Crime Audio Event Classification

Final project for *Digital Forensics and Biometrics*
Luca Domeneghetti (mat. 2159079)
University of Padova A.A. 2024/2025

Project description

The project required a Neural Network model to be built and trained in order to perform a **classification** task on audio samples, mostly concerning *crime events*.

In total, the model shall perform classification between 13 audio classes:

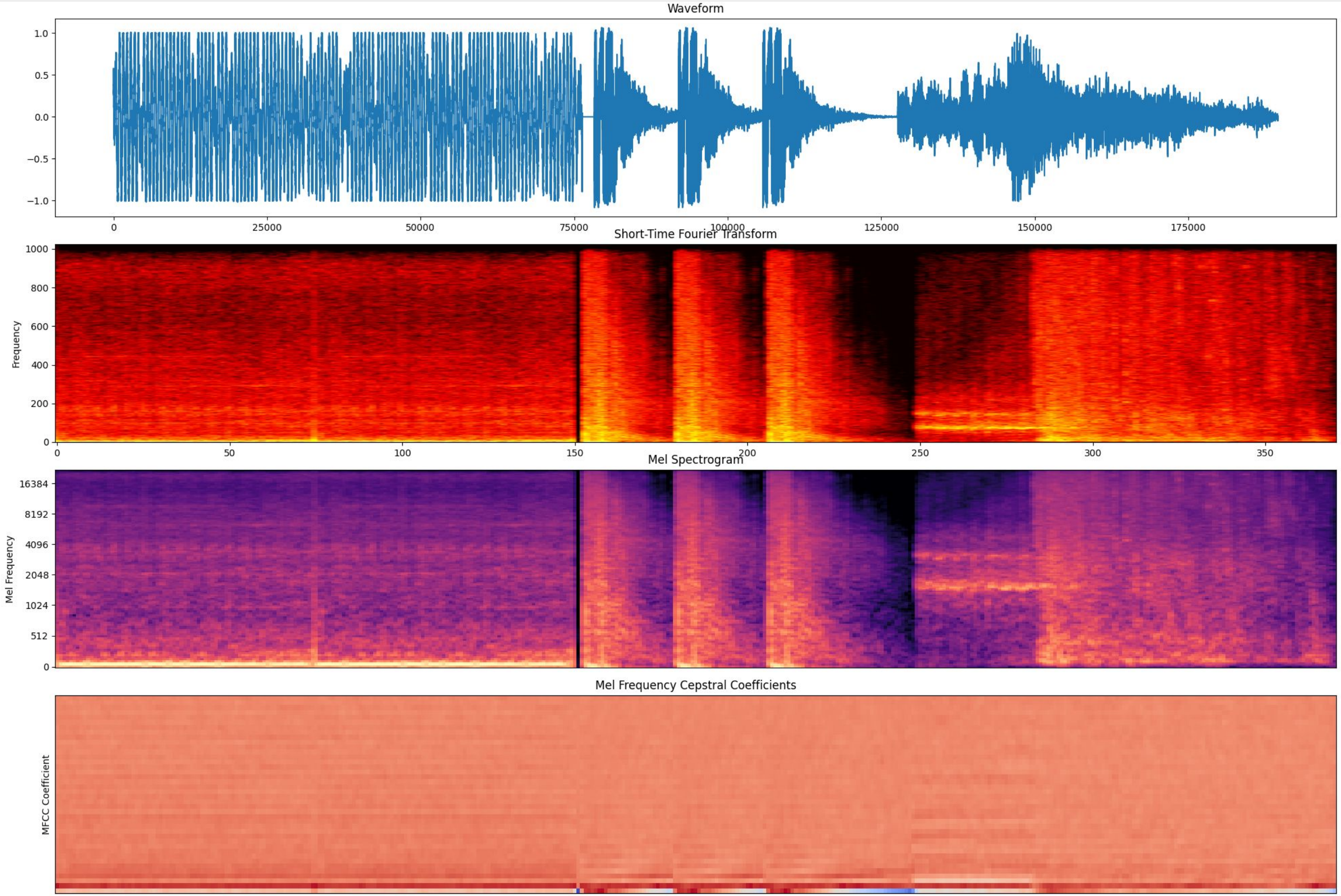
car_crash	conversation	engine_idling	gun_shot
jambret	maling	rain	rampok
road_traffic	scream	thunderstorm	tolong
wind			

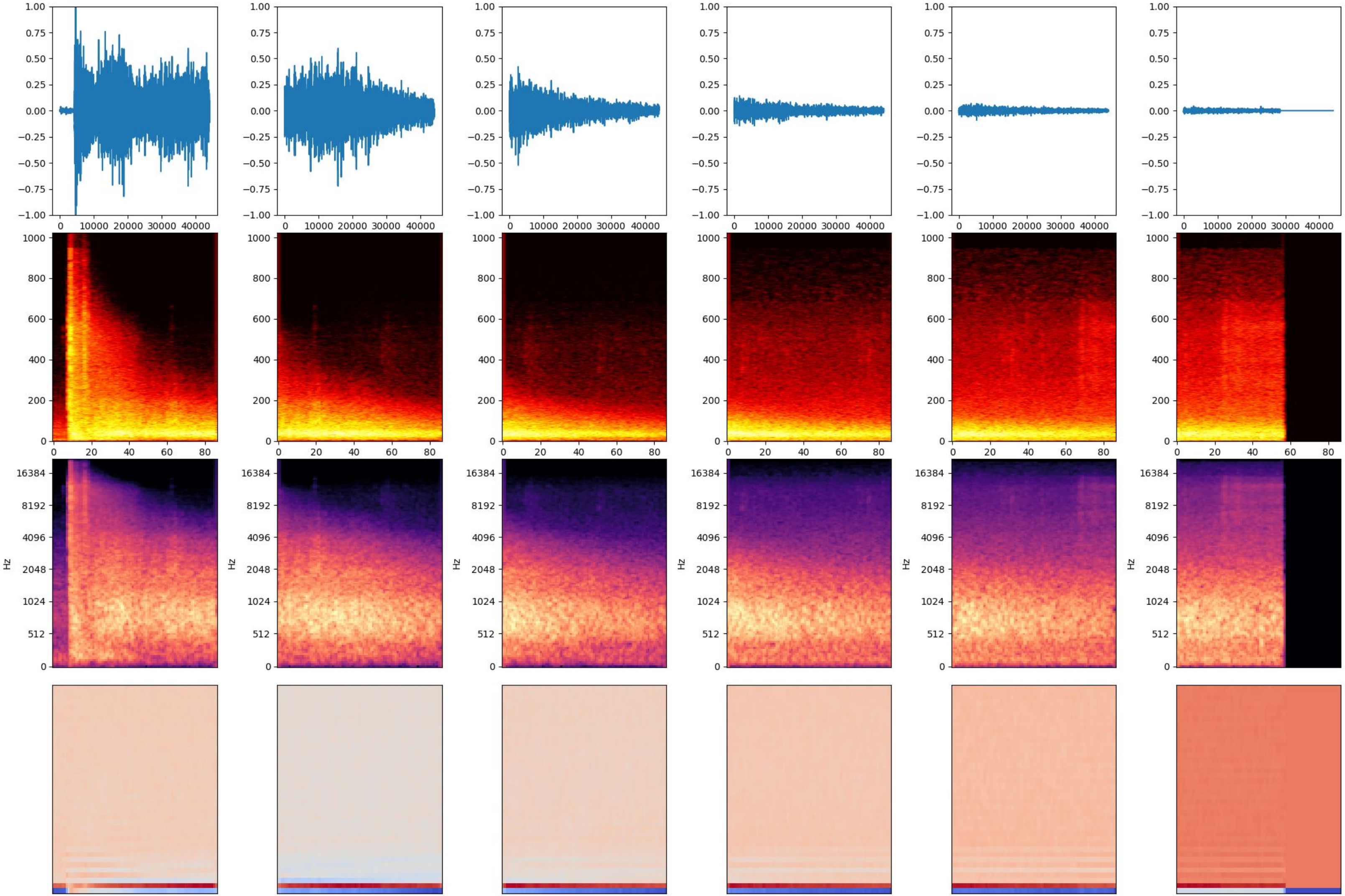
Datasets

- **Raw** audio of accident and crime detection
  - Around 1600 audio samples, each with variable duration
  - https://www.kaggle.com/datasets/afisarsy/raw-audio-of-accident-and-crime-detection
- **Enhanced** audio of accident and crime detection
  - More than 9000 audio samples
  - Generated by combining the raw dataset samples with noise effects (wind, thunderstorm, rain and road traffic)
  - Allows for a better generalization of audio features
  - https://www.kaggle.com/datasets/afisarsy/enhanced-audio-of-accident-and-crime-detection

Audio data preprocessing

- **Chunking** was adopted to split the longer audio files into smaller frames
  - *Partial overlap* to avoid information loss on the frame's boundaries.
- **Mel spectrogram**
  - Operate on the frequency domain instead of the time domain
- **Mel Frequency cepstral coefficients** further enhanced the capability of the model to extract hidden features from data
  - Both the Mel spectrogram and MFCC computations were performed within the LSTM network in order to better exploit the performance of the GPU


Waveform
Short-Time Fourier Transform
Frequency
Mel Spectrogram
Mel Frequency
Mel Frequency Cepstral Coefficients
MFCC Coefficient



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
10000
20000
30000
40000
1000
800
600
400
200
0
20
40
60
80
16384
8192
4096
2048
1024
512
Hz

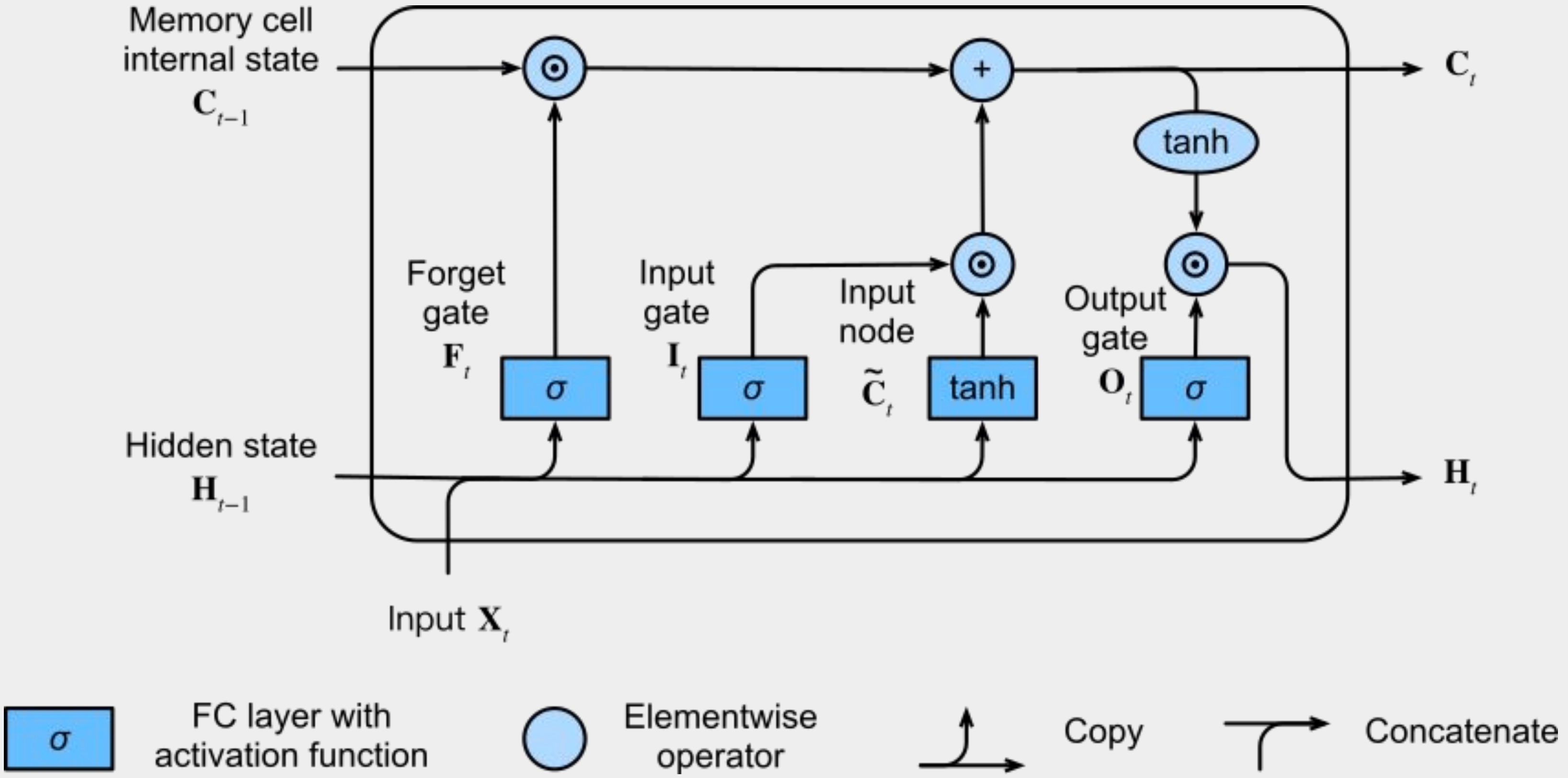
Neural Network and Framework

- Initial approaches involved the use of **Convolutional Neural Networks**:
  - **1D CNN** were used on a flattened vector made out of both the Mel spectrogram and the MFCC combined.
  - **2D CNN** were used to classify spectrograms and MFCC in a “image-recognition” fashion.
  - Although they are easier to understand, CNN failed to properly grasp a fundamental aspect of audio data: time correlation and dependencies.
- For such reasons, it was chosen to abandon CNNs to use some other architectures.

Neural Network and Framework

- The core of the project is a **LSTM RNN**
  - RNN is a good choice when it comes to time-based data.
  - LSTM network is efficient at solving some issues related with the vanishing/exploding of the gradient.
- Initially TensorFlow
  - TensorFlow simpler at handling the dataset and construction of LSTM network, but lacked the required CUDA support.
- Switched to **PyTorch**
  - PyTorch needed additional learning time, but fitted the scope of running the training process entirely on the GPU.

LSTM neuron



Memory cell internal state
$\mathbf{C}_{t-1}$
Forget gate
$\mathbf{F}_t$
Input gate
$\mathbf{I}_t$
Input node
$\tilde{\mathbf{C}}_t$
Output gate
$\mathbf{O}_t$
σ
tanh
+
$\mathbf{C}_t$
Hidden state
$\mathbf{H}_{t-1}$
$\mathbf{H}_t$
Input $\mathbf{X}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

LSTM Network

- Internal **Mel/MFCC** feature computation:
  - 128 Mel bands
  - 40 MFCC coefficients
- Two **bidirectional LSTM** with a single hidden layer of 256 units
- Combine **last two hidden layers** of the forward and backward layers of the LSTM
- Three **dense ReLU layers** with a **dropout** (0.5) in between the first and the last two
- Use of **batch normalization**
- Return **logits** value

Training function

- Loss: **cross entropy** applied on *sparse categorical* predictions against *one-hot encoded labels*
- Optimizer: **AdamW** with *learning rate* = `0.001` and *weight decay* = `1e-4`
- Scheduler: **OneCycleLR** to better fit learning rate to learning conditions

First training attempt

- Use of **raw dataset**
- **Frame size** of 0.5 seconds
- **Hop size** of 0.2 seconds
- **Batches** of 16 audio samples
- 16 **epochs**
- Training on Nvidia GeForce RTX 3060 with CUDA 12.7
- Training duration: ~10 minutes

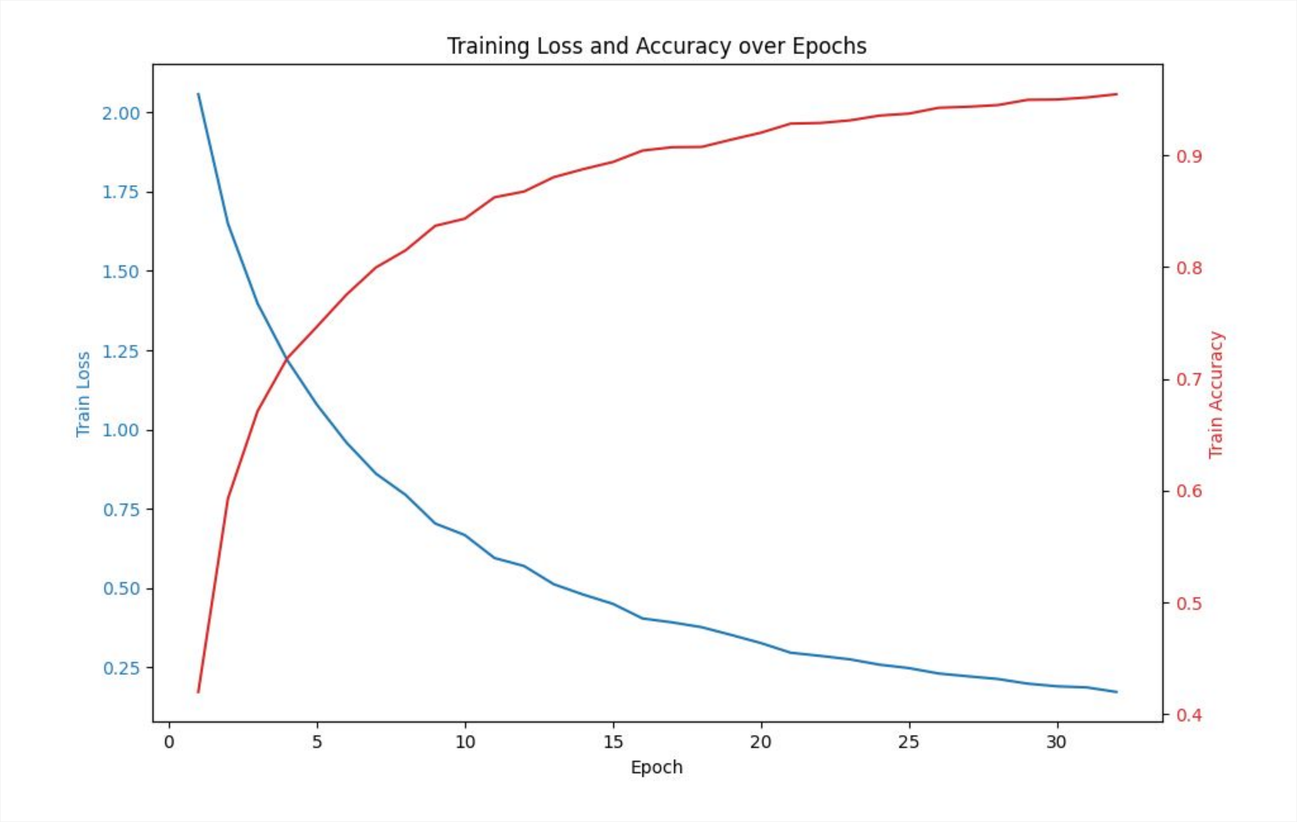
Training accuracy	**Test accuracy**
87%	71%

Final training results

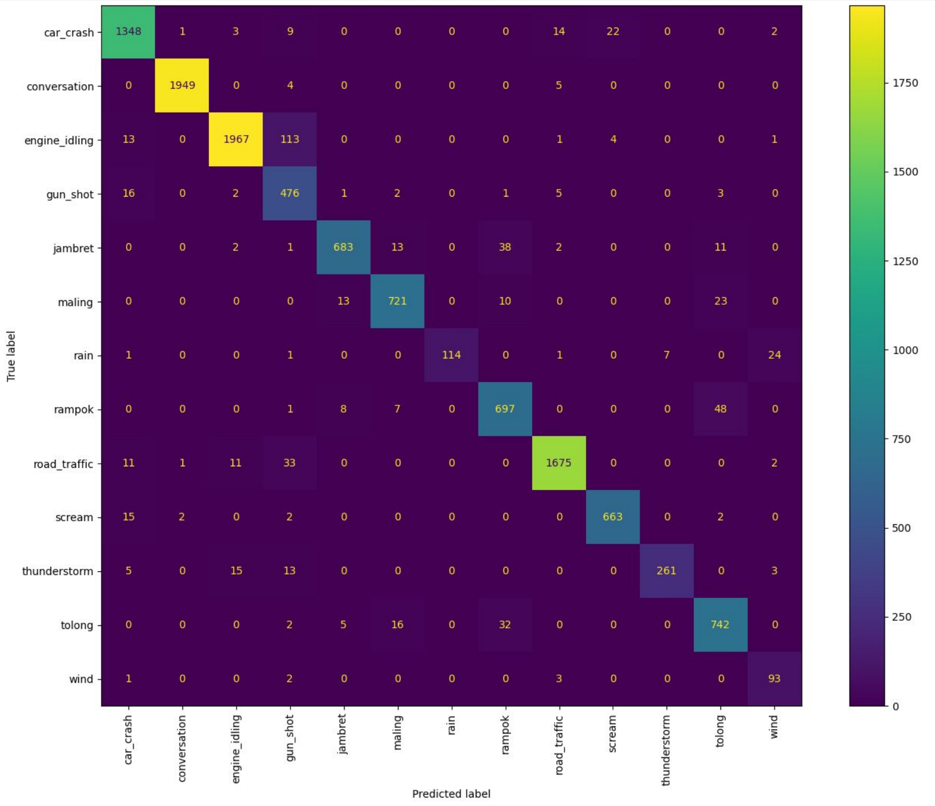
- Use of **enhanced dataset**
- **Frame size** of 0.4 seconds
- **Hop size** of 0.2 seconds
- **Batches** of 32 audio samples
- 32 **epochs**
- Training on Nvidia GeForce RTX 3060 with CUDA 12.7
- Training duration: ~15 minutes

Training accuracy	Test accuracy
95.45%	94.92%

Training graph



Confusion Matrix



Prediction

- Similar fashion to audio pre-processing:
  - Split full sample into smaller windows with partial overlap
  - Perform prediction on each window
  - Collate predictions to form “event segments”
  - Compute start/finish instant in seconds and print
- Samples taken from online sound effect databases
  - samplefocus.com pixabay.com myinstant.com
  - Complex audio samples created by merging together simpler samples using Audacity, resampling to 44.1 kHz, converting to `.wav`

Bibliography

- seth814/Audio-Classification, for the structure of the LSTM network
- Audio Classification Starter, for the use of Datasets and DataLoaders
- Audio Classification with LSTM, for the training and testing function