JETSONAR
SMART SECURITY SYSTEM

PROJECT by
GOSPODIN BODUROV

The story before the Challenge

Thanks to NVIDIA and the GPU Grant of Jetson TX1 for our Autonomous robot capabilities research (ARC) project, we had the opportunity to explore and research for more and different solution for NVIDIA Jetson.

INTRODUCTION

What is "danger" and "threat"? How do they look like? How does the gunshot look? That was hard question for us. But we know the sound of gunshot pretty well. Most of us know what the suspicious noise is in the middle of the night. Most of us know the sound of a car engine, a train, a heavy rain, a breaking glass. So the problem was how is this possible to be used? We combine these thoughts with the acceptation of Artificial Intelligence as the ability of non-biological system to perform tasks normally requiring biological intelligence. Therefore we focused on AI system that uses combination of passive sonar grids paradigm and Deep Learning that detects and recognize different noise types and their locations in real time. Moreover the system can be static or mobile with solar power panels and backup ups unit. Jetson TX1 enables us to look for elegant, simple and powerful tool for security. Hence we called the project Jetsonar. Jetsonar could be used for home security, public security and even ecological issues even, as the system is working in real time, it is lightweight, with a little consumption of electricity and can be independent. Jetsonar has the capacity to detect and localize the sound sources as moving vehicles, noises of human activity, a gust, running water, a storm of hail or even singing birds.

THE PROJECT CONCEPT

In regard of security buildings are really interesting. As real estates they cannot be moved, kidnapped or murdered. This characteristic limits all threats to robbery, vandalism, eavesdropping, natural disaster or terrorism. A perfect AI home security system has to detect all of these threat vectors, but in reality most of commercially available systems can detect only one or two of them. These commercial systems usually use combination of motion detection sensors, video surveillance and alarms over parameter. This is good enough for detection of robbery and vandalism, but what about eavesdropping, natural disasters and terrorism?
Our system consists of passive sonar array and central processing unit for recognition, detection and location. Passive sonar is an acronym for a technique that uses sound propagation and high frequency microphones to detect objects on or under the surface of the water, such as other vessels. Similar concept can be used to record and analyse various noises around buildings. The idea is to place around and inside the building multiple high frequency microphones. These microphones form a grid system as shown on Diagram 1 with which we can determine the position of the sound initiator. We can determine the intensity of the sound for every microphone of the system with which we can calculate the distance and with this information to triangulate and get the approximate position of the sound source.

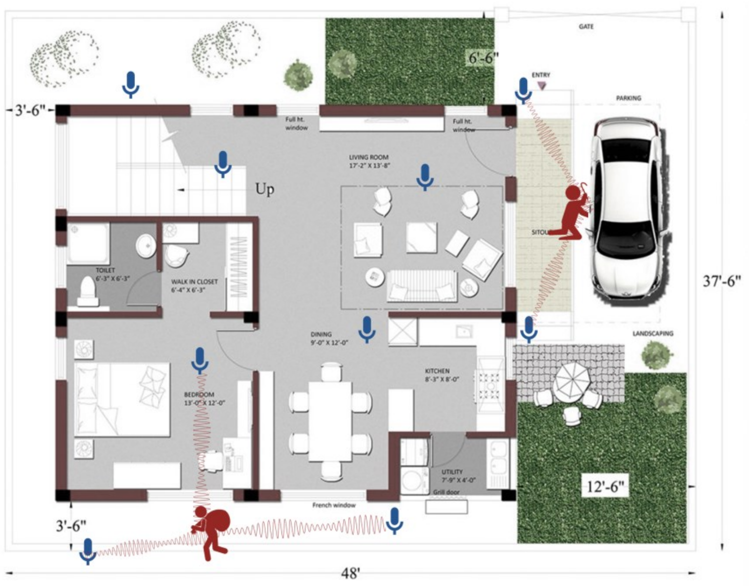


Diagram 1:
Example for microphones grid of Jetsonar for home security

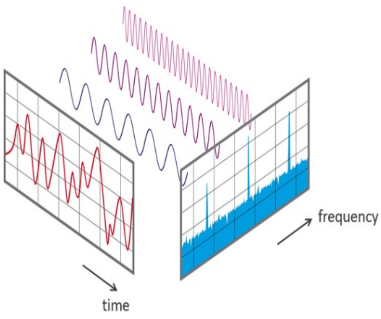
ShotSpotter have a similar solution to proposed one, but their service is strictly limited to gunshot detection and position location. Unfortunately they cannot detect firearm type and their system cannot be trained to recognize, detect and locate another noise types. Actually they have centralized manual validation of the event. Our system on other side is distributed, because recognition, detection and location can be done in place with hardware installed in the building. This means the system can be integrated with multiple layers of building defences as smart locks, smart alarms, and security robots and so on. In the case of smart lock our system can be extended to decide which zones of the building to be locked and command the smart locks to lock the zones. The distributed nature of this system gives us couple of advantages over the centralized ones: failover (our system can use some kind of p2p protocol), local control, data verification (nearby buildings can have the same system installed). Nice bonus feature is that the system can be programmed to initialize a call to 911 with pre-recorded incident report.

Our system can be deployed on scales of neighbourhood, areas, districts and cities, too. If auxiliary power unit is provided our system could run multiple hours in case of power outage.

Our processing unit can be static or mobile. In case of mobile unit, the system can be equipped with solar power panels and backup ups unit. This mean it can be installed in rural areas as farms, protected areas, parks and so on. Another use case is installation in vehicles or on humans similar to Boomerang (countermeasure). This modification transforms the system into mobile passive sonar array. Using p2p communication the system could detect and locate in real time noise types and their locations.

PROJECT DEFINITION

Obviously I cannot develop fully operational system with the given time frame and no learning datasets, but I could develop a working prototype. For this project I decided to use DeepLearning to detect and categorize gunshot sounds from The Firearm Sound Library. This library has sound data for 20 different kinds of firearms. Every firearm is recorded on 3 different distances in wav formats. Project goal is to train neural network that can recognize these firearm types with maximum accuracy. Ok, but DeepLearning is paradigm used for computer vision problems. Fortunately Fourier has invented a way to represent audio signal data in domain nearer to images. The algorithm is named Fast Fourier Transform and it is industry standard for dividing a signal frame into its frequencies representation.



When doing it for the whole audio signal we have multiple frames and can generate something like image frames sequence or video file. Having this nice representation we can now use the standard DeepLearning methods for categorization of images and videos.

After this reasoning I decided to work with the following configuration:

- **Standard Jetson TX1 unit**
- **Standard UPS unit like Eaton or APC**
- **Standard microphone and couple of loudspeakers**
- **A nice pc case with lock.**

The system is not something fancy, but I wanted to have working and fully functional prototype, not something with nice cover and bad functionality.

But could be pretty elegant! ☺

In real case scenario the pc case must be implanted into floor or wall and must be secured with safety lock. The cables must be implanted into walls and microphones must be located under the yard or on some difficult to get locations.

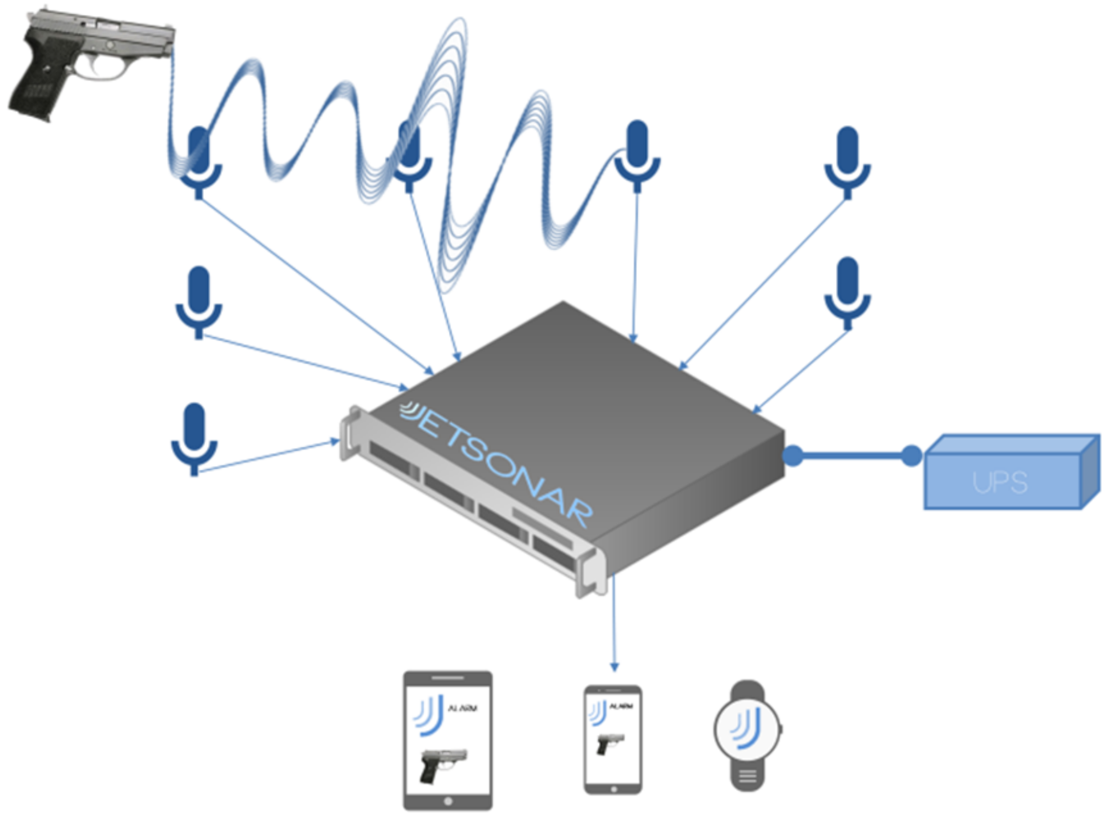


Diagram 2:
Jetsonar hardware diagram: System including microphones, UPS and smart alarm app

And the dataflow diagram of the ready system could be presented as follows:

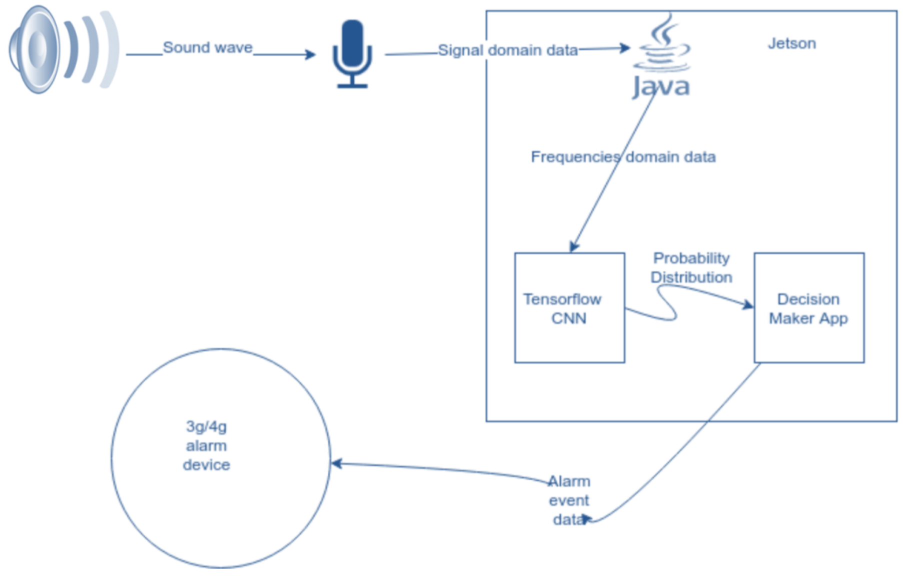


Diagram 3: Jetsonar Dataflow diagram

PROTOTYPE MAKING

After finishing my project definition I had to start work over it. I am stronger with Java than with Python, so I decided to port DeepLearning4j platform on Jetson TX1. Unfortunately it was not possible to use it because of missing libnd4j library for arm64. I had to recompile the whole library with aarch64-linux-gnu-gcc compiler. I installed cuda-cross-aarch64, changed the FindCUDA.cmake file and recompiled the library. After that I started working on dataset preparation. I had to implement Fast Fourier Transform on Java. After some digging and algorithm reading I understand for my case the best solution was to use in place FFT using Cooley-Tukey algorithm. Here I have to mention I totally had skipped the Signal Processing course (it was not mandatory) back in Sofia University so I had to learn the whole theory basis from zero. After that I was fortunate to find Robert Sedgewick and Kevin Wayne wonderful implementation in java of this algorithm. As a proud owner of couple of Sedgewick books in algorithms I decided to use this code for my solution instead of writing my own. Finally I had to implement the whole transform from wav files to mp4 files. I wanted to implement real-time microphone recording and real time transformation from microphone data to video, too. Later I could use this code to detect and categorize gunshots in real time. Speaking of detection I had to implement gunshot detection logic in the project. Initially I wanted to implement this logic with DeepLearning too, but after finding ShotSpotter which was founded 25 years ago, I realized I have to use simpler approach. During ShotSpotter foundation there weren't such powerful video cards. Fortunately if you check the FFT images of gunshot signal you can see that most of the gunshots have a really high contiguous intensity in frequencies between 0hz and 90hz. For better detection and later categorization I used median filter algorithm to reduce intensity noise. As result I implemented a simple scaling window algorithm that was enough for gunshot detection. I tested the logic with some real gunshot

YouTube videos and it gave very small false positives count. In case of false positive it was difficult for me to decide whether the sound is gunshot or not.

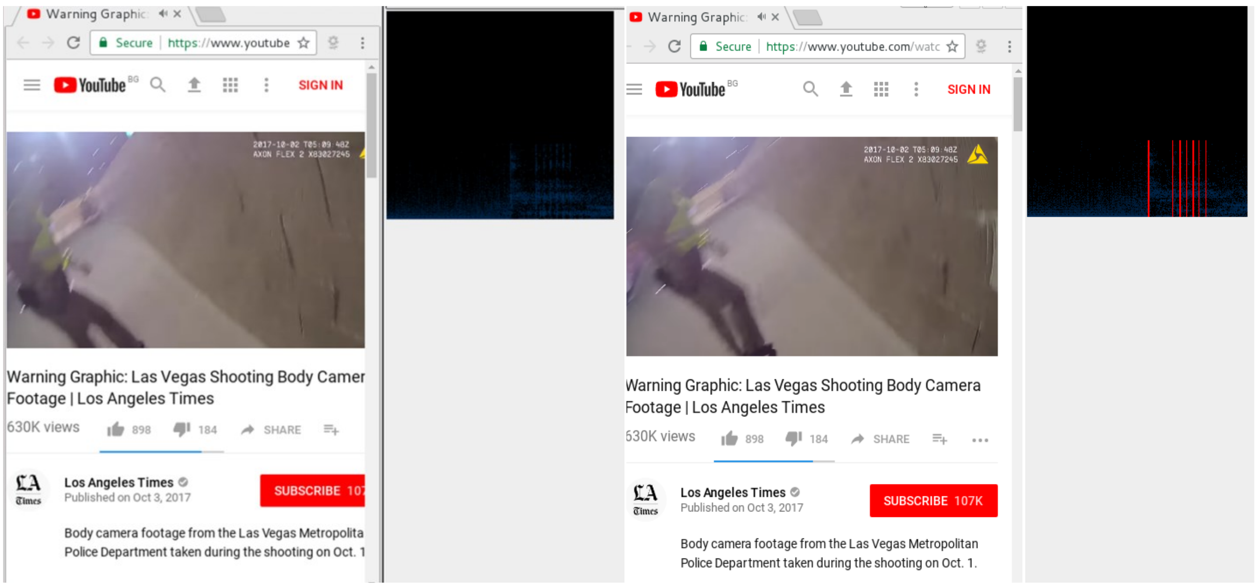
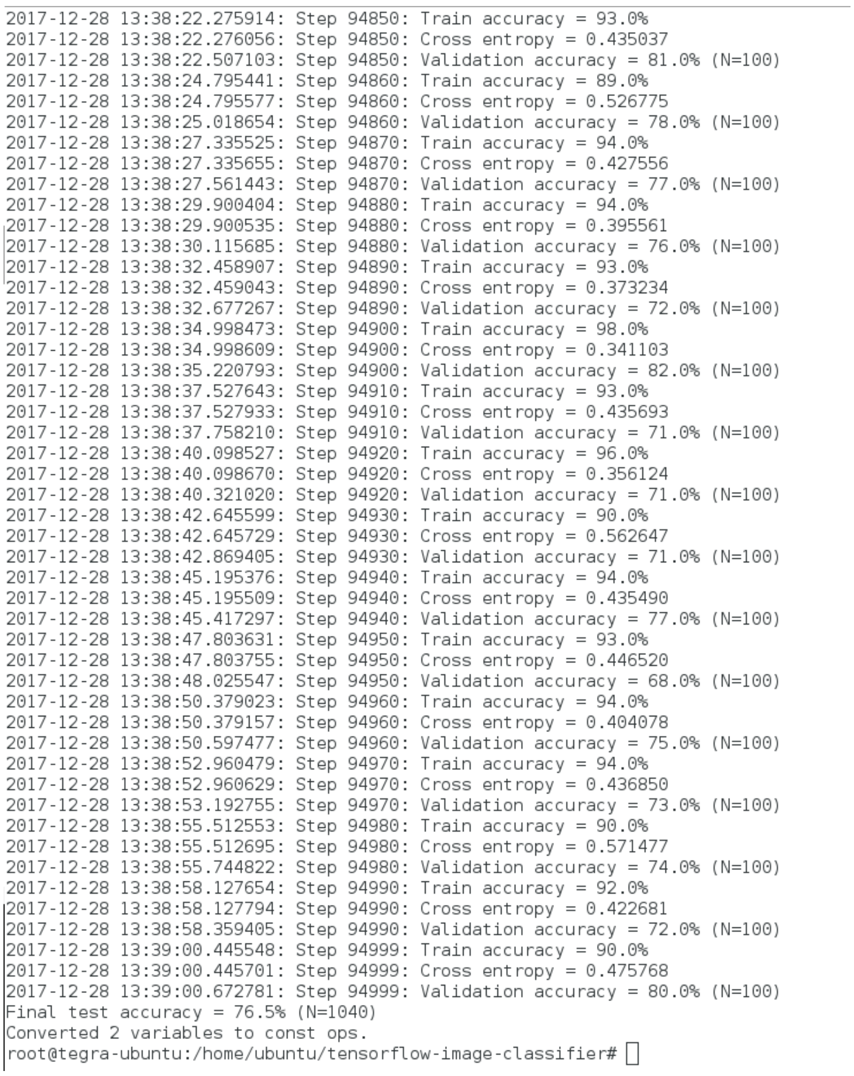

Warning Graphic
Secure
https://www.youtube
YouTube
SIGN IN
2017-10-02 T05:09:48Z AXON FLEX 2 X83027245
Warning Graphic: Las Vegas Shooting Body Camera Footage | Los Angeles Times
630K views
898
184
SHARE
Los Angeles Times
Published on Oct 3, 2017
SUBSCRIBE 107K
Body camera footage from the Las Vegas Metropolitan Police Department taken during the shooting on Oct. 1.

Image 1: Screenshots of tests of Jestsonar

After finishing all this logic I returned to categorization problem. And here I got the real problem. During my experiments with FFT and signal processing, Deeplearning4j team had changed the whole building logic of their application and I couldn't compile their library. As a result of this I had to choice whether to rewrite the whole compilation logic again or to switch to some more compatible with Jetson platform. My solution was to train my deeplearning4j model (it was already coded) using cpu on a Xeon Server with 12 cores. Meanwhile I researched couple of Tensorflow Cuda models. The deeplearning4j model couldn't reach more than 40% accuracy on test and validation data. For tensorflow ones I tested multiple models including sound cnn, modification of mnist model, and a combination between dnn and rnn networks. Unfortunately all of the models couldn't train for FFT frame data. I tried to modify them, but Jetson has only 4GB of video ram and most of my experiments finished with cuda not enough video memory error. As a result I couldn't manage to get better than 60% accuracy on test and validation data using these models. Sound-Cnn requires special attention, because it seems to solve the problem completely. It has similar problem definition. However it uses 64x64 images for recognition, works on only 1-3 second of audio data and FFT implementation is really slow under python, so I couldn't use it for real time. And last but not the least the model couldn't get more than 33% accuracy on test and validation data. With all this failures it looked like training a model from scratch is not a good idea. I decided to change the tactic and use Inception-v3. Tensor flow allows users to get latest Inception model and retrain the fully connected neural layers with their own images and labels. With this algorithm I managed to get approximately 80% accuracy on test and validation data, which is good enough for prototype system.



```
2017-12-28 13:38:22.275914: Step 94850: Train accuracy = 93.0%
2017-12-28 13:38:22.276056: Step 94850: Cross entropy = 0.435037
2017-12-28 13:38:22.507103: Step 94850: Validation accuracy = 81.0% (N=100)
2017-12-28 13:38:24.795441: Step 94860: Train accuracy = 89.0%
2017-12-28 13:38:24.795577: Step 94860: Cross entropy = 0.526775
2017-12-28 13:38:25.018654: Step 94860: Validation accuracy = 78.0% (N=100)
2017-12-28 13:38:27.335525: Step 94870: Train accuracy = 94.0%
2017-12-28 13:38:27.335655: Step 94870: Cross entropy = 0.427556
2017-12-28 13:38:27.561443: Step 94870: Validation accuracy = 77.0% (N=100)
2017-12-28 13:38:29.900404: Step 94880: Train accuracy = 94.0%
2017-12-28 13:38:29.900535: Step 94880: Cross entropy = 0.395561
2017-12-28 13:38:30.115685: Step 94880: Validation accuracy = 76.0% (N=100)
2017-12-28 13:38:32.458907: Step 94890: Train accuracy = 93.0%
2017-12-28 13:38:32.459043: Step 94890: Cross entropy = 0.373234
2017-12-28 13:38:32.677267: Step 94890: Validation accuracy = 72.0% (N=100)
2017-12-28 13:38:34.998473: Step 94900: Train accuracy = 98.0%
2017-12-28 13:38:34.998609: Step 94900: Cross entropy = 0.341103
2017-12-28 13:38:35.220793: Step 94900: Validation accuracy = 82.0% (N=100)
2017-12-28 13:38:37.527643: Step 94910: Train accuracy = 93.0%
2017-12-28 13:38:37.527933: Step 94910: Cross entropy = 0.435693
2017-12-28 13:38:37.758210: Step 94910: Validation accuracy = 71.0% (N=100)
2017-12-28 13:38:40.098527: Step 94920: Train accuracy = 96.0%
2017-12-28 13:38:40.098670: Step 94920: Cross entropy = 0.356124
2017-12-28 13:38:40.321020: Step 94920: Validation accuracy = 71.0% (N=100)
2017-12-28 13:38:42.645599: Step 94930: Train accuracy = 90.0%
2017-12-28 13:38:42.645729: Step 94930: Cross entropy = 0.562647
2017-12-28 13:38:42.869405: Step 94930: Validation accuracy = 71.0% (N=100)
2017-12-28 13:38:45.195376: Step 94940: Train accuracy = 94.0%
2017-12-28 13:38:45.195509: Step 94940: Cross entropy = 0.435490
2017-12-28 13:38:45.417297: Step 94940: Validation accuracy = 77.0% (N=100)
2017-12-28 13:38:47.803631: Step 94950: Train accuracy = 93.0%
2017-12-28 13:38:47.803755: Step 94950: Cross entropy = 0.446520
2017-12-28 13:38:48.025547: Step 94950: Validation accuracy = 68.0% (N=100)
2017-12-28 13:38:50.379023: Step 94960: Train accuracy = 94.0%
2017-12-28 13:38:50.379157: Step 94960: Cross entropy = 0.404078
2017-12-28 13:38:50.597477: Step 94960: Validation accuracy = 75.0% (N=100)
2017-12-28 13:38:52.960479: Step 94970: Train accuracy = 94.0%
2017-12-28 13:38:52.960629: Step 94970: Cross entropy = 0.436850
2017-12-28 13:38:53.192755: Step 94970: Validation accuracy = 73.0% (N=100)
2017-12-28 13:38:55.512553: Step 94980: Train accuracy = 90.0%
2017-12-28 13:38:55.512695: Step 94980: Cross entropy = 0.571477
2017-12-28 13:38:55.744822: Step 94980: Validation accuracy = 74.0% (N=100)
2017-12-28 13:38:58.127654: Step 94990: Train accuracy = 92.0%
2017-12-28 13:38:58.127794: Step 94990: Cross entropy = 0.422681
2017-12-28 13:38:58.359405: Step 94990: Validation accuracy = 72.0% (N=100)
2017-12-28 13:39:00.445548: Step 94999: Train accuracy = 90.0%
2017-12-28 13:39:00.445701: Step 94999: Cross entropy = 0.475768
2017-12-28 13:39:00.672781: Step 94999: Validation accuracy = 80.0% (N=100)
Final test accuracy = 76.5% (N=1040)
Converted 2 variables to const ops.
root@tegra-ubuntu:/home/ubuntu/tensorflow-image-classifier#
```

Image 2: Accuracy test screenshot

I had to convert FFT mp4 data to image frames. One important step for doing this is to decide which frames from the video can be labeled as particular firearm gunshot and which as uncategorized noise. I did this by measuring the total sum of intensity in the frame. With this logic and the fact that most of the training data has multiple gunshots recorded in one file, it was easy to determine which parts of the original wav file are gunshots and which are empty sound. Another problem was some of the gunshots have tail sound; I had to categorize frames with only such tail sounds as a noise too.

Next step will be to attach Inception's retrained model to RNN, in this case I could categorize video sequences instead of just simple frames.

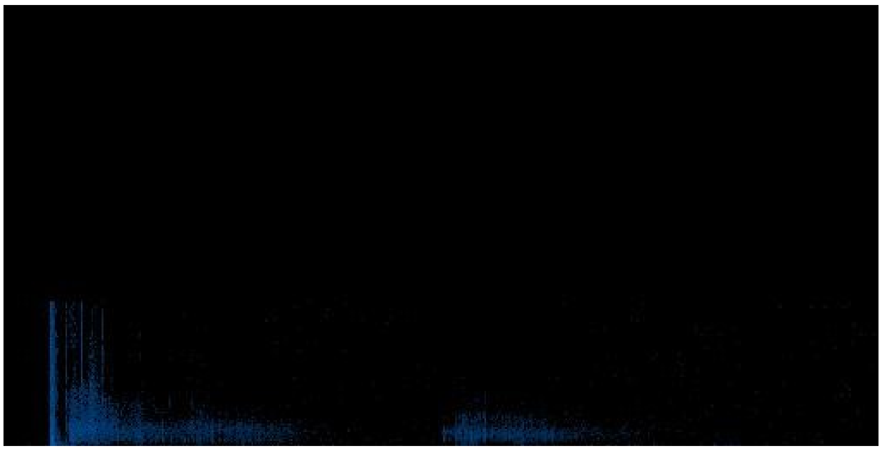


Image 3: Converted mp4 data into image frames

CONCUSION

Jetsonar is a conception that combines small and power-efficient as hardware and powerful software as security solution. However Jetsonar has great potential for wide impact not only for security purposes but as solution for many problems and some of them are proposed in the Table 1

Home and Business security

- Jetsonar static sonar array connected to multiple microphones placed into and around a house for detection, recognition and localization of atypical sounds including: human, vehicle or animals movement, vandalism, robbery, rainfalls or strong winds
- Multiple Jetsonar units can secure a whole distinct or residential area
- Jetsonar system can be integrated with third-party hardware and software to send notifications, activate alarms, make automatic emergency call or text
- In additional Jetsonar as home or office security system can be integrated with automatic smart home systems

Civil security

- Jetsonar can detect crimes or terroristic acts as shooting or bombing, or causing panic screams
- Static sonar array of Jetsonar system can be deployed as Natural disaster detector recognizing floodings, earthquakes, tsunamis, avalanches and similar sound high frequencies disasters

Military

- A number of soldiers in a squad can be equiped with Jetson on 18650 battery pack with connected microphones. Total weight of the unit will be less than average notebook weight and can be used as personal security system for soldiers and mercenaries

Public order

- Jetsonar system can monitors the noise levels as part of the public order in many cities and areas
- Jetsonar system can be deployed to detect, recognize and localize crowds of people and different kinds of atypical for unorganized crowd sounds

Law Enforcement

- Jetsonar can be installed under highways to detect whether truck drivers adhere to the rules and do not enter the highway during hot days and days with high traffic.

Protection of the environment

- Static array of Jetsonar machines can be deployed in protected rural areas and control of poachers during mating season of protected species
- In hunting areas Jetsonar can be deployed to control if the hunters use only allowed weapons
- Static array of Jetsonar machines can be deployed to monitor species migration periods and movings
- Static array of Jetsonar machines can be deployed to explore populations of different species

Table 1: Jetsonar use cases examples

Jetsonar is a project of smart security system, but thanks to Jetson and AI tools that use combination of passive sonar grids paradigm and DeepLearning can give us an opportunity to explore the world of sounds by simple as hardware but powerful manner for home and business security, civil security, ecological purposes and more.

Just listen!

JETSONAR
SMART SECURITY SYSTEM

JETSONAR SMART SECURITY SYSTEM

PROJECT PROPOSED BY

GOSPODIN BODUROV

PROJECT, DOCUMENTATION AND VISUALIZATION EDITOR
GERGANA MARKOVA

For NVIDIA® JETSON™ DEVELOPER CHALLENGE
2018