**Overview**

There are 4 steps:

Model Classification

Model Training

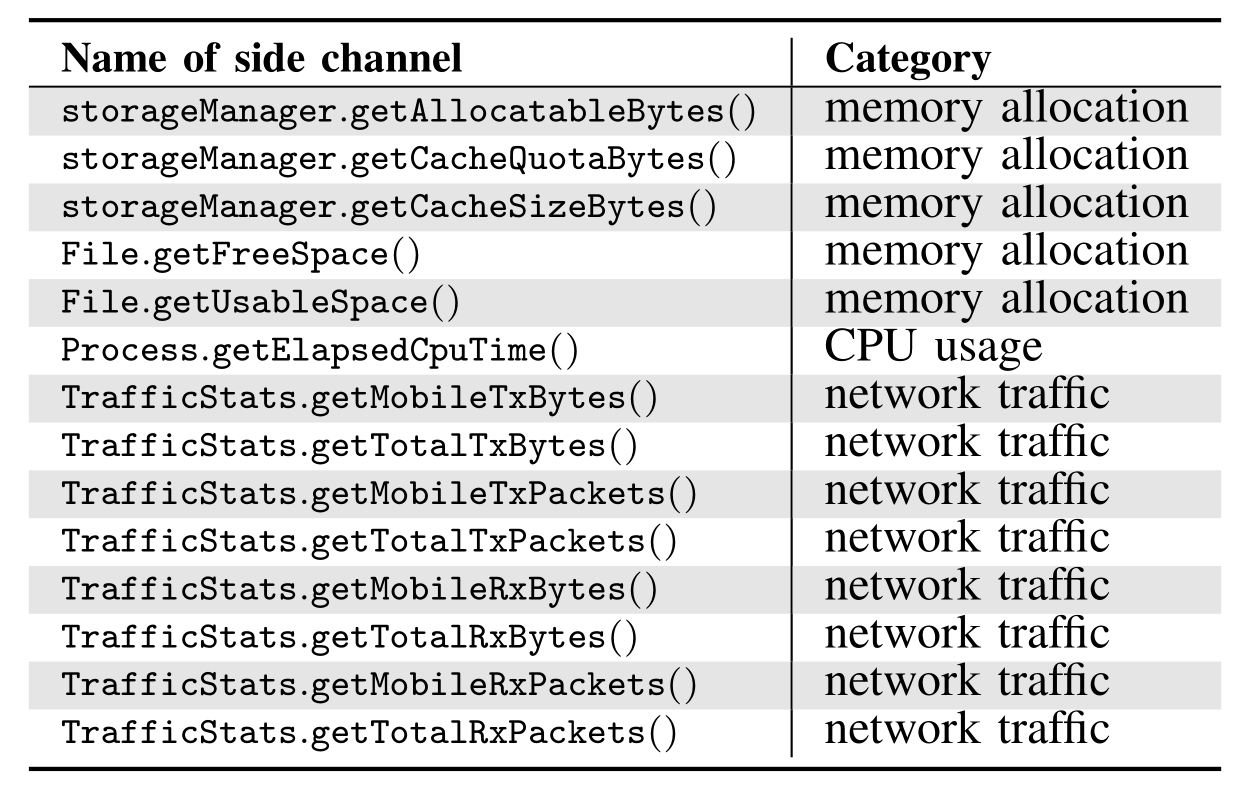
Training data Extraction

Ground Truth Collection

1. **Side-Channel Data and Ground Truth Collection**

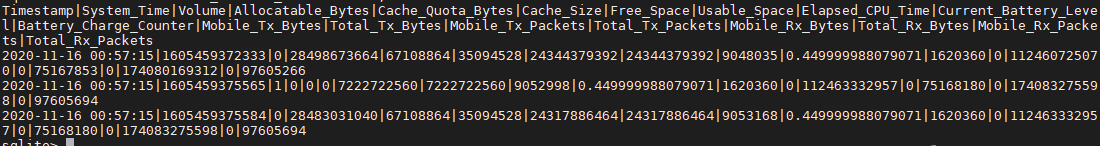
**Description:**

In this stage, we collection 17 types of side-channel values on mobiles. For ground truth, we makes use of *monkeyrunner* to automatically interact with test apps on the simulation devices, and log time stamps of the resulting sensitive behaviors.

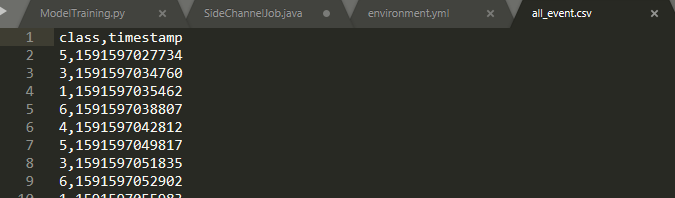


**Output:**

Sqlite database: SideChannelInfo.db



ground truth: all\_event.csv



1. **Extract all events from SideChannelInfo.db according the ground truth file. （process.py）**

**Description:**

In this stage, we extract a data matrix and labels from the side-channel database and ground truth csv for neural network input.

**Input:**

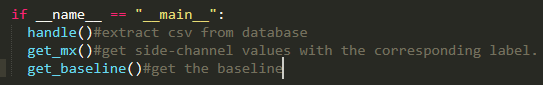
Database: SideChannelInfo.db

GroundTruth: all\_event.csv

**Output:**

Training Data: **multiple\_mx.pkl** (It’s a python pickle file containing SideChannelValues and the corresponding label)

**Function explanation:**



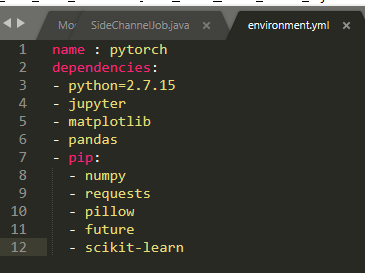
(Baseline means there is not any event happened during this period)

1. **Model Training(ModelTraining.ipynb/ModelTraining.py)**

**Description:**

In this stage, we use the data matrix from stage 2 to train our network.

**Environment.yml:**



And pytorch based on devices.

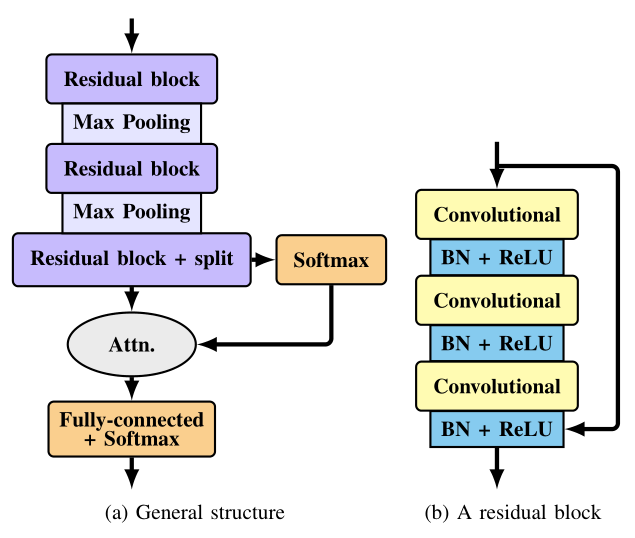
**Input:**

Training Data: multiple\_mx.pkl

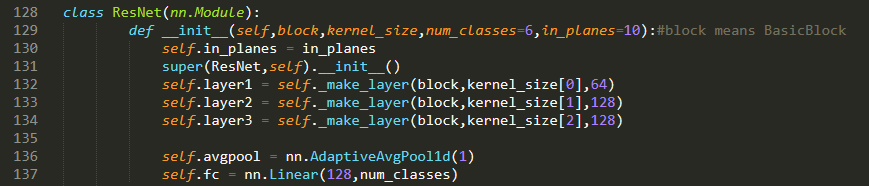
**Output:**

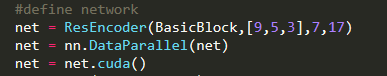
Network: testmodel.pkl

**Model Structure:**



**Model Definition:**



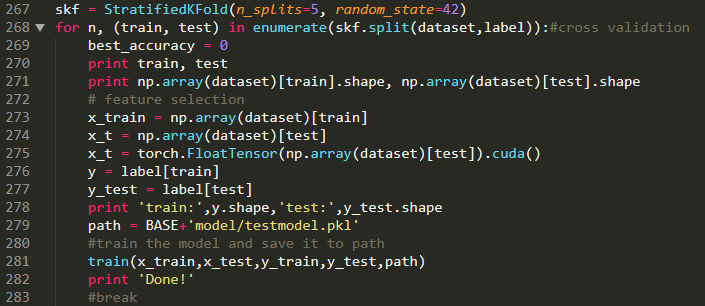


[9,5,3] are the kernel sizes; 7 is the number of classes; 17 correspond the channels.

**Loading the matrix:**



**Training Model:**





1. **Classification(ModelTraining.ipynb/ModelTraining.py/Classifier.java)**

**Description:**

In this stage, we apply the model in the detection.

**For server-end detection(upload sidechannel values to server)**

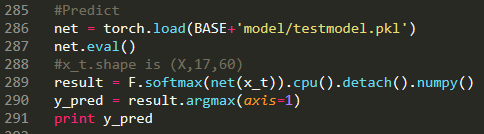
**Input:**

Trained network: testmodel.pkl

Sidechannel values: x\_t( the shape of x\_t must be [X,17,60], X is the number of samples)

**Output:**

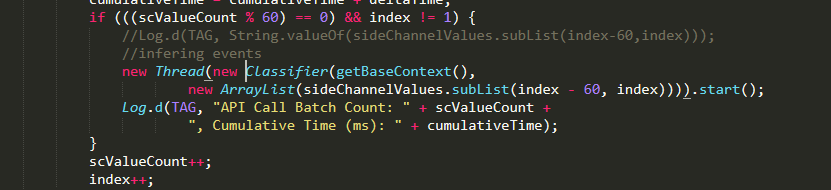
X predicted probability distributions: X \* [class1,class2,class3,class4,class5,class6,class7]

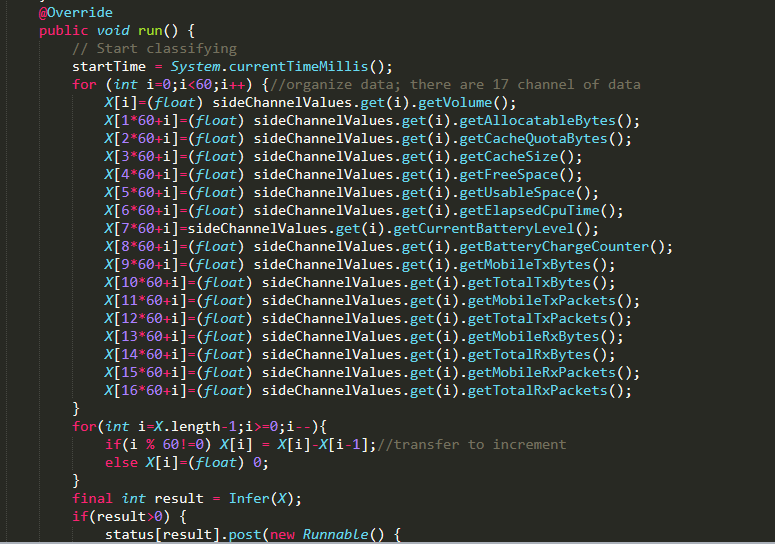


**For Android-end detection(infering on mobiles):**

**Input:**

**A list of 1020 side-channel values(1\*17\*60)**



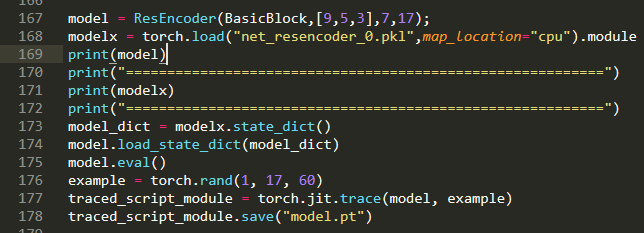


**Output:**

Predicted probability distributions: [class1,class2,class3,class4,class5,class6,class7]

**Explanation for each step:**

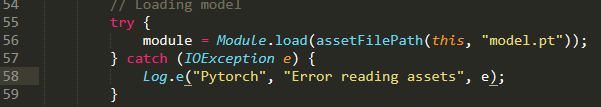
1. Transform a pytorch model to a model for android(transform.py)



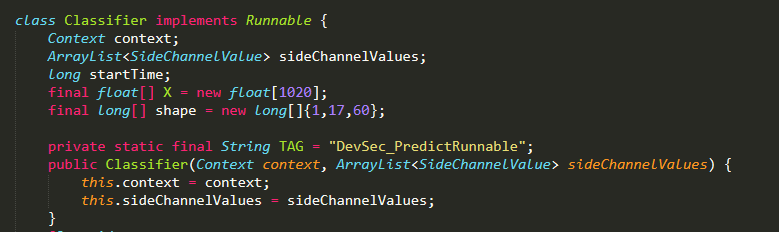
1. **Gradle Implementation:**



1. **Loading model:**



1. **Classifier class:**



1. **use the model to infer the event**

