PROTOTYPE MODEL(DL BASED) TO PREVENT JAILBREAKS

AIM: PERSON LESS SURVEILLANCE BUILT FOR HIGH SECURITY AREAS LIKE PRISONS.

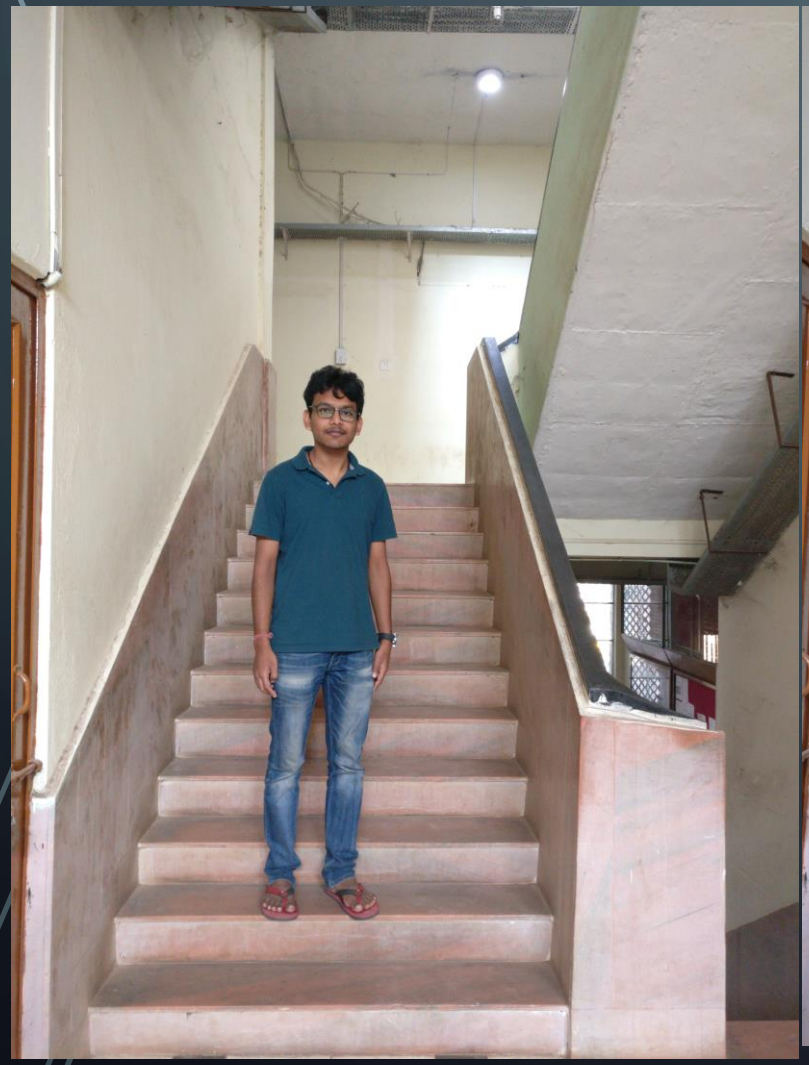
SPECIFIC PROBLEM WE ARE TACKLING : TO MONITOR IF A PRISONER IS TRYING TO ESCAPE THROUGH A CERTAIN PATHWAY SAY A CORRIDOR OR STAIRCASE. VALID ASSUMPTIONS: WE HAVE A STATIC CAMERA. WHAT WE HAVE BEEN ABLE TO DO UP TILL NOW? WE WERE ABLE TO CREATE A MODEL THAT CAN CORRECTLY CLASSIFY WHETHER A PERSON IS PRESENT ON THE STAIRS OR NOT.

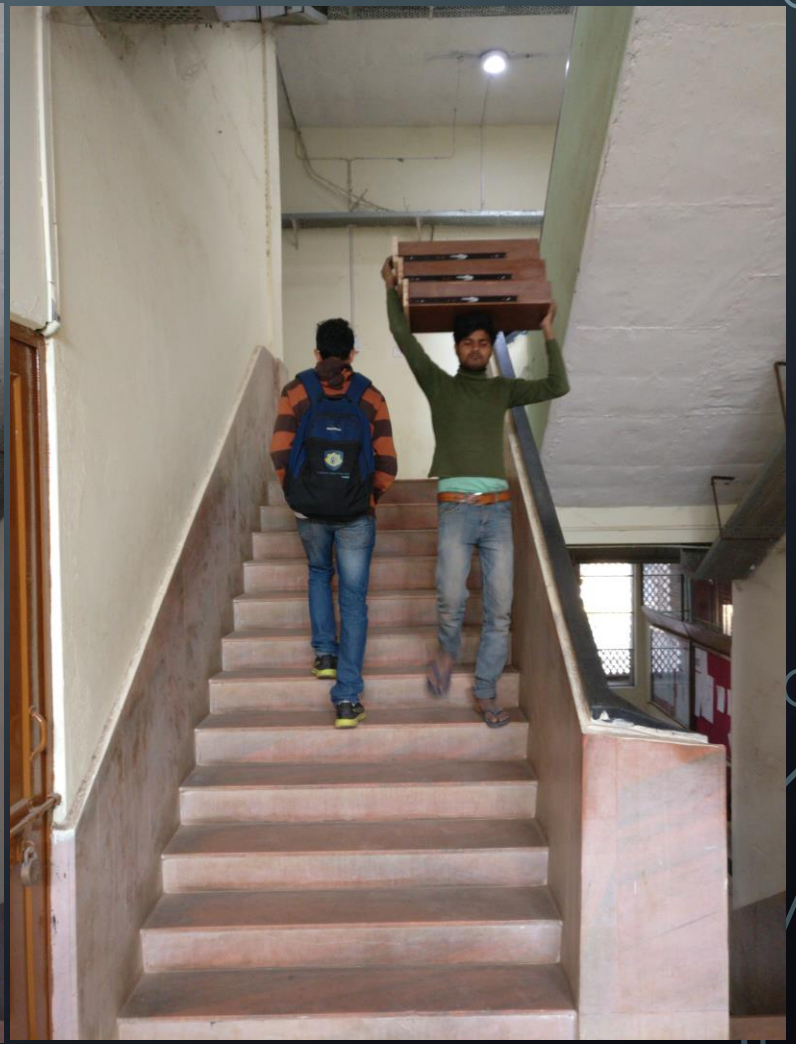
OUR APPROACH TO THE PROBLEM STATEMENT

WE DECIDED TO USE DL MODEL AS THEY HAVE BEEN PROVED TO BE EXCLLENT IN TASKS SUCH AS IMAGE CLASSIFICATION AND OBJECT RECOGNITION.
THEY ARE DYNAMIC, AS IN THEIR WEIGHTS CAN BE TUNED/CHANGED EVEN AFTER THE PROJECT IS LIVE. THIS ALLOWS FOR ADAPTING TO NEWER PROBLEMS AND IMPOROVING THE MODEL WOTH TIME.

PHASE 1: PREDICTING ON STILL IMAGES

- OUR FIRST CHALLENGE WAS JUST TO RUN MODEL ON STILL IMAGES.
- TO SIMULATE A STILL CAMERA IN A PRISON HAVING A FIXED FIELD OF VIEW WE DECIDED TO TRAIN AND TEST OUR MODEL ON A STAIRCASE WITH PEOPLE WALKING ON THE STAIRCASE POTRAYING PRISONERS TRYING TO ABSCOND - "ALERT" AND EMPTY STAIRS AS THE USUAL CONDITION - "NO ALERT".



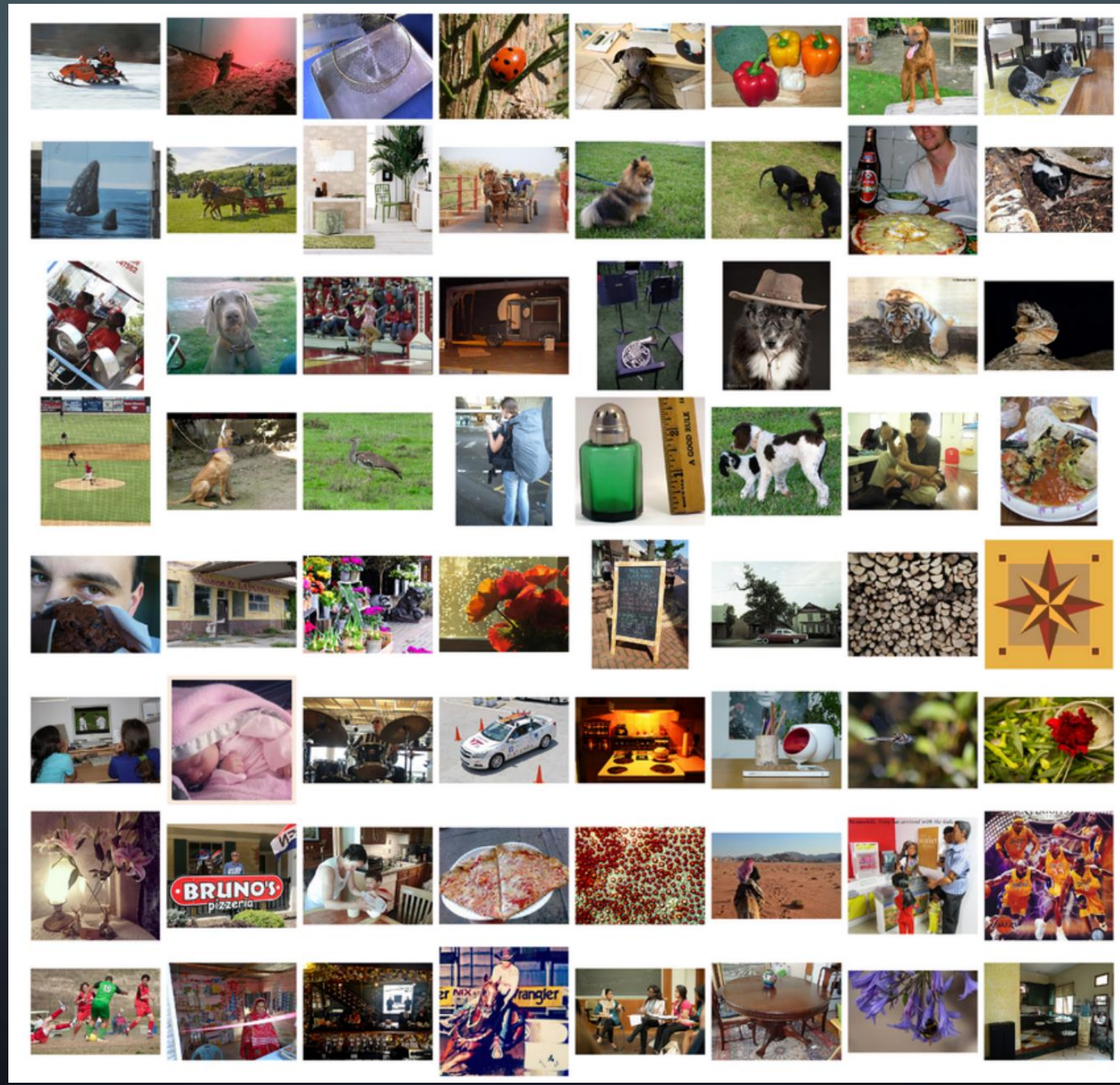


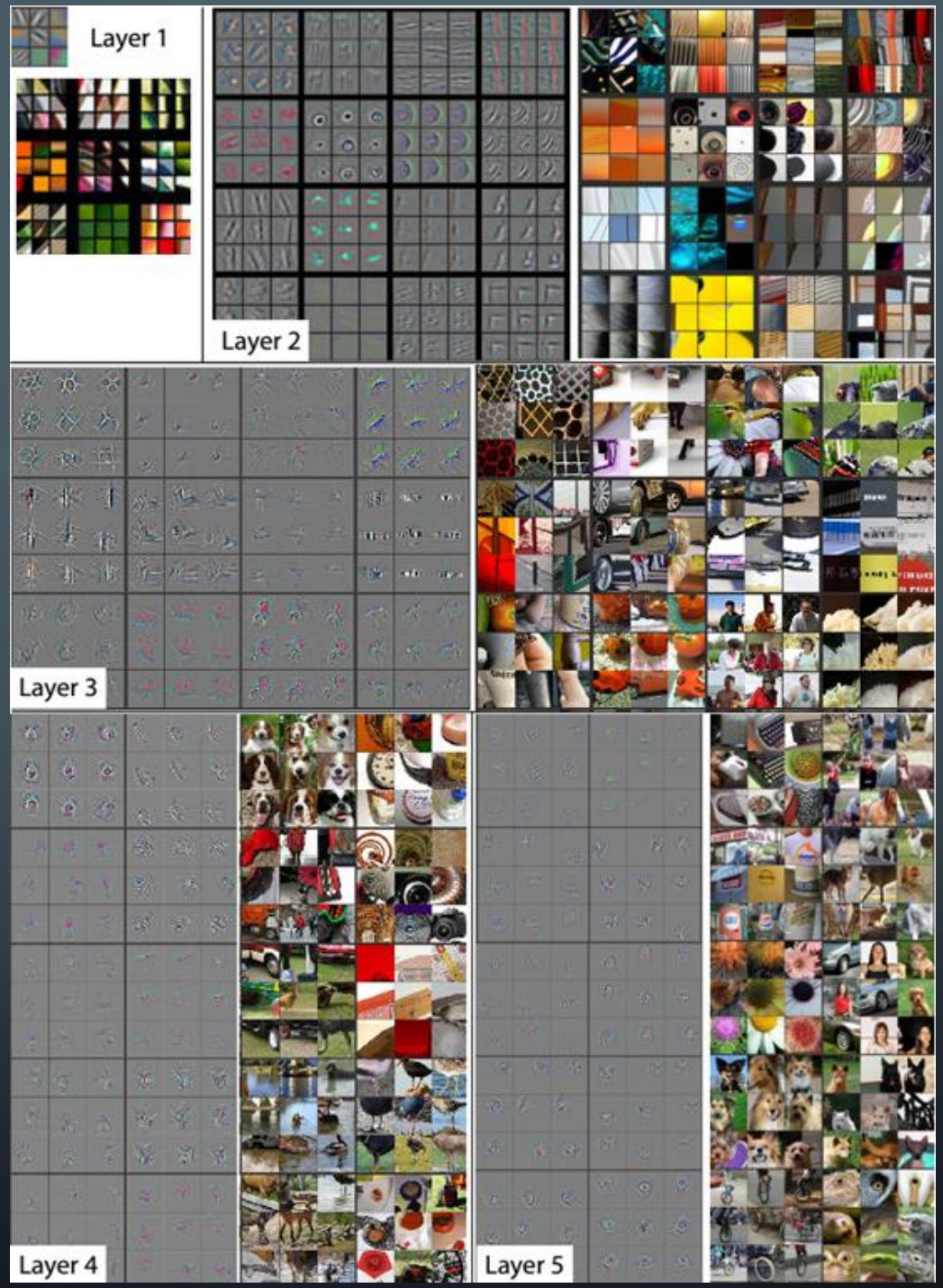
PROBLEM FACED - "VERY LESS DATA"

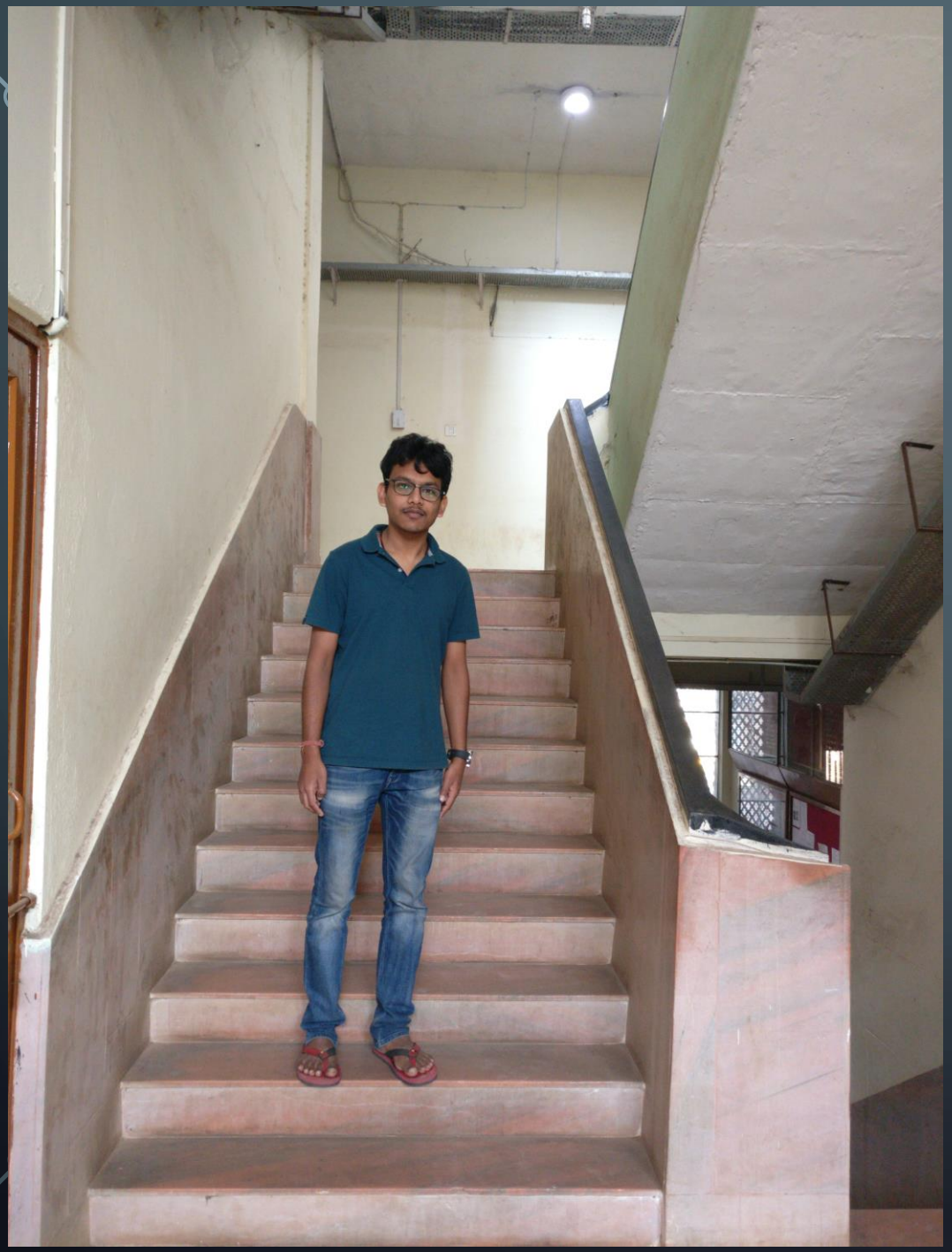
- DEEP LEARNING MODELS ARE VERY DATA INTENSIVE AND MAY FAIL TO GIVE ANY MEANINGFUL RESULT IN ABSENCE OF HUGE DATASET.
- SIMILAR DEEP LEARNING PROBLEMS ARE USUALLY TRAINED ON TENS OF THOUSANDS OF IMAGES BUT CREATING SUCH A LARGE DATASET WOULD HAVE BEEN INFEASIBLE.

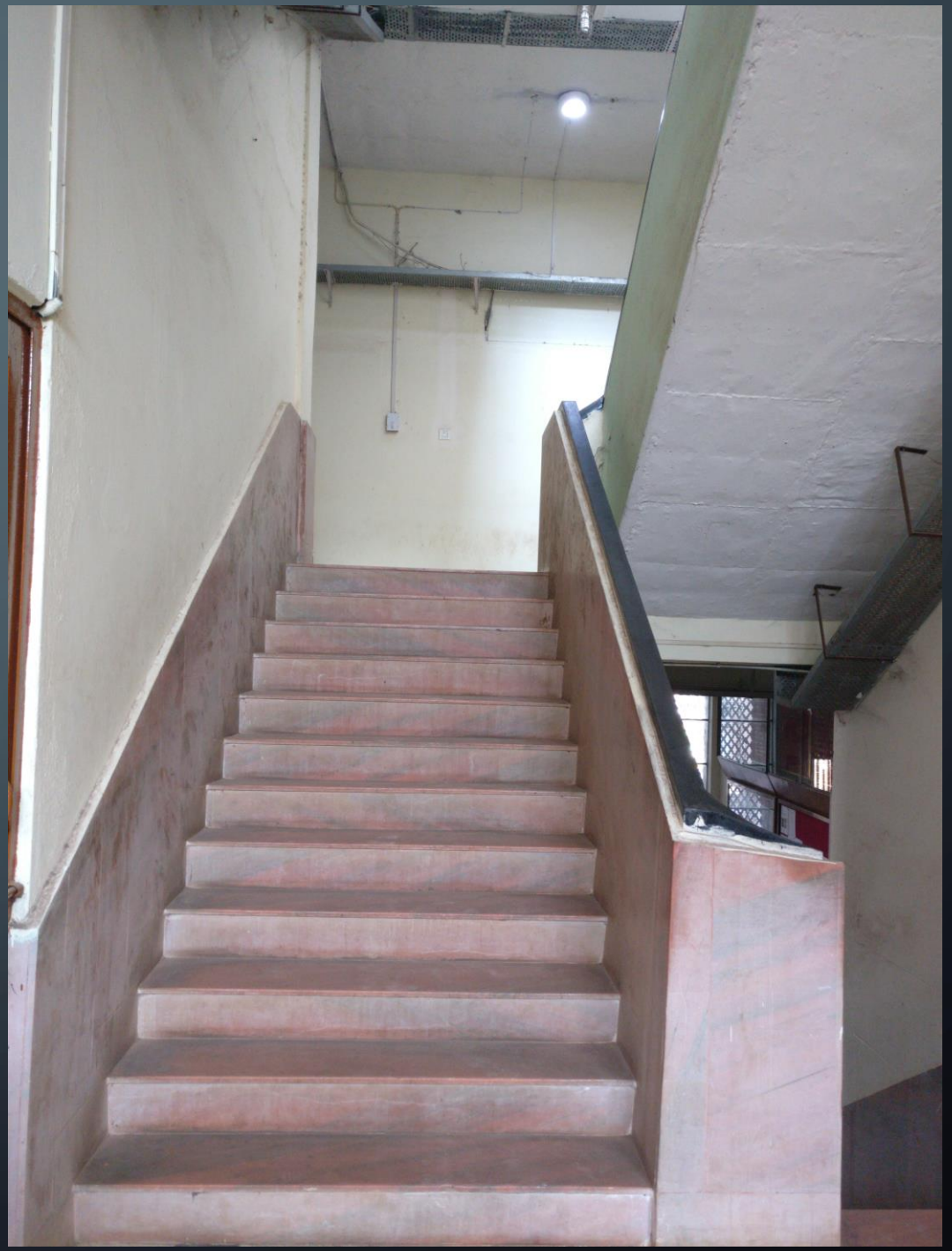
APPROACH USED - "TRANSFER LEARNING"

- IT IS A PROCESS OF USING AN APPLICATION OR MODEL WHICH IS DESIGNED TO SOLVE A PARTICULAR PROBLEM AND TUNING IT TO OUR OWN PURPOSE.
- WE USED RESNET34 MODEL TRAINED ON IMAGENET DATASET.tTHIS DATASET CONTAINS MORE THAN 14 MILLION IMAGES. THE MODEL IS ALREADY VERY GOOD AT PREDICTING LOW LEVEL FEATURES AND EVEN OUR IMAGES WILL HAVE SAME LOW LEVEL FEATURES.
- PUT OUR IMAGES IN TRAINING AND VALIDATION TEST


BRUNO'S
pizzeria


Layer 1
Layer 2
Layer 3
Layer 4
Layer 5

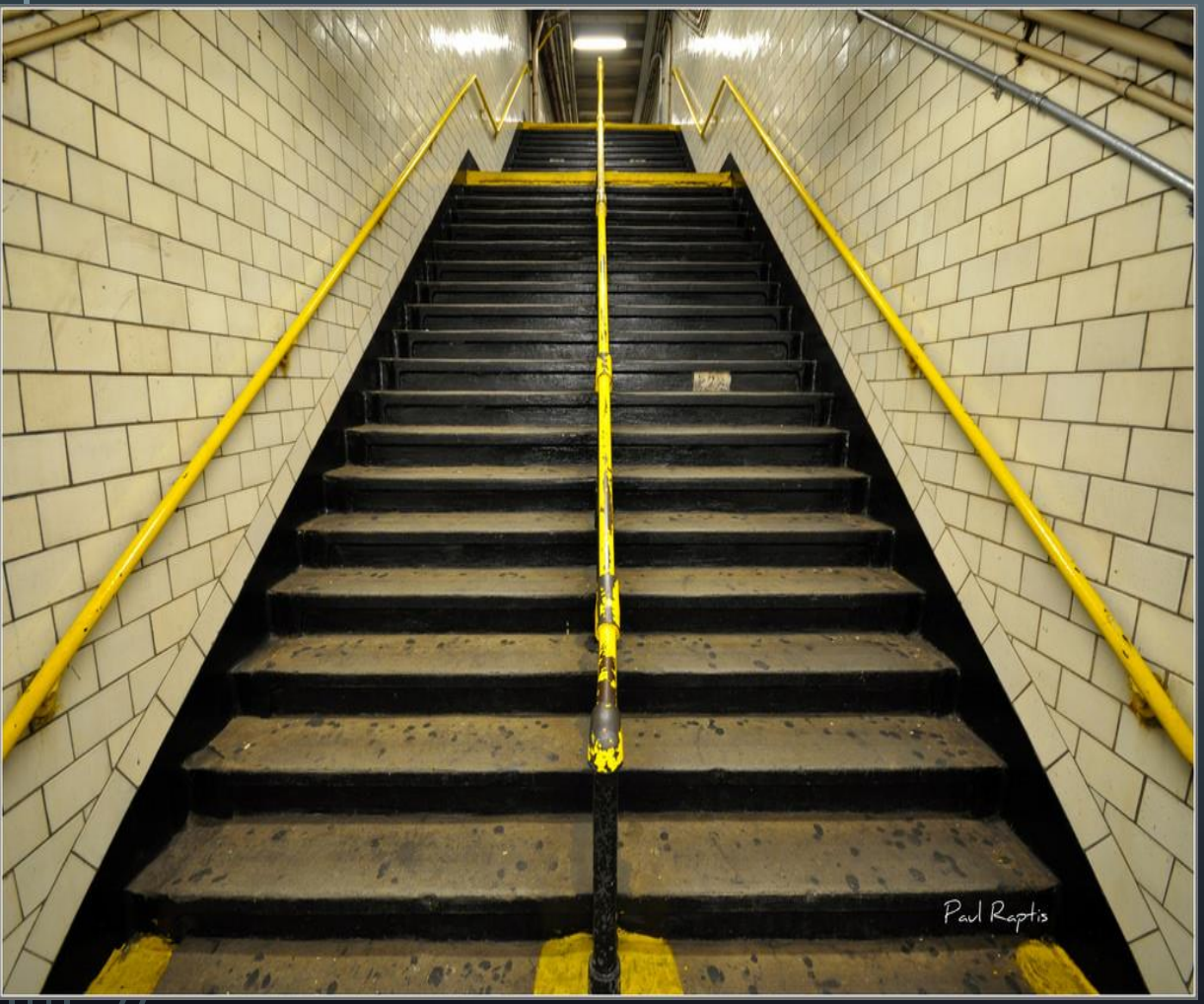




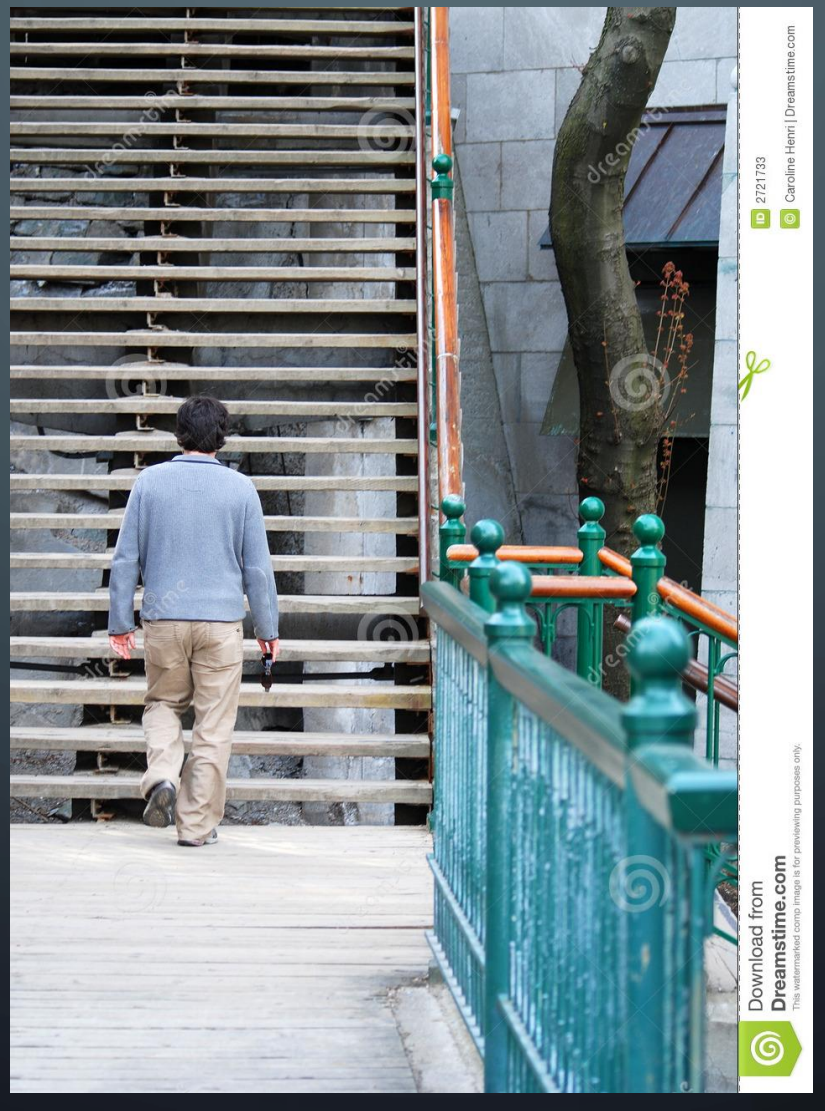
PROBLEM FACED - "OVER-FITTING"

- Training and development data:
  Since the data we have is very minuscule hence we faced the problem of overfitting that is because the model is so huge it so happens that our model becomes exceptional at identifying the "exact" images which on which it trained on The reason for this is that since we have assumed a certain fixed camera and the images are few thus model can't generalize enough.
- Solution: We downloaded images similar to our requirements and used these to increase the number of training images and also the variation in them so that our model could finally generalize better. To keep the results unbiased and relevant to problem we still had images specific to problem in our test data.
- Result: We were able to achieve 100% accuracy on our test set.

EMPTY STAIRS

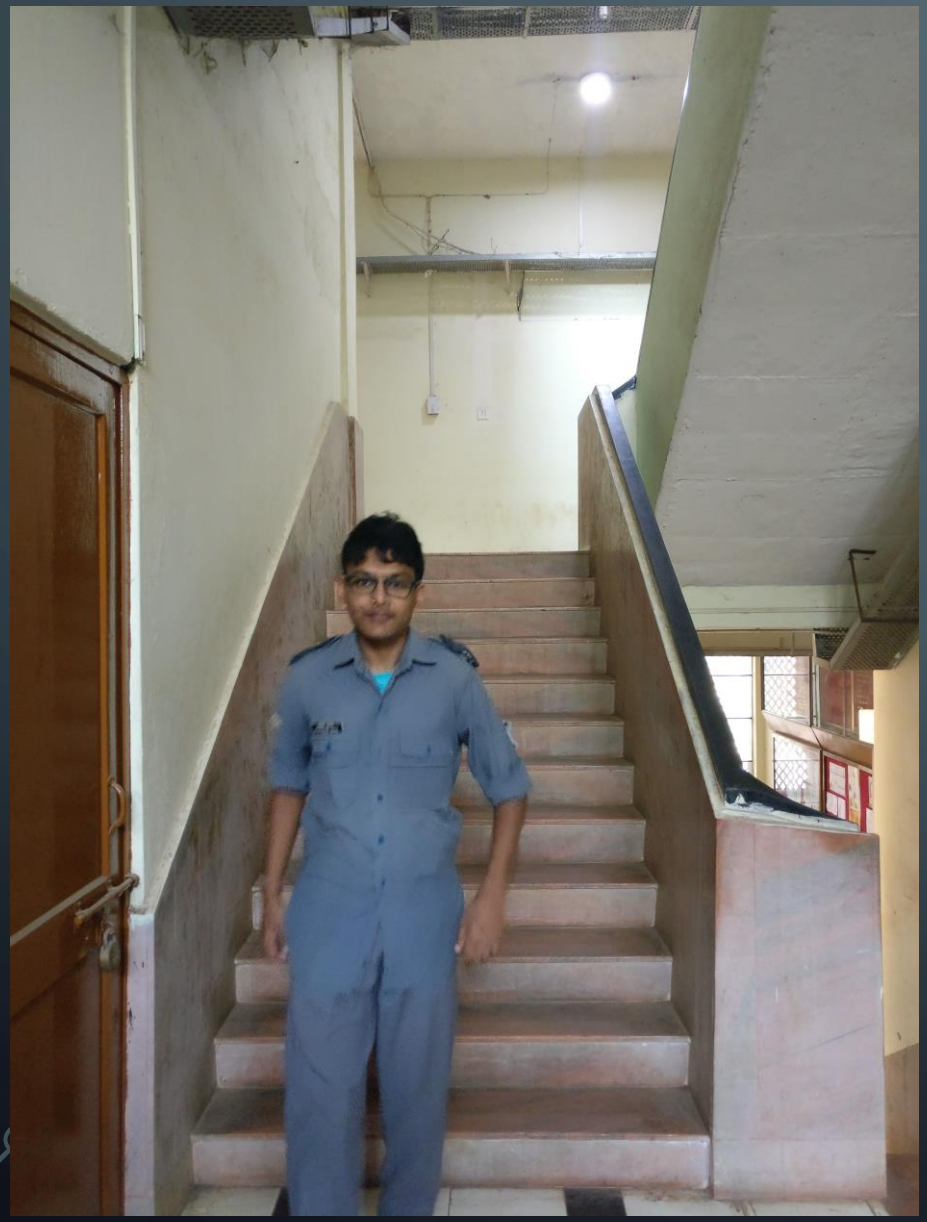


NON-EMPTY STAIRS



PHASE 2 : DIFFERENTIATING POLICE PERSONNELS FROM PRISONERS

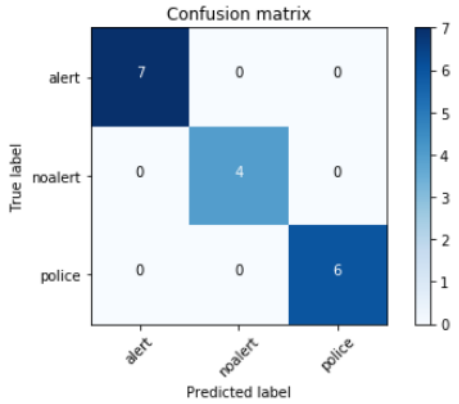
- OUR PREVIOUS MODEL COULD JUST PREDICT IF SOMEONE IS WALING ON THE STAIRS OR NOT, BUT IN REAL LIFE SUCH A SYSTEM SHOULD BE ABLE TO DIFFERENTIATE A POLICE PERSONNEL FROM A PRISNOR.
- OUR EQUIVALENT FOR A POLICE PERSONNEL WAS A STUDENT IN NCC UNIFORM.
- THIS TIME MODEL REQUIRED MUCH MORE ITERATIONS TO REACH SAME ACCURACY.



We can just print out the confusion matrix, or we can show a graphical view (which is mainly useful for dependents with a larger number of categories).

In [58]:
```
plot_confusion_matrix(cm, data.classes)
```

```
[[7 0 0]
 [0 4 0]
 [0 0 6]]
```



Looking at pictures again

In [57]:
```
plot_val_with_title(most_by_correct(0, False), "Most incorrect alerts")
```

```
Most incorrect alerts
```

```
<matplotlib.figure.Figure at 0x7faf180babe0>
```

In [58]:
```
plot_val_with_title(most_by_correct(1, False), "Most incorrect no-alerts")
```

```
Most incorrect no-alerts
```

```
<matplotlib.figure.Figure at 0x7faf180b0b70>
```

In [35]:
```
import sys
import argparse
import os
import time
import cv2

from IPython.display import HTML
```

PHASE 3 : PREDICTION ON A VIDEO

- IN REAL LIFE SCENARIO THE CCTV CAMERA WILL GIVE OUR MODEL A VIDEO FEED AND NOT STILL IMAGES HENCE NEED TO MAKE OUR APPLICATION CAPABLE OF TACKLING THE SAME.
- APPROACH- WE KNOW THAT A VIDEO IS MADE OF MULTIPLE FRAMES OR STILL IMAGES HENCE WE USED OPENCV LIBRARIES TO EXTRACT IMAGES FROM THE VIDEO FEED.
- A VIDEO GENERALLY HAS A FPS GREATER THAN 30 BUT WE DON'T NEED TO CHECK CAMERA 30 TIMES EACH SECOND ! HENCE WE DECIDED TO TAKE IMAGES AFTER FIXED TIME CONSTANT. IT IS VERY EASY TO CHANGE THIS TIME CONSTANT IN THE CODE.

jupyter Stairs_and_Police Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted Python 3

Run Markdown

```
from IPython.display import HTML
```

In [25]:
```
def video_to_frames(input_loc, output_loc):
    """Function to extract frames from input video file
    and save them as separate frames in an output directory.
    Args:
        input_loc: Input video file.
        output_loc: Output directory to save the frames.
    Returns:
        None
    """
#     try:
#         os.mkdir(output_loc)
#     except OSError:
#         pass
    # Log the time
    time_start = time.time()
    # Start capturing the feed
    cap = cv2.VideoCapture(input_loc)

    # Find the number of frames
    video_length = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) - 1

    # find fps of the video
    fps = cap.get(cv2.CAP_PROP_FPS)


    print("fps: {}".format(fps))
    print ("Number of frames: ", video_length)
    count = (int)(0)

    # WE ARE TAKING A FRAME EVERY 2 SECONDS hence gap is fps*5
    gap = (int)(fps*0.5)
    print ("Converting video..\n")


    # Start converting the video
    while cap.isOpened():
        # Extract the frame
        ret, frame = cap.read()
        # Write the results back to output location
        if count%gap == 0:
            num = count//gap
            cv2.imwrite(output_loc + "/%#05d.jpg" % (num), frame)
        count = count + 1
        # If there are no more frames left
        if (count > (video_length-1)):
            # Log the time again
            time_end = time.time()
            # Release the feed
            cap.release()
            # Print stats
            print ("Done extracting frames.")
            print ("{} frames extracted".format(num+1))
            print ("It took {} seconds forconversion." .format((time_end-time_start)))
            break
```

In [38]:
```
input_loc = '/home/himanshu/Videos/Video3/001.mp4'
```

In [28]:
```
output_loc = "/home/himanshu/data/stairs/test"
```