecting network traffic data and preprocessing it. The Deep Residual CNN consists of multiple residual blocks that allow the network to learn residual functions.

Model is then trained using a loss function and optimizer and tested for performance using metrics like accuracy, precision, recall, and F1-score. Finally, the model is deployed in a real-time environment by integrating it into an application or network infrastructure. This approach requires careful data preparation, model tuning, and testing and evaluation to ensure its effectiveness in extracting features from the network traffic data.

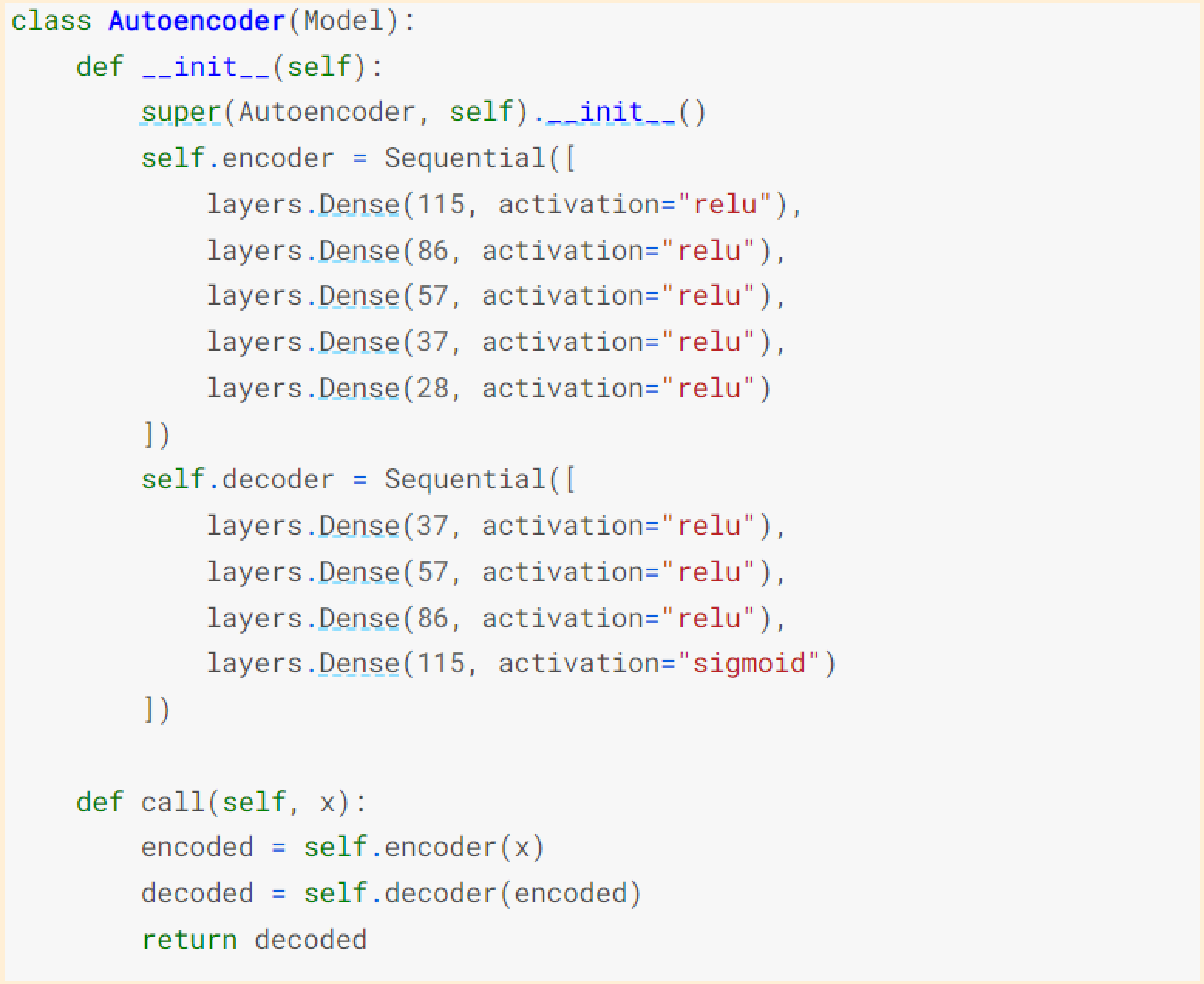
In our experiments, four hidden layers of encoders were set at decreasing sizes of 75%, 50%, 33%, and 25% of the input layer’s dimension. The next layers were decoders, with the same sizes as the encoders, however with an increasing order (starting from33%).

PROPOSED METHODOLOGY

Our approach is based on deep autoencoders for each device that are skilled in statistical patterns acquired from benign traffic data. Applying discovered abnormalities to fresh (potentially infected) data from an IoT device may indicate that the device is compromised. The four primary stages of this strategy are.

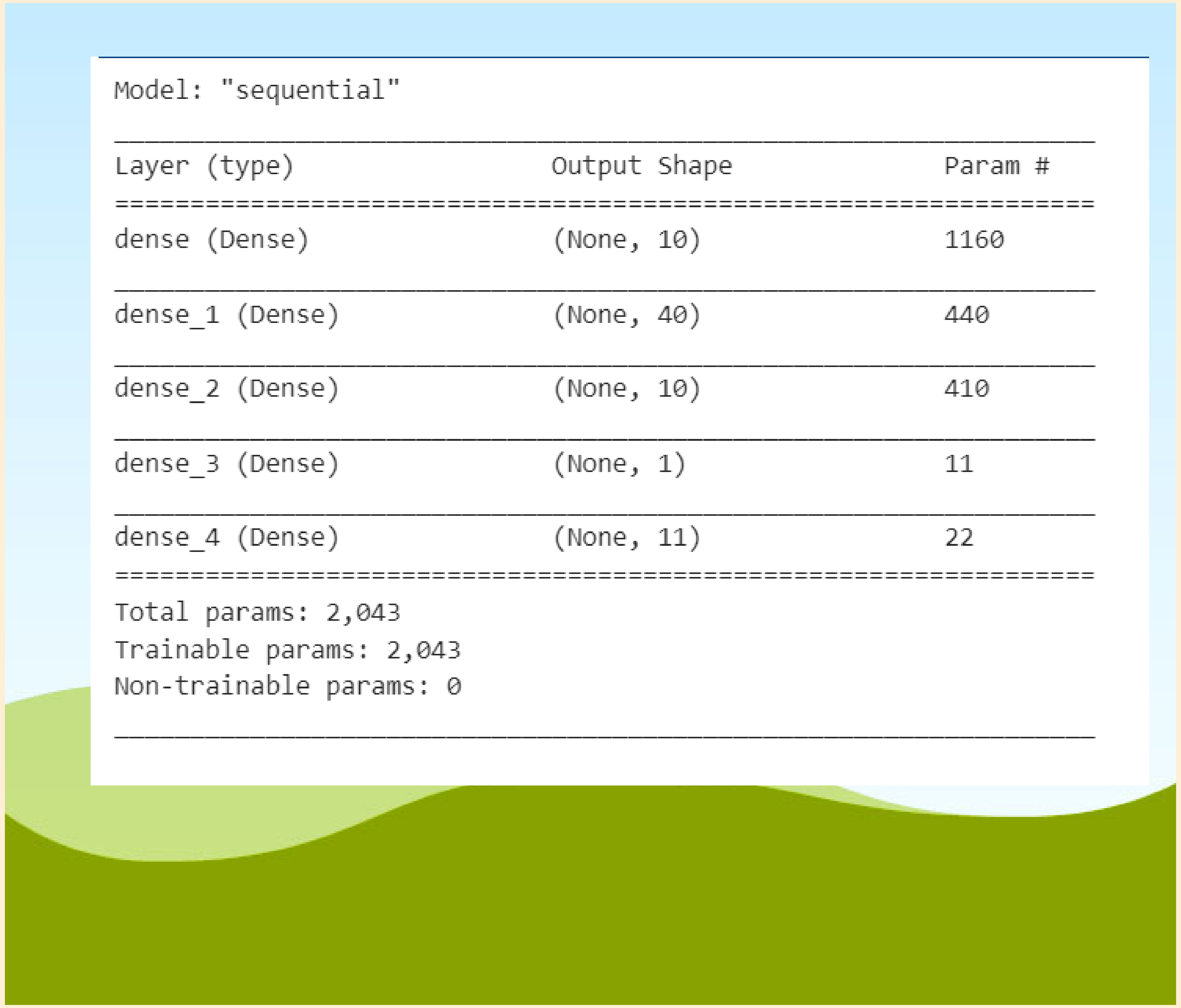
·data collection
·feature extraction
·training an anomaly detector
·continuous monitoring

ARCHITECTURE CODE 1 - AUTOENCODERS



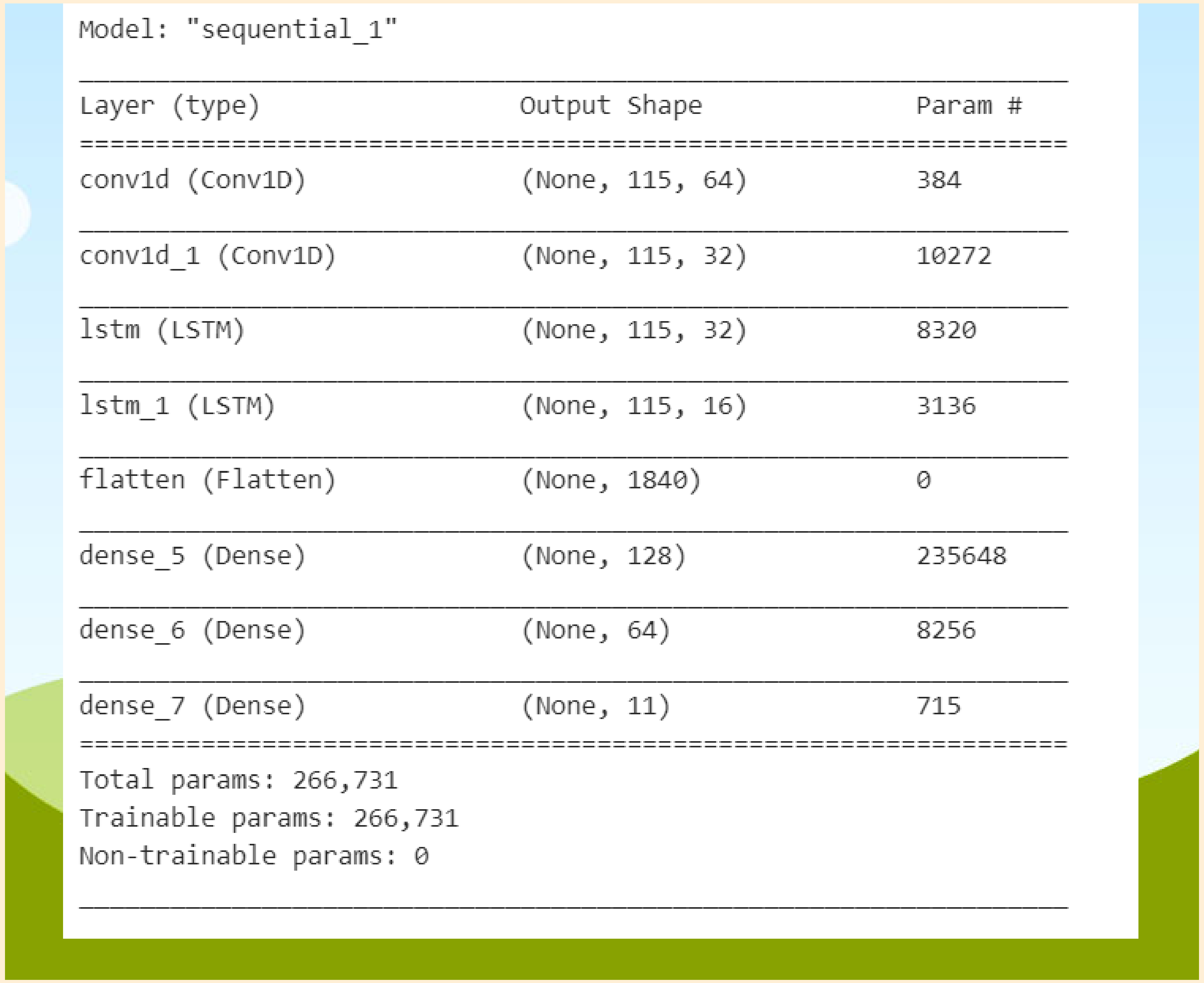
```
class Autoencoder(Model):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = Sequential([
            layers.Dense(115, activation="relu"),
            layers.Dense(86, activation="relu"),
            layers.Dense(57, activation="relu"),
            layers.Dense(37, activation="relu"),
            layers.Dense(28, activation="relu")
        ])
        self.decoder = Sequential([
            layers.Dense(37, activation="relu"),
            layers.Dense(57, activation="relu"),
            layers.Dense(86, activation="relu"),
            layers.Dense(115, activation="sigmoid")
        ])

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

ARCHITECTURE CODE 2 - ANN



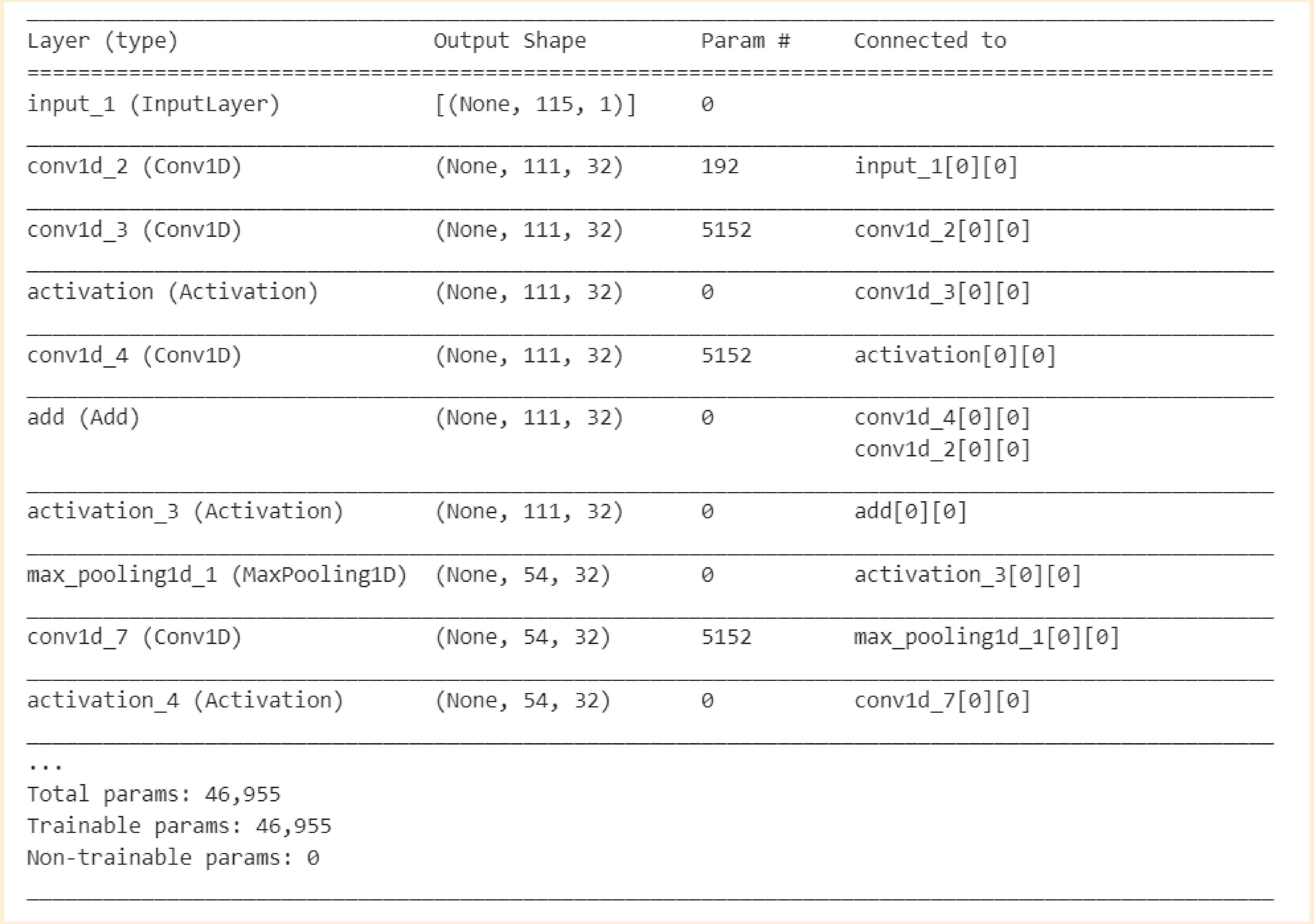
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                1160
_________________________________________________________________
dense_1 (Dense)              (None, 40)                440
_________________________________________________________________
dense_2 (Dense)              (None, 10)                410
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
_________________________________________________________________
dense_4 (Dense)              (None, 11)                22
=================================================================
Total params: 2,043
Trainable params: 2,043
Non-trainable params: 0
_________________________________________________________________
```

ARCHITECTURE CODE 3 - LSTM & CNN



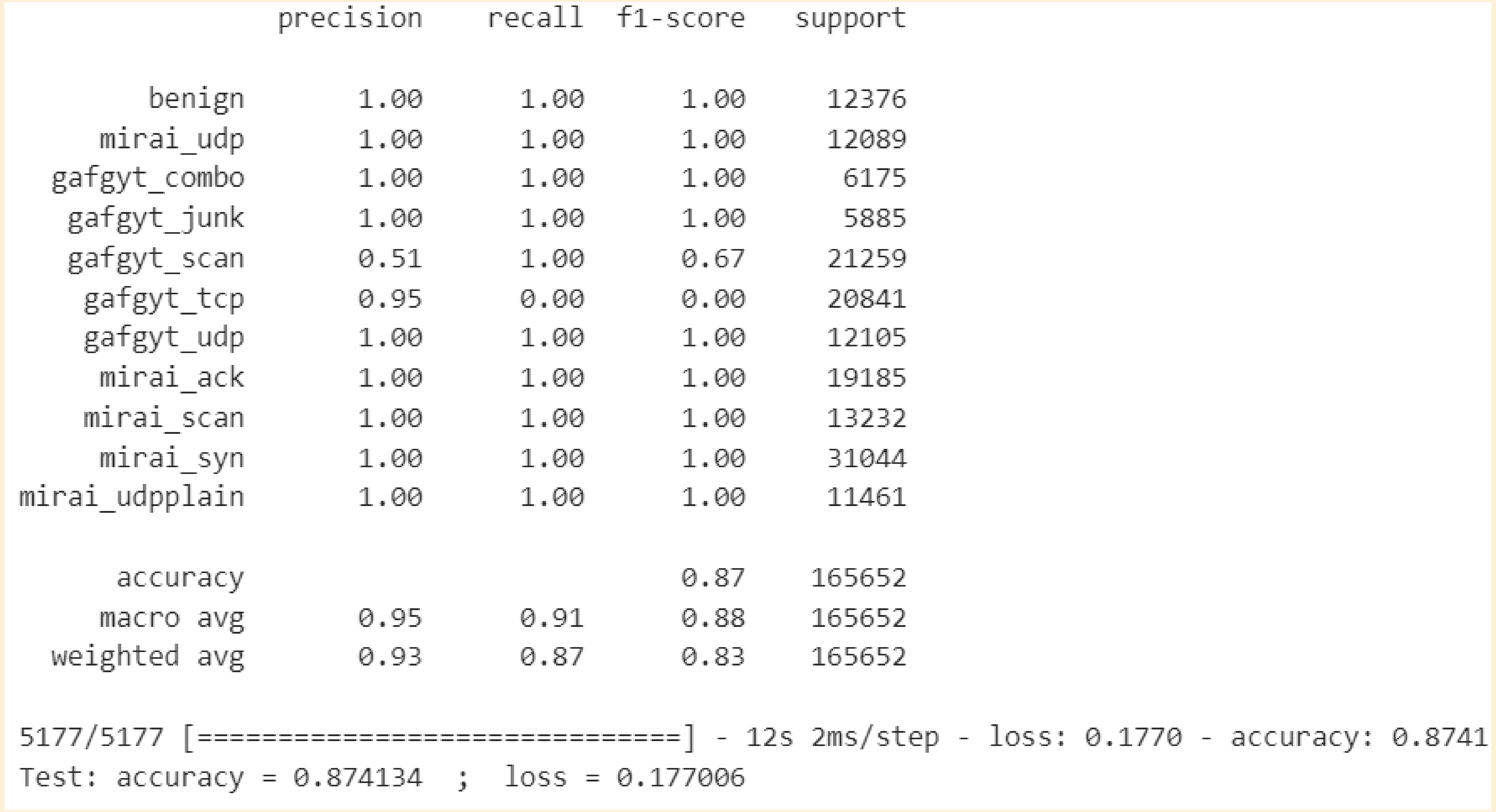
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 115, 64)           384
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 115, 32)           10272
_________________________________________________________________
lstm (LSTM)                  (None, 115, 32)           8320
_________________________________________________________________
lstm_1 (LSTM)                (None, 115, 16)           3136
_________________________________________________________________
flatten (Flatten)            (None, 1840)              0
_________________________________________________________________
dense_5 (Dense)              (None, 128)               235648
_________________________________________________________________
dense_6 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_7 (Dense)              (None, 11)                715
=================================================================
Total params: 266,731
Trainable params: 266,731
Non-trainable params: 0
_________________________________________________________________
```

ARCHITECTURE CODE 4 - DEEP RESIDUAL CNN



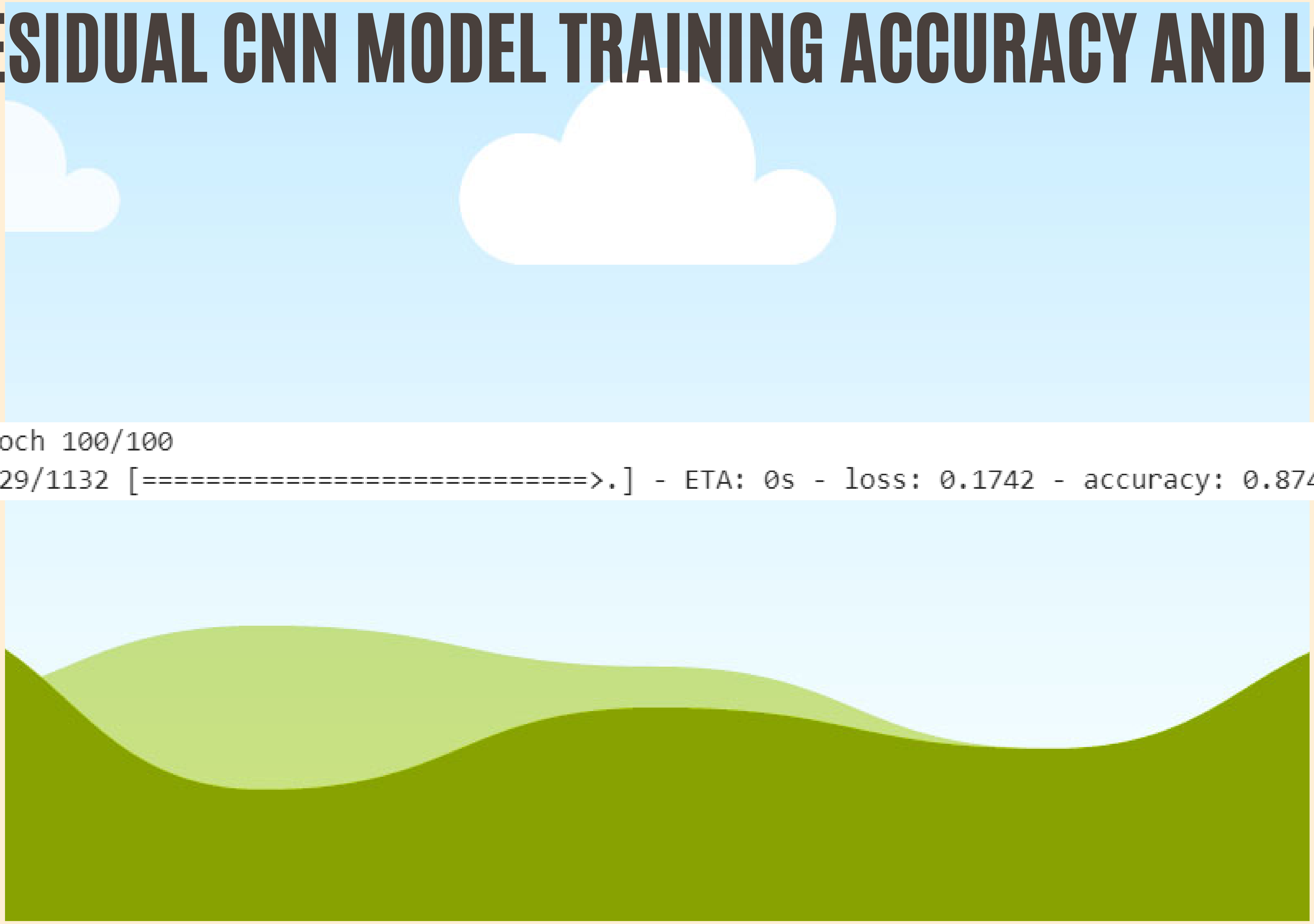
```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 115, 1)]     0

__________________________________________________________________________________________________
conv1d_2 (Conv1D)               (None, 111, 32)      192         input_1[0][0]

__________________________________________________________________________________________________
conv1d_3 (Conv1D)               (None, 111, 32)      5152        conv1d_2[0][0]

__________________________________________________________________________________________________
activation (Activation)         (None, 111, 32)      0           conv1d_3[0][0]

__________________________________________________________________________________________________
conv1d_4 (Conv1D)               (None, 111, 32)      5152        activation[0][0]

__________________________________________________________________________________________________
add (Add)                       (None, 111, 32)      0           conv1d_4[0][0]
                                                                 conv1d_2[0][0]

__________________________________________________________________________________________________
activation_3 (Activation)       (None, 111, 32)      0           add[0][0]

__________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)  (None, 54, 32)       0           activation_3[0][0]

__________________________________________________________________________________________________
conv1d_7 (Conv1D)               (None, 54, 32)       5152        max_pooling1d_1[0][0]

__________________________________________________________________________________________________
activation_4 (Activation)       (None, 54, 32)       0           conv1d_7[0][0]

__________________________________________________________________________________________________
...
Total params: 46,955
Trainable params: 46,955
Non-trainable params: 0

__________________________________________________________________________________________________
```

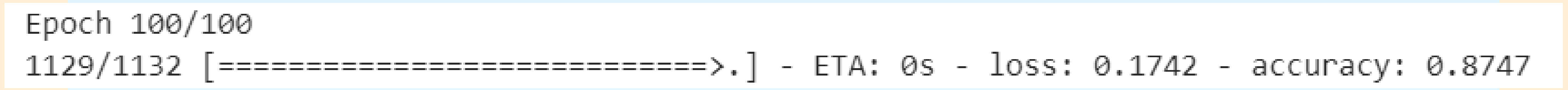
CLASSIFICATION REPORT ON BOTNET DATASET



```
               precision    recall  f1-score   support

       benign       1.00      1.00      1.00     12376
    mirai_udp       1.00      1.00      1.00     12089
 gafgyt_combo       1.00      1.00      1.00      6175
  gafgyt_junk       1.00      1.00      1.00      5885
  gafgyt_scan       0.51      1.00      0.67     21259
   gafgyt_tcp       0.95      0.00      0.00     20841
   gafgyt_udp       1.00      1.00      1.00     12105
    mirai_ack       1.00      1.00      1.00     19185
   mirai_scan       1.00      1.00      1.00     13232
    mirai_syn       1.00      1.00      1.00     31044
mirai_udpplain      1.00      1.00      1.00     11461

     accuracy                           0.87    165652
    macro avg       0.95      0.91      0.88    165652
 weighted avg       0.93      0.87      0.83    165652

5177/5177 [==============================] - 12s 2ms/step - loss: 0.1770 - accuracy: 0.8741
Test: accuracy = 0.874134  ;  loss = 0.177006
```

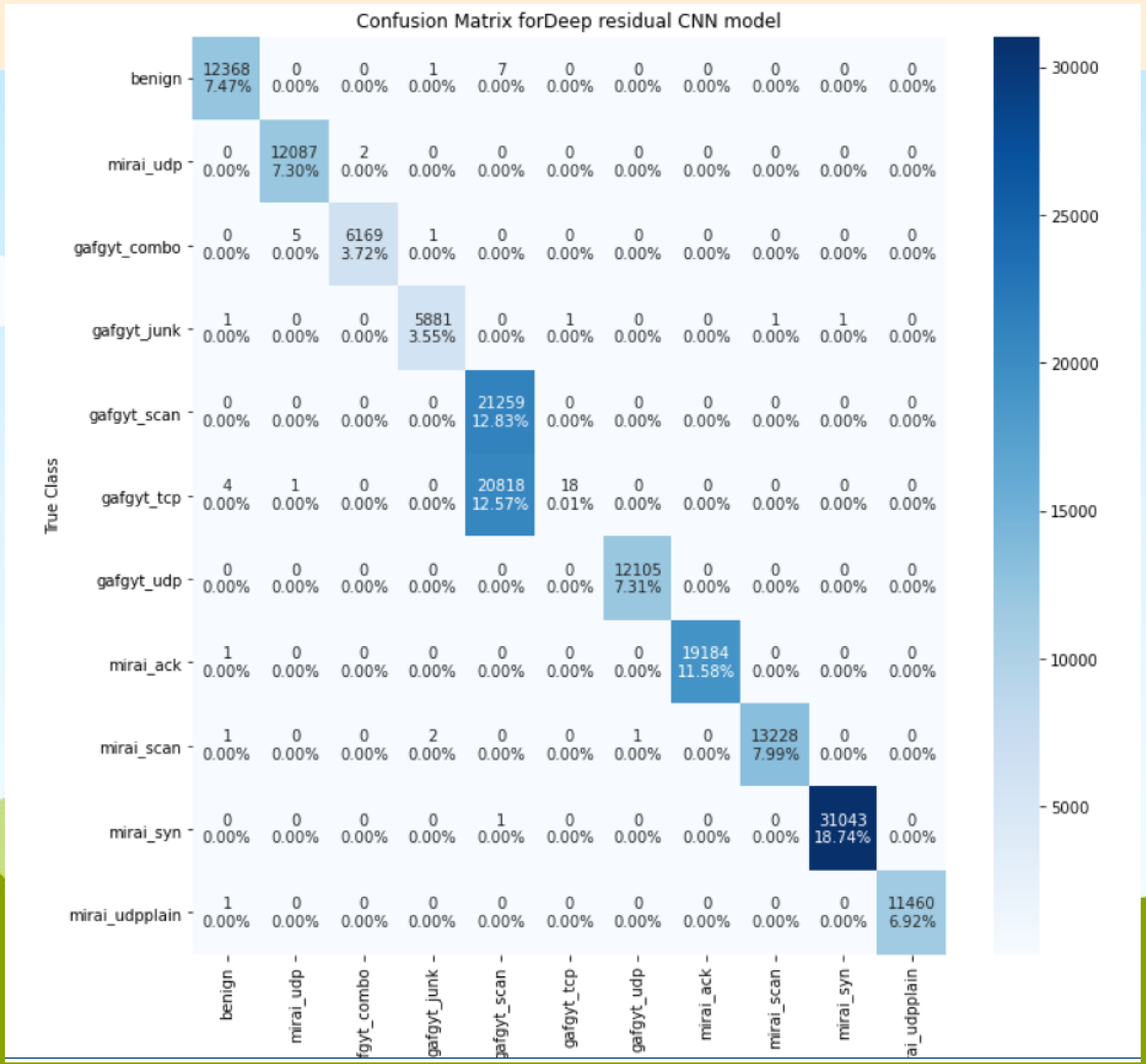
DEEP RESIDUAL CNN MODEL TRAINING ACCURACY AND LOSS EFF





```
Epoch 100/100
1129/1132 [============================>.] - ETA: 0s - loss: 0.1742 - accuracy: 0.8747
```

CONFUSION MATRIX - DEEP RESIDUAL CNN



	benign	mirai_udp	fgyt_combo	gafgyt_junk	gafgyt_scan	gafgyt_tcp	gafgyt_udp	mirai_ack	mirai_scan	mirai_syn	ai_udpplain
benign	12368 7.47%	0 0.00%	0 0.00%	1 0.00%	7 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
mirai_udp	0 0.00%	12087 7.30%	2 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
gafgyt_combo	0 0.00%	5 0.00%	6169 3.72%	1 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
gafgyt_junk	1 0.00%	0 0.00%	0 0.00%	5881 3.55%	0 0.00%	1 0.00%	0 0.00%	0 0.00%	1 0.00%	1 0.00%	0 0.00%
gafgyt_scan	0 0.00%	0 0.00%	0 0.00%	0 0.00%	21259 12.83%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
gafgyt_tcp	4 0.00%	1 0.00%	0 0.00%	0 0.00%	20818 12.57%	18 0.01%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
gafgyt_udp	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	12105 7.31%	0 0.00%	0 0.00%	0 0.00%	0 0.00%
mirai_ack	1 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	19184 11.58%	0 0.00%	0 0.00%	0 0.00%
mirai_scan	1 0.00%	0 0.00%	0 0.00%	2 0.00%	0 0.00%	0 0.00%	1 0.00%	0 0.00%	13228 7.99%	0 0.00%	0 0.00%
mirai_syn	0 0.00%	0 0.00%	0 0.00%	0 0.00%	1 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	31043 18.74%	0 0.00%
mirai_udpplain	1 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	0 0.00%	11460 6.92%

MODULES EXPLAINED

The objective of the Projectcan be split up into the following

1. First data preprocessing is done.
2. Getting the input data.
3. Extracting features from the dataset.
4. Training an anomaly detector.
5. Continuous monitoring for anomaly detection.
6. Finally, we get the detected output.

TEAM MATES

CICIL MELBIN DENNY J	20BAI1284
MOHITH S	20BAI1098
JYOTHI MURUGAN M	20BAI1200
MANIEESH K R	20BAI7083

REFERENCES

- [https://arxiv.org/pdf/1909.11573.pdf
- https://medium.datadriveninvestor.com/5-ai-ml-research-papers-on-deep-fake-detection-you-must-read-1dbf8aba55b8
- https://www.ijert.org/research/review-of-deepfake-detection-techniques-IJERTV10IS050425.pdf
- https://www.theguardian.com/technology/2020/jan/13/what-are-deepfakes-and-how-can-you-spot-them