Reading Analog Clocks

COMP 4102
Malcolm Yeh
101109829

Abstract

While there are many resources available for optical text recognition, analog devices such as clocks present a different computer vision problem. A working interface for reading analog meters can have a wide variety of usages, including monitoring of equipment to further extraction of information in images and videos.

Changes from Initial Proposal

The focus of the project shifted from reading both analog clocks and gauges to clocks only. Furthermore, due to limitations of the implementation of the feature extraction based approach, the test harness in order to compare speed and accuracy of the two methods was scrapped.

1. Introduction

The goal of this project is to explore the effectiveness and efficiency of different techniques for reading analog clocks. In particular, a feature extraction based program using OpenCV functionality and a CNN model will be implemented in order to detect time from clock images. Furthermore, a data pipeline to generate, label and load clock images is required to train the CNN model.

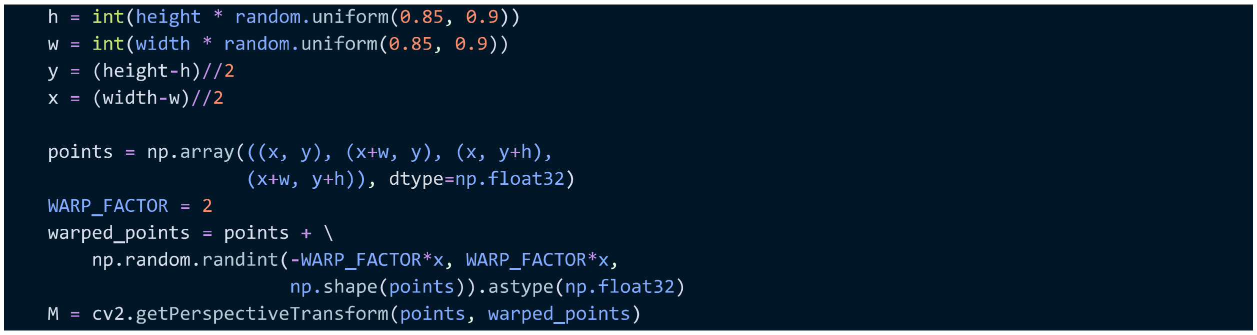
2. Background

The convolutional neural network model is inspired by the findings in Krizhevsky et al.'s paper on image classification using deep convolutional neural networks [1]. In particular, the network contains eight learned layers, five of which are convolutional and three are fully-connected. Networks with rectified linear units (ReLU) trained much faster with lower training error compared with tanh neurons. Thus, ReLU non-linearity is applied to the output of each convolutional and fully-connected layer. Pooling layers in the network summarize the output of neurons, resulting in lower computation cost and overfitting by reducing dimensionality. To further reduce overfitting, dropout is used, which sets the output of neurons to 0 with a specified probability, which results in dropped neurons not influencing forward pass and backpropagation. For the OpenCV program, the algorithm to deskew by first calculating the bounding rectangle and minimum enclosing circle of the contour and calculating the resulting matrix from the bounding box to circle was modeled after a StackOverflow discussion[3].

3. Methodology

3.1 Dataset

The dataset consists of clocks whose values can be manually verified. Since the first goal of the project is focused on image processing and feature extraction, this dataset does not need to be large and can be easily procured. However, for the training and testing of a convolutional neural network, there needs to be a much larger set of labeled images of clocks that is not readily available on open resources such as Kaggle. Therefore, 2D clock images are automatically generated and labeled as input. Using the program generate_clocks.py, clocks are generated with random hand lengths and thickness, number of hands (second hand optional for analog clocks), border thickness, ticks and numerals (arabic, roman or none) and skew.



```
h = int(height * random.uniform(0.85, 0.9))
w = int(width * random.uniform(0.85, 0.9))
y = (height-h)//2
x = (width-w)//2

points = np.array(((x, y), (x+w, y), (x, y+h),
                  (x+w, y+h)), dtype=np.float32)
WARP_FACTOR = 2
warped_points = points + \
    np.random.randint(-WARP_FACTOR*x, WARP_FACTOR*x,
                      np.shape(points)).astype(np.float32)
M = cv2.getPerspectiveTransform(points, warped_points)
```

Figure 1: Skew calculation to obtain transformation matrix which is use to warp perspective

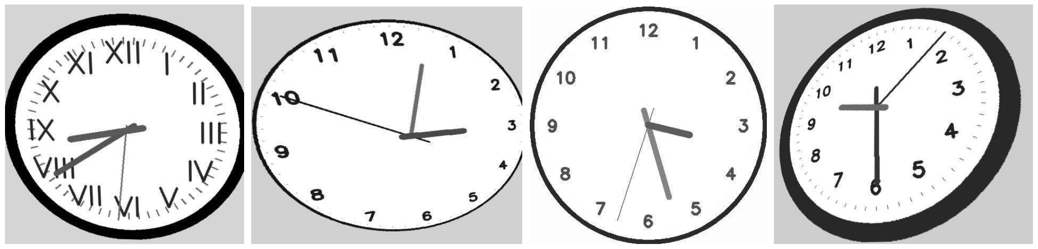


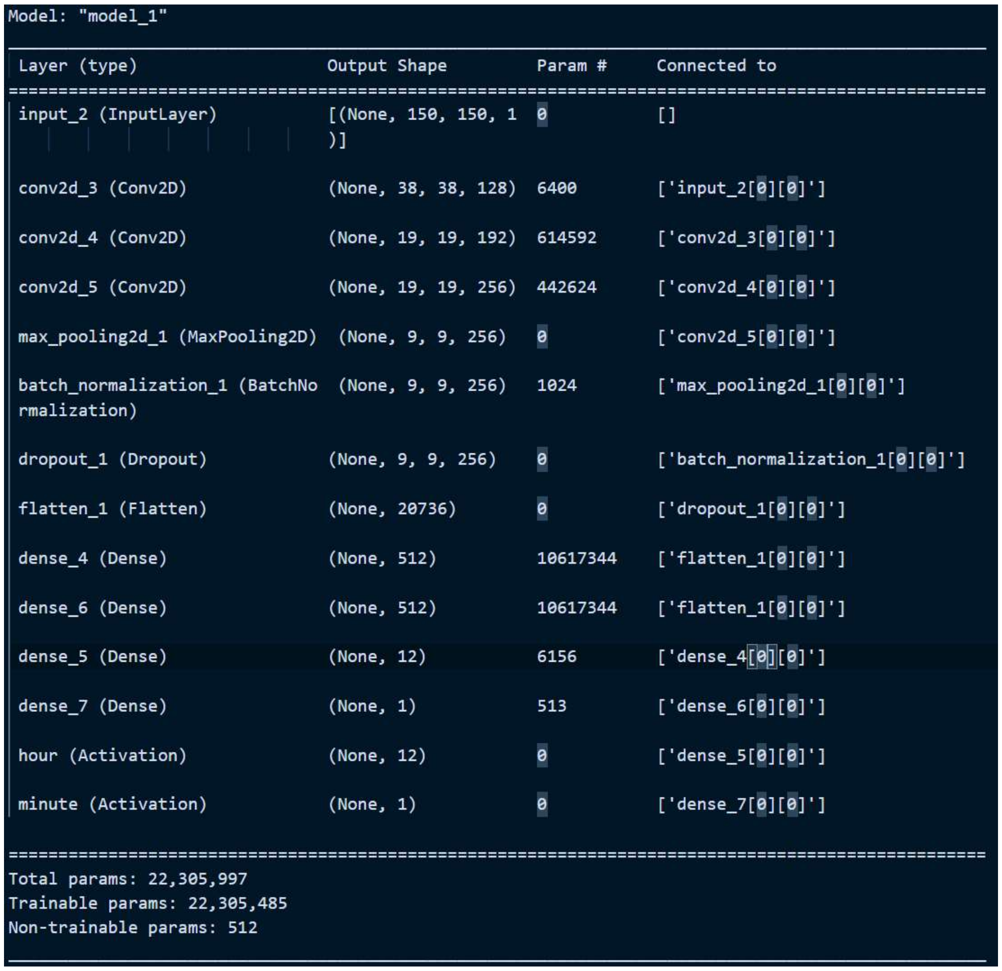
Figure 2. Examples of clocks generated using generate_clocks.py

3.2 OpenCV Feature extraction

The program first attempts to deskew the image by using a bounding rectangle and maximum enclosing circle or contour ellipse transformation. Next, using Hough transform, lines corresponding to the clock hands are detected. Since the detected lines correspond to the edges of the clock hands, they are then grouped based on similarity of angles (threshold determined by thickness of clock hands) and an average is taken to find the most accurate angle value. The angle is determined based on the relative position of the hand endpoint to the center, which is calculated as the intersection of the averages of detected lines, in order to correct angles that are off by factors of pi. The lines are then ordered by length, with the second hand being the longest followed by the minute hand and the hour hand. Lastly, the time is determined by converting the calculated angle of each hand.

3.3 CNN

Since there are similarities when determining separate hand values in a clock, this model will have two outputs, one for hour and another for minute. Since the dataset only consists of clocks and the features are not that diverse, it is not necessary to have 5 convolutional layers like the model presented in Krizhevsky et al.'s paper. Thus, the model starts 3 convolutional layers before normalization, pooling and dropout is applied. The first convolutional layer has a larger kernel and stride allowing the model to extract high-level features and decrease the shape of the output significantly to optimize the trainable parameters. The second and third convolutional layers detect more detailed features and thus have a smaller kernel and stride and more trainable parameters. After flattening, it branches into two fully connected layers for the hour and minute values. Since there are only 12 hours and the hour hand angle varies by 30 degrees, it is clear that classification is best suited for hour. Thus, the final layer has 12 output nodes and a softmax activation and produces a vector representing the probability of each hour. On the other hand, it can be argued that minute prediction is better suited as a regression task. However, it can also be implemented using classification with 60 output nodes. Using regression, there is a single output node and no activation (linear).



```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 150, 150, 1  0           []
                                )]

 conv2d_3 (Conv2D)              (None, 38, 38, 128)  6400        ['input_2[0][0]']

 conv2d_4 (Conv2D)              (None, 19, 19, 192)  614592      ['conv2d_3[0][0]']

 conv2d_5 (Conv2D)              (None, 19, 19, 256)  442624      ['conv2d_4[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 9, 9, 256)   0           ['conv2d_5[0][0]']

 batch_normalization_1 (BatchNo  (None, 9, 9, 256)   1024        ['max_pooling2d_1[0][0]']
 rmalization)

 dropout_1 (Dropout)            (None, 9, 9, 256)    0           ['batch_normalization_1[0][0]']

 flatten_1 (Flatten)            (None, 20736)        0           ['dropout_1[0][0]']

 dense_4 (Dense)                (None, 512)          10617344    ['flatten_1[0][0]']

 dense_6 (Dense)                (None, 512)          10617344    ['flatten_1[0][0]']

 dense_5 (Dense)                (None, 12)           6156        ['dense_4[0][0]']

 dense_7 (Dense)                (None, 1)            513         ['dense_6[0][0]']

 hour (Activation)              (None, 12)           0           ['dense_5[0][0]']

 minute (Activation)            (None, 1)            0           ['dense_7[0][0]']

==================================================================================================
Total params: 22,305,997
Trainable params: 22,305,485
Non-trainable params: 512
__________________________________________________________________________________________________
```

Figure 3: Model summary using regression for minute prediction

4. Results

4.1 OpenCV Feature extraction

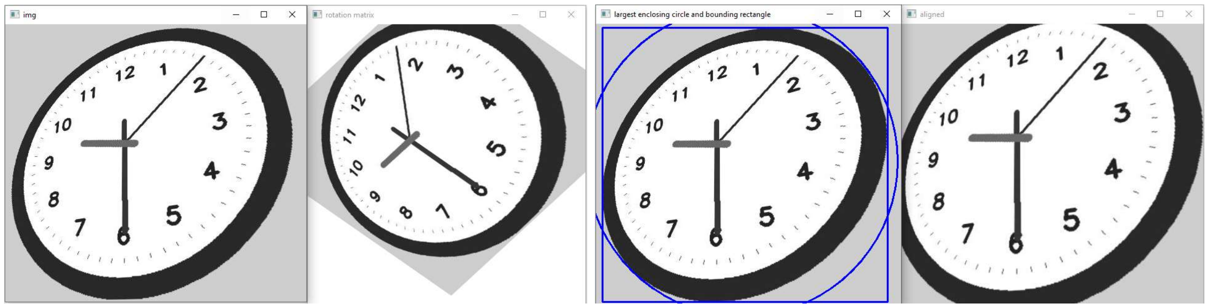
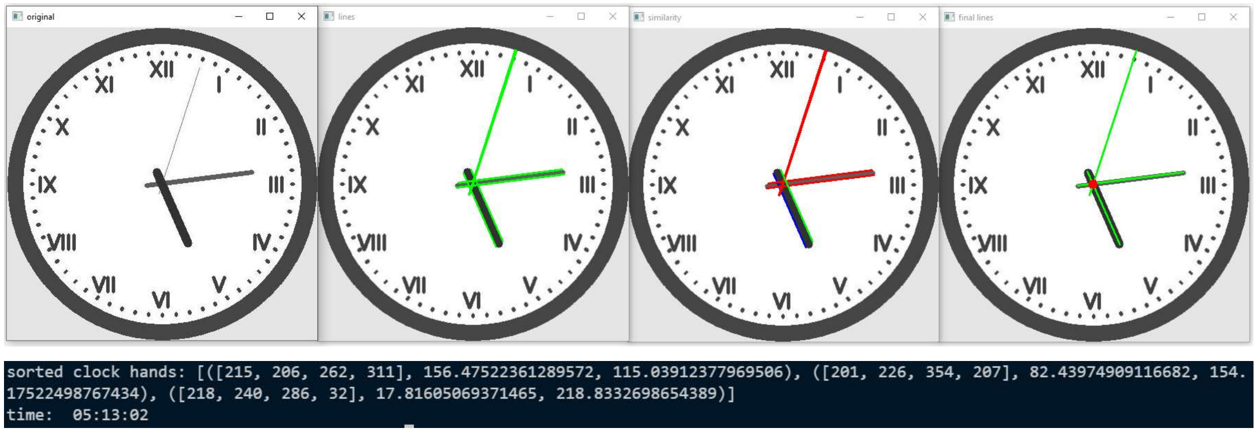


Figure 4: results after applying warps based on 1. Fit ellipse and 2. bounding rectangle + largest enclosing circle

As shown in Figure 4, both proposed methods do not achieve a proper deskew image. While using the ellipse method warps the clock into a circle as expected, the resulting clock is still at an angle (as demonstrated by the difference in border thickness) and there is no clear way to rotate the image such that the numbers are in the correct location. A possible solution to the rotation problem would be to use a forum of character recognition for the arabic and roman numerals and calculate a homography matrix from the respective bounding boxes and rectangle calculated from the coordinates. Similarly, if the input consists of square or rectangular clocks, a corner detection method such as Shi-Tomasi combined with a similar rectangle calculation would result in a properly deskew image. As a result, skew and viewpoint variation was discarded for this aspect of the project and the parse_clock program only works with minimally-skewed or flat clock input images.



```
sorted clock hands: [([215, 206, 262, 311], 156.47522361289572, 115.03912377969506), ([201, 226, 354, 207], 82.43974909116682, 154.
17522498767434), ([218, 240, 286, 32], 17.81605069371465, 218.8332698654389)]
time:  05:13:02
```

Figure 5: Hough lines, grouping lines based on angle similarity, average of each group and detected time

The algorithm relies heavily on the detection of lines using the Hough transform. Depending on the characteristics of the image, the thresholds and other parameters must be adjusted. Since the dataset was generated using known values for lengths and thicknesses, the current values may not be applicable for real world clock images. The current implementation is also not well suited in the cases where multiple hands align as only a single line is detected if the hands are of similar thickness.

4.2 CNN

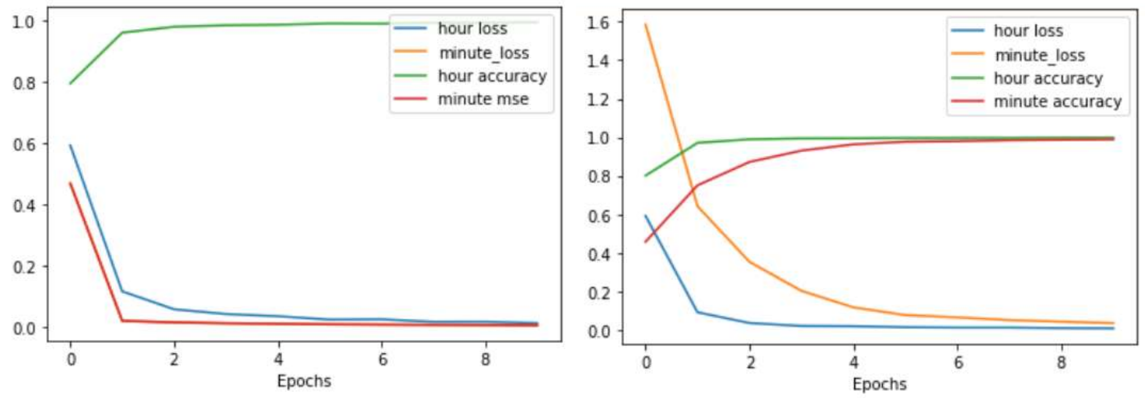
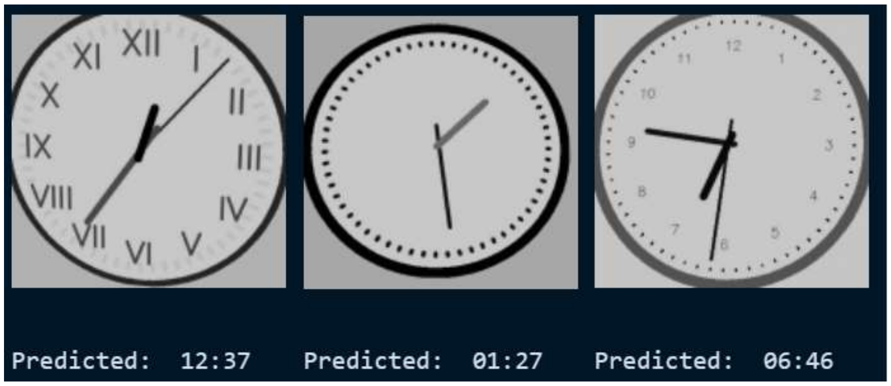


Figure 6: Training loss and accuracy over 10 epochs comparing regression and classification for minute

As evidenced by the above graphs, both regression and classification are appropriate as the loss and mse quickly converge to 0 within the 10 epochs. The training time was ~10% faster when using classification for minute values. Using regression, normalizing the minute value so that the value is between [0, 1] reduced training time significantly.



```
Predicted:  12:37    Predicted:  01:27    Predicted:  06:46
```

Figure 7: Example predictions using model trained with skewed images.

Like with the feature extraction approach, aligned clock hands can lead to predictions significantly off from the expected value. However the rate of such error is far less in the CNN model prediction.

5. Discussion

5.1 Implications

The CNN model implemented for the project is able to predict the time of the test clocks with reasonable accuracy for skewed images while the OpenCV program provides a baseline implementation for a feature extraction based detection algorithm that requires some tuning for certain input images.

5.2 Limitations

The current clock generation algorithm does not account for variations in shadows, artifacts, illumination, deformation, background clutter, occlusion and other intra-class properties (such as rectangular clocks, clocks with multiple faces, different hand types) and therefore is not a good representation of real world clock examples and as such, the trained model has room for improvement with a more comprehensive dataset. The current feature extraction program does not implement a reliable deskew and rotation correction algorithm, which is required as the hand angle calculation and resulting conversion to time assumes that the input is correctly oriented. Furthermore, hand detection is not robust in cases where hands overlap, particularly when dealing with 3 hands (such as 12:30:30).

5.3 Direction for future work

As mentioned in the limitations, the clock image dataset consists of very clean generated images and therefore isn't a good representation of real world clock images. Therefore, a next step would be to generate images with more variation as well as taking images from a dataset of real images. For real images, there would need to be a function to detect and crop out the clock itself (such as YOLO object detection) as well as a way to label the corresponding hour and minute. Different methods of calculating an appropriate homography matrix need to be explored for further development of the OpenCV only program.

Resources

[1] Krizhevsky, Alex & Sutskever, Ilya & Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.
[2] Adrian Rosebrock. (2018). Keras: Multiple outputs and multiple losses. Pyimagesearch. https://pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses
[3] https://stackoverflow.com/questions/23240141/unwarp-curved-surface