NVIDIA Jetson AI Certification Project

Outdoor Cat-A-Logger

Deep Learning on an NVIDIA Jetson Nano Keeps Track of Outdoor Cats Health, Movements and Reproductive Status

Purpose

Outdoor cats are present nearly everywhere in the world. Some regions allow them to roam freely, some want them vaccinated, and others want them to be spayed or neutered. Outdoor cats are notoriously difficult to track and manage without specialized equipment, personnel, training, and procedures.

This project proposes a solution using Deep Learning to keep track of outdoor cats. It can distinguish cats from other objects and ultimately allow the user to track metadata on individual cats.

Metadata can include but is not limited to: The coming and goings of a particular cat over time, their vaccination, health, and reproductive status.

The Problem

Deep Learning can be used to distinguish cats from other critters, objects, and noise. A model can also be trained to recognize individual cats, but **there are no publicly available models of cats taken with a night vision camera nor are their publicly available images with which to train such a model.**

Images need to be generated, found, collected, or otherwise captured in order to train the model.

The initial model was bootstrapped with a highly augmented dataset and built up over time using real captured video to retrain the model. The Jupyter Notebooks and Datasets are publicly available on Kaggle.

Today’s State of the Art Motion Detection

The initial thought was to gather images using the Surveillance Cameras built-in motion detection to collect images of cats during the day and at night.

Retail Motion Detection Cameras are virtually useless for this because they tend to trigger on almost anything.

They are getting better at detecting people which makes them more useful, but that does not help with our cat imaging problem.

Another solution was needed.

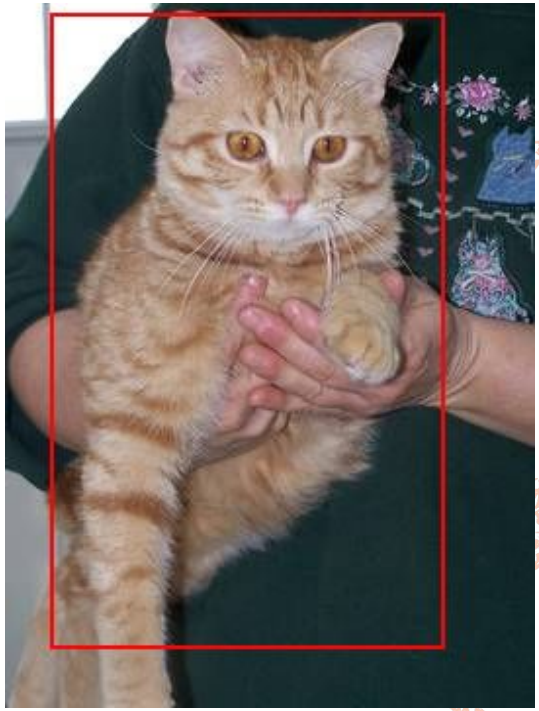
The Solution - Synthetic Data

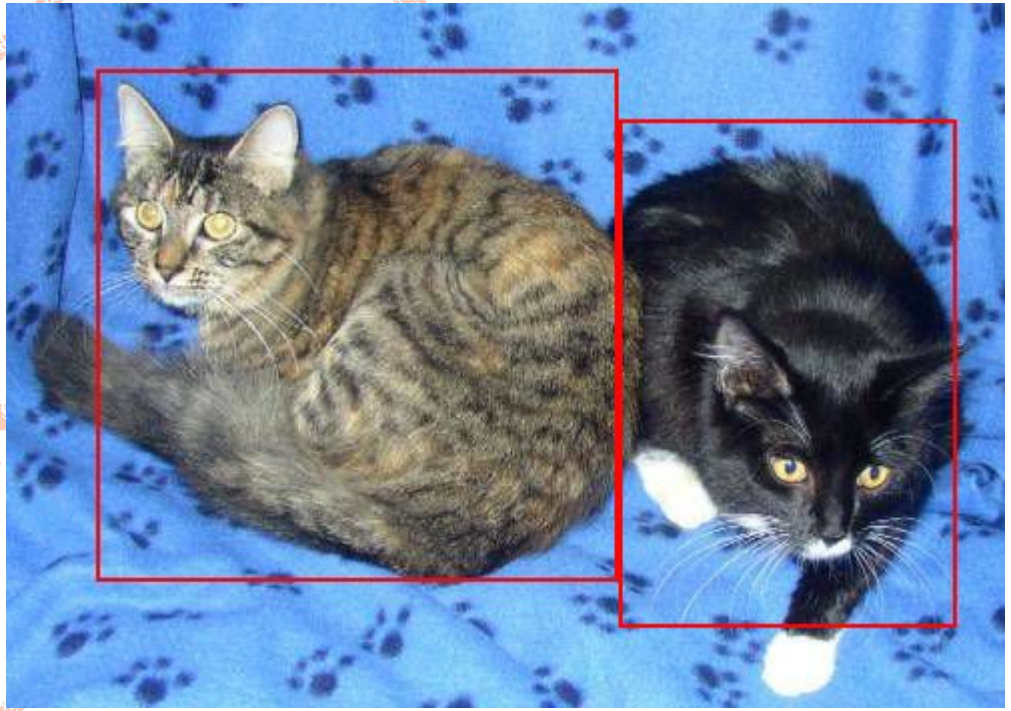
Build a model from publicly available images of cats and attempt to simulate images taken by a Night Vision Camera hoping to “trick” the initial model into recognizing images of cats at night. The original dataset was created as follows:

- Crop publicly available daylight images extracting only cats
- Convert the images to black and white
- Invert the colors to make them negative images
- Enhance the images by randomly sizing, skewing and adding noise

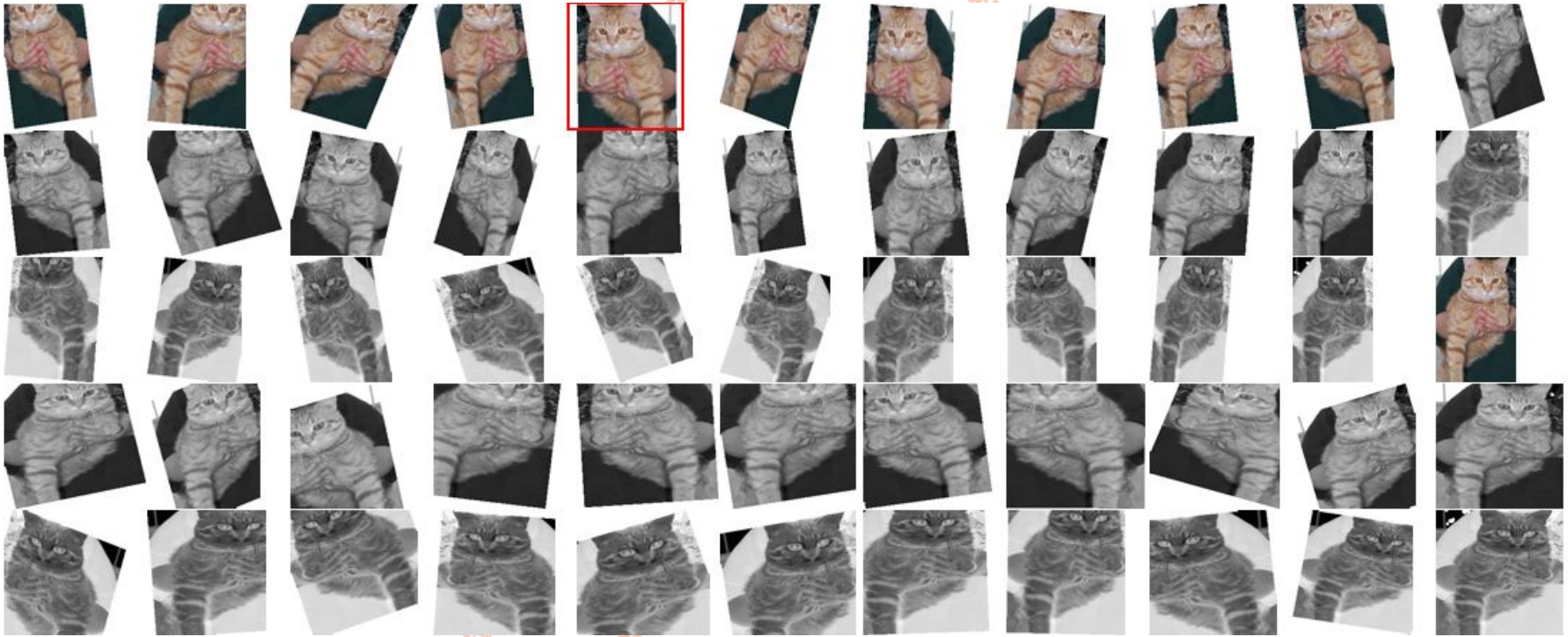
To make a larger more robust dataset, the entire set of source, intermediate, and final images were randomly paired with random backgrounds which were randomly augmented. The program also created metadata such as bounding boxes and labels to inform the model where the cats are in the composited images. The final images were used to train the model. This algorithm currently produces 55 unique output images per input image.

Extract Cats from Dataset Using YOLOv3





Augment Images



Composite Images and Emit Bounding Boxes

Negate and Grayscale Some of Them

Bootstrapping the Initial Model

- Used the previously created composite images along with their bounding boxes and metadata to train an initial model
- Tested the model against images taken with a night vision camera such as a Deer Cam or whatever was available
- Manually went through each detected image and approve or reject it
- Manually went through each image rejected by the model and create metadata if it is a positive hit which originally went unrecognized by the model
- Trained the model again using all of the collected images
- Iterated these steps until sufficient images were collected and the desired level of inference accuracy achieved

Model Choice (TBD) Need Accuracy Numbers

[Jetson Nano Benchmarks]

Model Name	FPS
inception_v4	10.6
vgg19_N2	10.1
super_resolution_bsd500	15.3
unet-segmentation	16.8
pose_estimation	14.6
yolov3-tiny-416	47.9
ResNet50_224x224	36.9
ssd-mobilenet-v1	42.7

The Cat Box

To capture images of cats, a unit was built to house a night vision capable surveillance camera and a Wifi-based monitoring system. The cats were baited in order to get them to come close to the Cat Box for imaging.

The unit is portable, battery powered, wireless, and can be moved to various locations as needed. Currently, the system requires Wifi but could support embedded cellular or satellite communications. The initial unit is rather large and power hungry but could be significantly miniaturized and made to be completely solar powered for remote operation.

To reduce bandwidth for cellular or satellite communications and AI Inference Device could be placed inside of the Cat Box so it would only transmit pertinent data.

Cat Box BOM

Outdoor Wifi Surveillance Camera

Monitor & Trigger

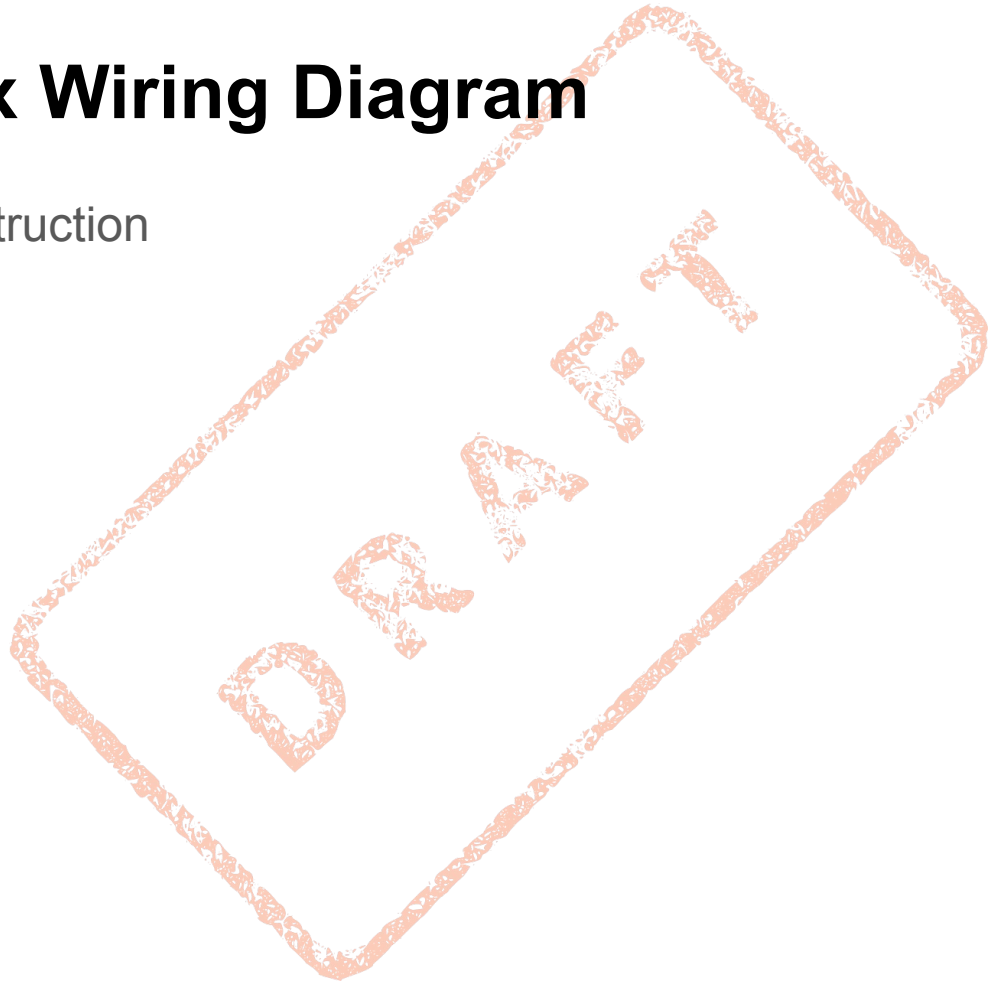
- ESP32 - Read Sensors, Process and Communicate Status Data via Wifi
- Accelerometer - Sense tampering and other excessive motion
- PIR Sensor - Critter Detector: Tells the system to begin recording
- Temperature & Humidity Sensors - Sense environmental parameters inside the Cat Box
- 12V Battery Voltage & Current Sensors - Track the health and power usage of the battery

Power

- DC to DC Power Stabilizer for Camera
- DC to DC Power Converter for ESP32
- Deep Cycle Battery
- Battery Charger
- Circuit Breaker

Cat Box Wiring Diagram

Under Construction



Server & Inference Device

The NVIDIA Jetson Nano was chosen as the Server and Inference Device because of its relatively small size and low-cost to performance ratio

- This procedure can be refined significantly, but for this demonstration, the Nano receives video full-time as an RTSP Stream but does not process it until triggered over Wifi by the ESP32 connected PIR Motion Detector in the Cat Box
- The ESP32 and its Sensors use an RTOS to significantly reduce power consumption by Parallelization via Threading for use in a Low-SWaP Environment.
- When the Server is triggered, it begins recording and running inferences to determine whether the heat signature detected by the PIR belongs to a cat. Recorded video and detections are manually reviewed for accuracy and the results fed back into the model for further training
- In addition, each hour the ESP32 in the Cat Box sends battery voltage and vital statistics to the server. This acts as a Ping to inform the server the Cat Box is alive and well
- Communication from the ESP32 is performed over Wifi using a rudimentary REST API on the Jetson

First Time Setup for NVIDIA Jetson Nano

- Install Jetson Image and Perform Initial Setup
- Configure Jetson Inference by Dustin Franklin

This project follows "Building the Project from Source". Docker is also available: Docker Container.

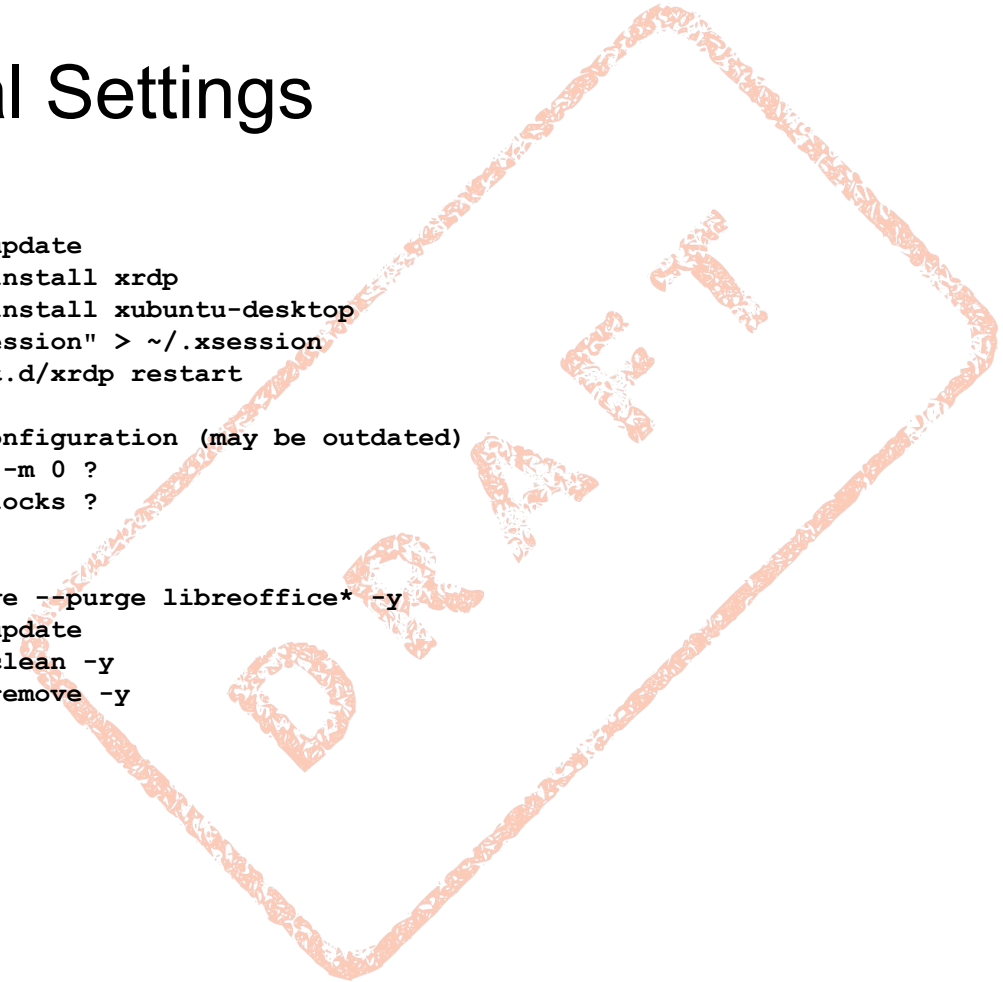
```
$ sudo apt-get update
$ sudo apt-get install git cmake libpython3-dev python3-numpy
$ git clone --recursive https://github.com/dusty-nv/jetson-inference
$ cd jetson-inference
$ mkdir build
$ cd build
$ cmake ../
$ make -j$(nproc)
$ sudo make install
$ sudo ldconfig
```

Optional Settings

```
Remote Desktop
$ sudo apt-get update
$ sudo apt-get install xrdp
$ sudo apt-get install xubuntu-desktop
$ echo "xfce4-session" > ~/.xsession
$ sudo /etc/init.d/xrdp restart

Power and GPU Configuration (may be outdated)
$ sudo nvpmodel -m 0 ?
$ sudo jetson_clocks ?

Clean Up
$ sudo apt remove --purge libreoffice* -y
$ sudo apt-get update
$ sudo apt-get clean -y
$ sudo apt autoremove -y
```

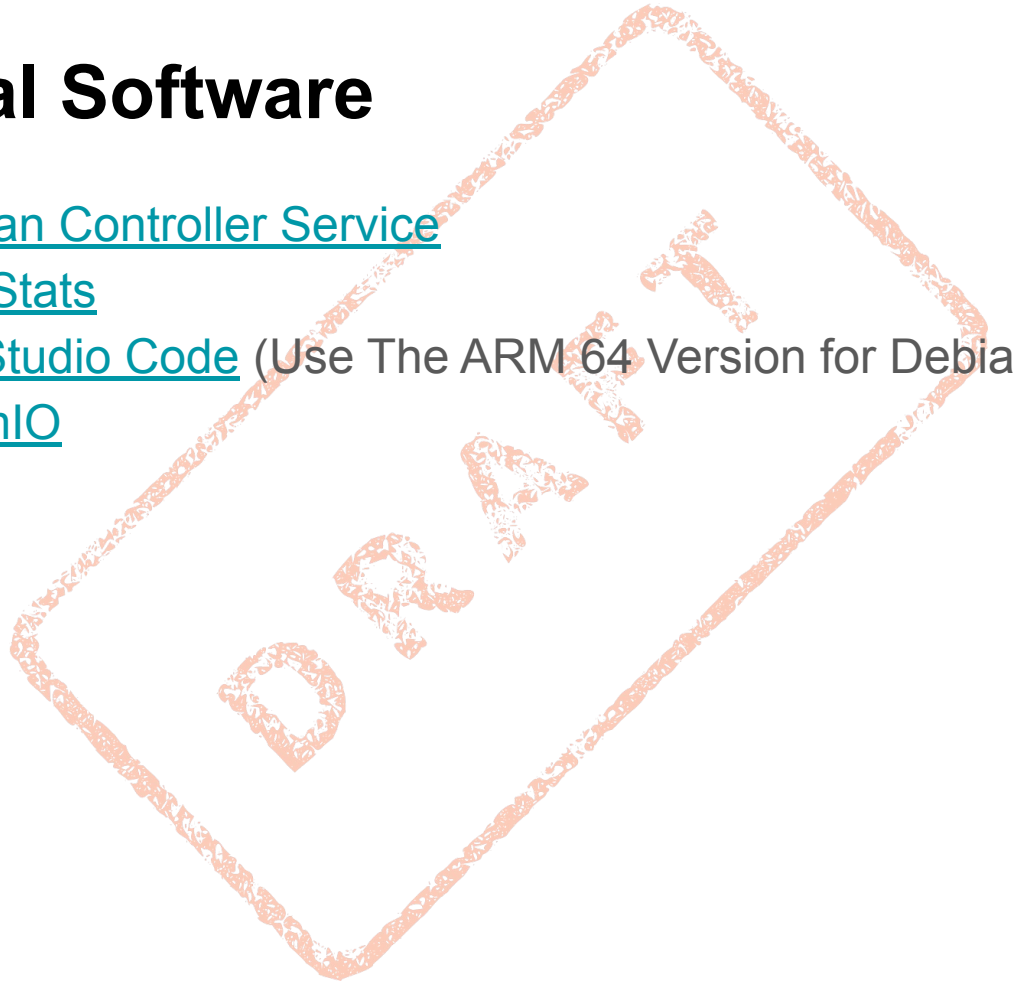


Pieces Parts

- 4GB NVIDIA Jetson Nano
- 64GB microSDXC UHS-I Memory Card
- RTSP Capable Camera
- Power Supply 12V 4A (Minimum)
- Cooling Fan - The unit gets hot while performing heavy processing
- Minimal Wifi Adapter - Very slow and unreliable
- Better Wifi Adapter - Seems like most adapters in this class work
- Internal Wifi Adapter - This is probably good option especially when using a case
- Optional Case

Optional Software

- PWM Fan Controller Service
- Jetson Stats
- Visual Studio Code (Use The ARM 64 Version for Debian)
- PlatformIO



AI Catter Waller

Although most of the cats in this area are used to human contact, cats that are not used to human contact can inflict serious harm, can be difficult to capture, and carry the risk of Rabies. They are essentially wild animals and should be given a wide latitude.

They are notoriously difficult to capture because they are typically very wary. **The Catter Waller uses Deep Learning to identify the desired subject and only closes the trap when that particular Cat enters**. Otherwise, less wary cats who may have already been treated and checked may trigger the trap making the target Cat even more wary of being walled in. caterwauling

Security

The items constructed for this project should not be considered secure

- No Internet Connection
- In this case, the data not sensitive at all
- The cameras are accessed using a username and password which is passed as clear text over the wire
- The Cat Box Controller uses an HTTP REST API over Wifi on the Server which is not secure
- The only security is as a result of everything running on a separate Wifi Network Router from my Primary Network which should help keep my internal network secure
- One can be reasonably assured a hacker could breach these devices and snoop or create havoc

Credits

Example Software & Configurations Referenced

- Kaggle - Thanks to everyone for making their notebooks and datasets publicly available
- Dustin Franklin
- Adrian Rosebrock
- Maintainers of OpenCV, NumPy, Pillow, and the other numerous libraries without which this project would have not been possible

Initial Cat Datasets

- Kaggle: Cat VS Dog Dataset
- Cats and Dogs Breeds Classification Oxford Dataset

Initial Background Datasets

- Messy vs Clean Room
- 15-Scene Dataset
- Qutub Complex Monuments' Images Dataset