**Files:**

1. get\_frame.py samples frames from the videos
2. test\_model.ipynb shows the work flow of creating the ML model and pipeline
3. test\_model.html is a html copy of the jupyter notebook
4. The trained model is saved under folder “my\_model”
5. predict\_img.py takes a single image and prints its classification
6. “data” folder stores videos and frames, only include frames due to size issue

**Libraries:**

1. moviepy
2. tqdm
3. PIL
4. Matplotlib
5. Pathlib
6. Tensorflow
7. Keras

**Execution:**

Run predict\_img.py, and you will be prompted to enter the path of an image, then the program would print the classification of that image.

Example:

A picture containing text

Description automatically generated

**Algorithm:**

For pre-processing step, I noticed that drill appears in the middle left area of all frames, so I cropped all images to focus on the drill part.

Using the cropped images, I split them into 80% training and 20% testing and normalize the dataset.

I decided to build a CNN using tensorflow & keras. The model contains three 2D convolution layers with activation function “relu” and each followed by a max pooling layer. Then I add a dropout layer to avoid overfitting. Finally, the model ends with a flatten layer and output layer.

I have experimented with some other combinations of kernel size, number of layers and activation function, and the above model is the best performing one. The strength of my model is that it has high accuracy (around 96%). The weakness of my model is that it requires the target object, which is drill in this case, to be on the right and consistent position, lack of variability. And there maybe an overfitting issue with my model.

**Performance:**

Test accuracy: 96%

A picture containing diagram

Description automatically generated