```
import os
import numpy as np
import cv2

from keras.models import load_model

# Load saved model
model_folder = os.path.join(os.getcwd(), "w251_model\FirstModel")
model_file = os.path.join(model_folder, "cifar10_ResNet20v1_model.086.h5") # first
saved_model = load_model(model_file)

# load training set mean image array (x_train_mean)
mean_file = os.path.join(model_folder, 'x_train_mean.npy')
x_train_mean = np.load(mean_file)

# Live video classification

# Set class labels
labels = ['zinc','stainless steel','copper','brass','aluminum']

# Set resize dimensions
img_size_h = 224
img_size_w = 224

cap = cv2.VideoCapture(0)

# set exposure to brighten workspace and also mitigate LED light "banding"
cap.set(cv2.CAP_PROP_EXPOSURE, -8.0)

while(True):

    # Capture frame-by-frame
    ret, frame = cap.read()

    # frame center coords (for center crop and rectangle reference)
    center = (frame.shape[0]/2, frame.shape[1]/2) # (240, 320)

    # center crop corners
    upper_left = (int(center[1])-100, int(center[0])-100)
    bottom_right = (int(center[1])+100, int(center[0])+100)

    # define image crop
    img_crop = frame[upper_left[1] : bottom_right[1], upper_left[0] :
    bottom_right[0]].copy()

    # predict on center crop only
    img_crop_res = cv2.resize(cv2.cvtColor(img_crop, cv2.COLOR_BGR2RGB), (img_size_h,
    img_size_w))
    img_crop_res = img_crop_res.reshape((1,img_size_h, img_size_w,3))
    img_crop_res_no_mean = img_crop_res - x_train_mean
    result = saved_model.predict(img_crop_res_no_mean)

    # visual ref for center of image (as desired)
    cv2.rectangle(frame, upper_left, bottom_right, (0, 255, 0), 2)

    ### if predict on center crop not desired; use this block instead

    # Inference
    #frame_res = cv2.resize(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB), (img_size_h,
    img_size_w))
    #frame_res = frame_res.reshape((1,224,224,3))
    #frame_res_no_mean = frame_res - x_train_mean
    #result = saved_model.predict(frame_res_no_mean)

    ###

    # Classify "unknown" if prediction precision is too high
```

```
        if np.isclose([1.0], [np.max(result)], atol=1e-08)[0]:
            # Unrealistic class prediction
            # source image not a relevant match to data model trained on
            overlay = 'Unknown'
            #pred_class = result.argmax()
            #overlay = 'Unknown or ' + labels[pred_class] + ' ' + '({})'.format(result.max())
        else:
            pred_class = result.argmax()
            overlay = labels[pred_class] + ' ' + '({0:.4f})'.format(result.max())

        # format overlay text
        font                   = cv2.FONT_HERSHEY_SIMPLEX
        bottomLeftCornerOfText = (int(frame.shape[0]*0.1),int(frame.shape[1]*0.1))
        fontScale              = 1
        fontColor              = (0,0,0)
        lineType               = 2

        cv2.putText(frame,overlay,
            bottomLeftCornerOfText,
            font,
            fontScale,
            fontColor,
            lineType)

        # Display frame
        cv2.imshow('frame',frame)

        # Loop termination criteria
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```