## 1. Problem statement

Create a weapon detection system capable of displaying a warning message when potentially harmful objects are uncovered in an image and/or a recording. The application should either run on the web or a mobile application.

## 2. Data Preprocessing

### 2.1. Data Description

My final dataset contains firearms images separated into training and testing (https://sci2s.ugr.es/weapons-detection). Initially, I chose to detect different weapons, such as firearms or knives, but I do not think my model would be extremely accurate if trying to detect too many classes. Therefore, I've decided to only detect firearms, as I prefer a more reliable model over a more flexible one. I also changed my dataset, as the initial one only contained images of a static weapon with no background, which would increase the chance of bias. The new dataset contains a mix of firearms as well as people holding a firearm in 3000 training images. I've also added 3000 more images of normal human activity without firearms from the human pose dataset (http://human-pose.mpi-inf.mpg.de/#dataset). The labelling is binary: 0 if there are not firearms, 1 if there are. There are also 608 testing images of which 304 contains a firearm.

### 2.2. Data Preprocessing Methods

i. Quickly look through the dataset and manually delete any aberrant images
ii. Center all images on the firearm and convert them to the same dimensions
iii. Change the images to greyscale
iv. Visualize the results (optional, for testing)
v. Separate into 2 arrays: features & labels
vi. Convert to numpy arrays

## 3. Machine Learning Model

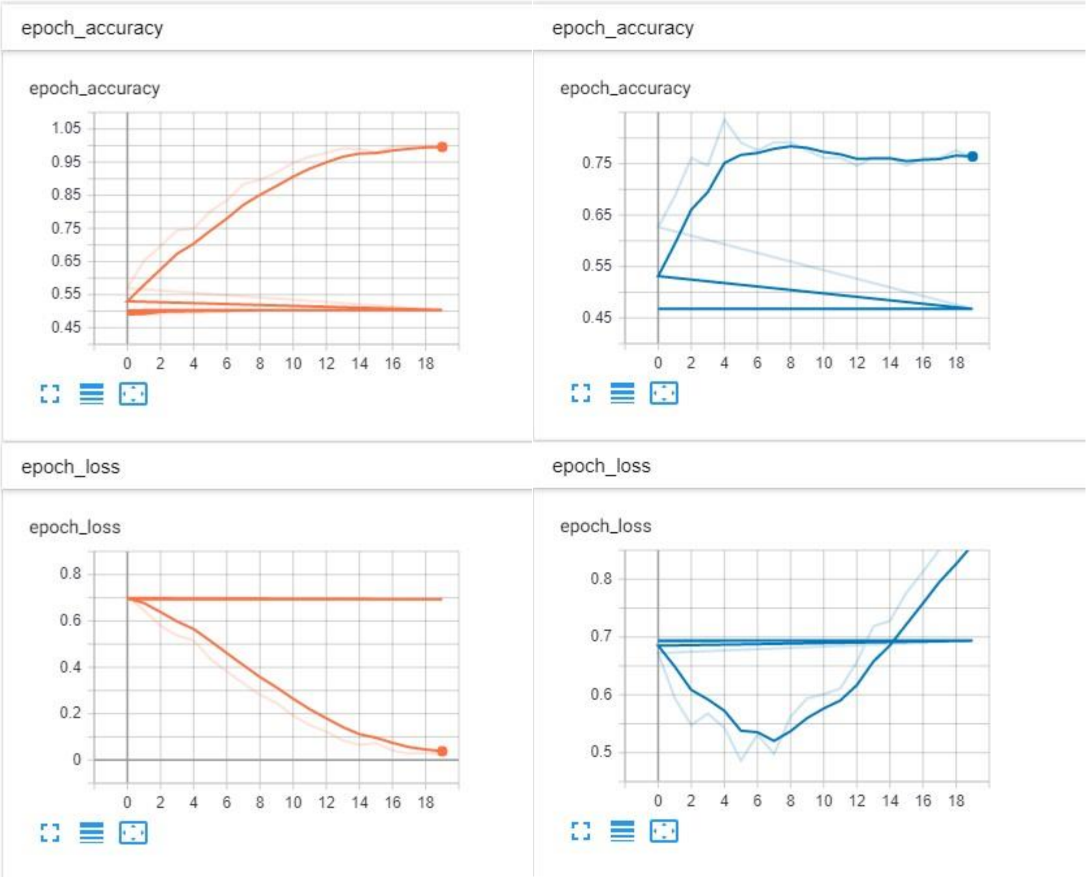
As stated in the first deliverable, I wanted to use a supervised deep learning algorithm. More specifically, I will use a convolutional neural network to classify my images. I will determine an appropriate number of layers (filters) to detect any patterns that will help me determine if an image has a firearm or not.

I used os, cv2 and matplot to extract the data, visualize it and store it in a list. I also used numpy for matrix computations and numpy array manipulations. Finally, I used tenserflow & keras for the model, as I can easily add layers to my CNN and specify a variety of hyperparameters in each.

For the model, I added a initialized a Sequential model and added 2 convolutional layers, which consists of a 2D convolution using a 3 x 3 window. A ReLU activation layer and a 2x2 window max pooling. I then flattened the matrix to 1D and passed it to a 64 units dense layer. Finally, the 1-unit dense layer output is passed to a sigmoid activation function. The model used was inspired from a cat/dog classifier tutorial and will be changed later. I have not tried changing hyperparameters yet, as I have not decided on the right dataset.

I tested the model using 30% of the test data as a validation set. The validation set was less accurate than the training set, therefore, my model was overfitting. I will use regularization in the dense layer to address the issue in the next deliverable. I will also add more images to the training set.

## 4. Preliminary results



The model performed a lot better than expected, but it is still biased and overfitting. Also, only a total of 600 images were used to get this result.

## 5. Next steps

I did not have much time for this deliverable, as I was researching on CNNs and how they work. The datasets were also extremely hard to find because it is a sensible topic. Therefore, I plan on expanding the dataset to make the application more reliable and test out different layers/hyperparameters using the knowledge I gained in this deliverable. I'll also have to decide my application's viewpoint (I plan on using a simple webcam or phone camera POV, as I got terrible results using the surveillance camera POV). My goal is to have an accuracy of about 85% using more than 10k images.