https://doi.org/10.2352/EI.2023.35.7.IMAGE-282

Lightweight Single Pass Numerical Reading Extraction for Displays in the Wild

Shanmukha Yenneti, Yan-Ming Chiou, Bob Price

Abstract

Although considerable progress has been made in recognizing multi-character text from images, there are still cases where there is a lack of robust computationally-efficient methods that can execute on portable devices to read device displays in the wild. We specifically address the problem of parsing digits from 7 segment displays. Recognizing these displays is important for many tasks such as assisting users with tasks using augmented reality agents that need to verify actions or connecting legacy devices to the internet for process control using cheap cameras. Legacy techniques based on image processing operators and OCR are brittle whereas massive deep networks are too computationally expensive. We describe a computationally tractable VGG style backbone combined with a novel digit inference head that can be trained using a synthetic display generator with novel augmentations. We show the model trained on augmented synthetic data generalizes well to a corpus of real-world display images getting 97.8% single-frame accuracy and obtaining a throughput of 30 frames per second. We describe how the output can be further stabilized to improve accuracy through a kind of mode filtering.

Introduction

The ability to read digital displays can be helpful in a variety of real-world situations where the user needs data to be automatically filled out or would benefit from having display readings verified. A phone-based app, for instance, might automatically read the display on a glucose meter in a health-care application and record the value for the user. During cooking, visual analysis could be used to confirm that the scale was fully zeroed and the ingredient weight was consistent with cookbook requirements. Since display readings can have hundreds or thousands of different values, it is not practical to simply train a classifier to output a discrete label for each possible appearance. Thus, there is a need for recognizing and parsing texts in the wild, but not much work has been done for single step digit recognizing. Traditional methods first perform image pre-processing such as image binarization, thresholding and remove gaps in characters fonts using erosion techniques. Then, they segment digit candidates followed by the classification of individual digits in order to recognize the entire text/ number. We propose a model that uses a VGG16 [1] style network as a feature generator combined with a novel digit extraction head which extracts features and outputs digits in sequence including punctuation such as decimal points for fractional numebers and colons used in times. In the following sections of the paper we describe the dataset creation, model architecture, results and conclude by laying out the limitations & future work.

Related Work

Although many works have been published in OCR (Optical Character Recognition) of documents, not much attention is given to recognition and parsing of 7-segment displays in the wild. Existing software solutions like ssocr3 (Seven Segment Optical Character Recognition) [5], have trouble with recognition in displays in unsuitable orientations and distortions in the image. This does not work well to support recognition in images from dynamic ego-centric cameras in real life situations where camera angles can be oblique, lighting varies and motion blur occurs. Amazon Rekognition [7] and Google's Tesseract4 [6] are marketed more for general text recognition. Google's Tesseract4 [6] is able to read a 7-segment digits in a ideal settings but struggled with low contrast displays and when there is distracting text near the displays. A few research works have attempted to solve this recognition problem by considering it as a multi stage problem. In an attempt to recognize data in medical monitoring devices, [3] performs feature extraction through image processing techniques like binarization, noise removal and individual digit segmentation after which the features are passed through a machine learning algorithm to perform classification. In a similar work [4] proposed the usage of robust object detectors like YOLOv5 to detect and recognize individual digits in order to read the displays on pulse oximeter devices. A second phase is needed to orient and chain together the detections.

Approach

Although some of these existing solutions give good results, none of them provide a single step 7-segment digit recognition method which will be able to generalize these results to a more general category of seven segment displays. With this motivation, we devised a single stage seven-segment digit recognizer and also created a comprehensive dataset that contains samples with most distortions found in the real world.

Dataset: 7-Segment Displays in the Wild

Based on our research, no publicly available labelled dataset containing images of seven segment displays in the wild were found. Although we could find MNIST [2], it only deals with handwritten digits and does not provide any data for multi-digit recognition.

This motivated us to build a dataset from scratch. Generating our own images from actual devices would not allow us to attain the level of generality we sought. Hand labeling images of digital appliances proved difficult as displays are often tiny in images and labeling is tedious. The created dataset contains images that imitate real world devices having 7-Segment LCD displays. We used a high quality parametric online LCD display simulator to generate a large variety of display images

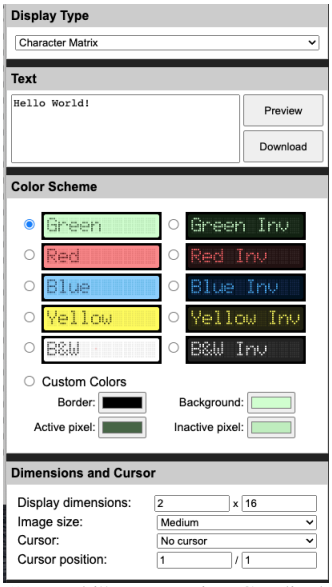


Figure 1: The “avtanski” parametric LCD display synthesizer

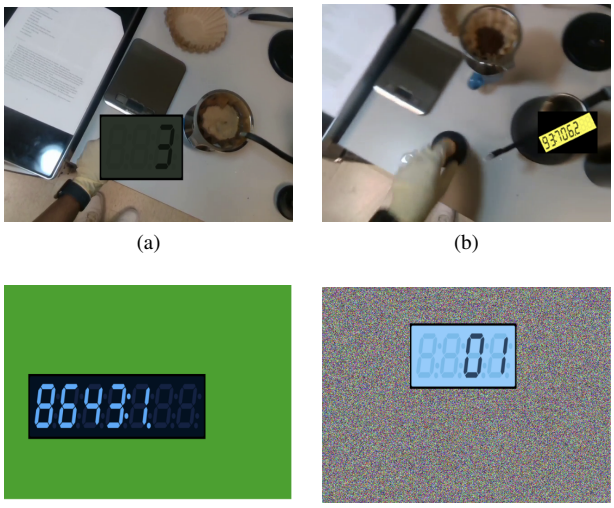
(http://avtanski.net/projects/lcd/). The simulator allows for different colors of background and foreground, different sizes of displays and several style attributes. We note that the capacity of a display may be different than the number of digits displayed. For instance, we might have a 5-digit display that is only displaying 3 digits in a particular image with the remaining two potential digits left blank. Notice that the simulator also captures the faint background element of seven segment display positions even when those segments are not active. We created a loop to systematically sample a diverse set of image in various sizes, styles and colors.

We considered digits 0,1,2,3,4,5,6,7,8,9 and characters ’.’ & ’:’ that are commonly found in displays to show decimal and time values respectively. Using these 12 symbols (both digits and characters together will be referred to as digits in the paper from now) we generated 25,500 random multi-digit numbers that have a randomly chosen length between 1 and 8. Sample generated displays can be seen in Figure 2. We also had a place holder symbol for empty digits.

Figure 2: Randomly generated numbers (with punctuation characters) on various display backgrounds in random colors and fonts.

In order to make our training data as similar to real world 7-segment LCD displays, we resize these 7-segment display readings and the following augmentations randomly:

- Random resizing to include tiny displays
- Random rotation up to 60 degrees
- Randomly shifting Contrast, Brightness, Hue, Saturation
- Randomly pasting on random backgrounds like such as real world images, random noise, solid colors



(a) (b) (c) (d)

Figure 3: Sample images from the training set consisting of displays in various alignments, contrasts, backgrounds and display samples.

To cater for all kinds of distortions in the real world, additional augmentations like gaussian blur, motion blur and affine transformations were performed on the images. Affine transformations simulator the perspective distortion seen when displays are viewed from an angle. No vertical or horizontal flips were made because that would possbily change the meaning of the digits (e.g., ”25” becomes ”52”). We also limit rotations to less than 60 degrees for the same reason. All images have displays reading from left to right. Figure 3 shows a sample set of images from the dataset that is used for training our model.

The labels for these images are of the format $[d_1, d_2, d_3, \ldots d_8]$ where d_1 to d_8 are the digits being shown in the display area in the image. As described above, these digits can be any of the 10 numeric digits, or the symbols ’.’ and ’:’ in the order of digits as shown in the corresponding image. The label for numeric digits is encoded as the value of the digit while ’.’ and ’:’ are encoded as ’10’ and ’11’ respectively. As we have images that have length of numbers shown in the image between 1 and 8, we use a dummy digit code ’12’ to represent the absence of a digit. Table 1 gives examples of encoding for several values.

Value	Label Encoding								Length
’10:42’	1	0	11	4	2	12	12	12	5
’:’	11	12	12	12	12	12	12	12	1
’20.3’	2	0	10	3	12	12	12	12	4

Table 1: Sample encodings for several image values

Model Architecture

The inference was done using a deep neural network architecture show in Figure 4. The digit recognizer is based on the VGG16 backbone [1] which is an older but smaller and computationally efficient model suitable for deployment at the edge. We found that we needed to add a couple of fully connected non-linear layers at the end to integrate over the features available in the last layer of the VGG network to get good performance. These non-linear layers act as features for 8 one-hot categorical units

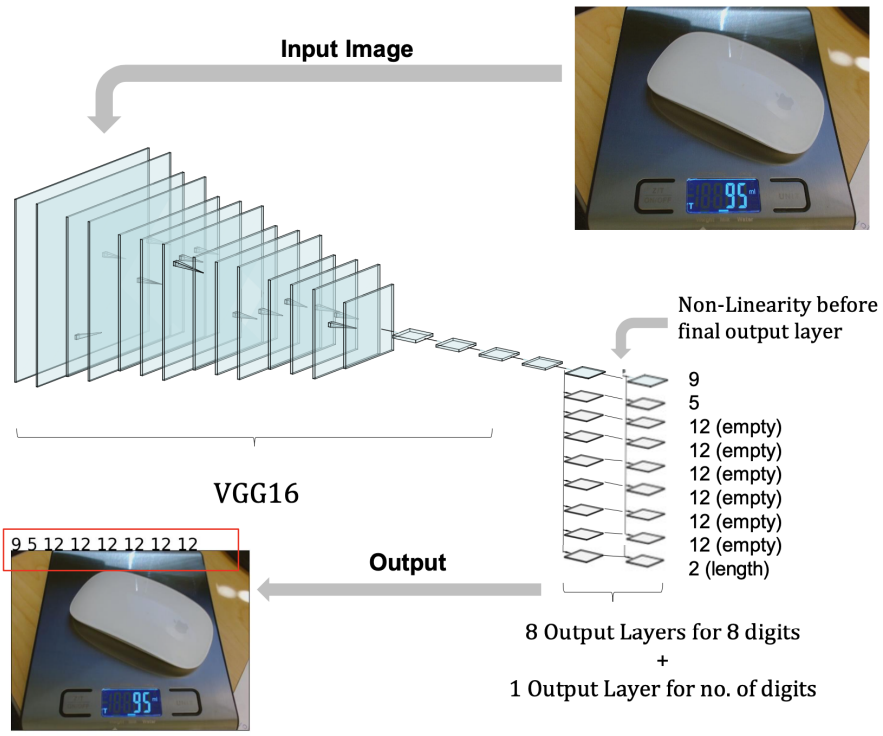


Figure 4: Proposed Architecture for 7-Segment Multi-Digit Inferencing

that when fully trained output the observed digits (or a blank) in order from left to right in the image in a single step without any reordering or grouping required. In addition to specific digits, the network also outputs a continuous length for the target number in digits using a linear unit. So a reading of 495 would have length 3. We found that including the length as well as the specific digits helped the network to get the correct number of non-blank categorical digits.

The categorical units were trained with standard cross-entropy loss against the ground truth digits from the simulation and the loss on the length prediction was trained using mean squared error. The loss function for the prediction of a single multi-symbol value is

$$Loss(g,p,x,y) = -\sum_{i=1}^{D}\sum_{j=1}^{M} g_{i,j}\log(p_{i,j}) + (x-y)^2 \qquad (1)$$

where D is number of predicted digits; M is the number of possible values for each digit including punctuation and spaces (12); g is the ground truth label and g_i is the i^{th} digit of the label and $g_{i,j}$ is one for exactly one possible value of the i^{th} digit and zero otherwise; p is the network's predicted softmax probabilities over digit values, p_i is i^{th} digit and $p_{i,j}$ is the probability assigned to the j^{th} possible value for the i^{th} digit; x is the actual length and y is the predicted length.

A post inference cleanup procedure is used to improve accuracy. For instance, for readings with trailing colons and decimals, the filter removes the final punctuation symbol. This sometimes occurred if there was additional text on the housing next to the display.

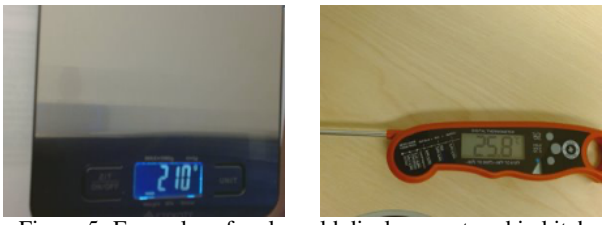


Figure 5: Examples of real-world displays captured in kitchen

Results

We tried using both pretrained and trained-from scratch VGG backbones. We observed that VGG trained from scratch performed better. Network was trained for 300 epochs on a NVIDIA Titan XP for about 6 to 7 hours. The CUTMIX augmentation was used to improve generalization of the trained model.

Results are presented here on network trained end-to-end from scratch. We evaluated the network on a hold-out validation set of synthetic images and observed that training error and hold-out validation set were similar suggesting the network was not dramatically overfitting within the synthetic data. We also used the validation set to guide the number of epochs of training. In Table 2, we can see that the digit accuracy and number length estimation accuracies are high and very close suggesting the network converged but didn't badly overfit. Synthetic training and testing runs did not use post inference filtering of trailing punctuation.

The trained network was evaluated on a real-world image test set taken from kitchen appliances. None of these images were included in the training set. Examples of real-world images are shown in Figure 5.

In Table 2 one can see that the network performs well on

Dataset	Digit Accuracy	Length Accuracy
Synthetic training	100.0	88
Synthetic hold-out	99.4	98
Real-world images	92.0	88.3

Table 2: Digit inference network performance after 300 epochs of training

real world images despite not having any training on real world images. The real world inference results made use of filtering for trailing punctuation.

We were not able to obtain results for embedded inference performance at the time of publication, but did observe 30 FPS performance of the model on an older NVIDIA 1080 card.

These results are for inference of digits from single images. The next section discusses sequences of images.

Post Inference Stabilization

In practice, we are often able to obtain a stream of images from a camera instead of a single image. One might be tempted to simply perform inference on a number of frames and then average the results over a window or use exponential averaging but this turns out to be suboptimal. We observed that the errors or noise in detections arises from two distinct sources. There is some underlying noise in the process being measured. In a scale, tiny vibrations might cause the exact measured weight to vary continuously. In many cases, process noise is approximately Gaussian and may be smoothed by simply averaging. However, in our application, there is also noise in the visual inference process. Visual noise is not Gaussian and simple averaging is generally not the optimal solution. Consider the case of the digit 1. If a shadow causes inference to hallucinate a top bar the 1 could be detected as a 7. Even if the 1 is detected more frequently, the average would still be significantly off. Imagine 10 instances where a 1 is detected 9 times and a 7 is detected once leading to an average of $1*9 + 7*1 \,/\, 10 = 1.6$ which is not a valid digit, but would be rounded up to 2 resulting in an error of 100%. We designed a simple mode-filter to address this by histogramming the individual digits over a time window and choosing the most frequent value for the digit. Because each digit only has 12 values this remains tractable and practical. The stabilized digits are then assembled into a number which is much more accurate than an estimate of the number from a single frame.

Conclusion

In this paper, we proposed a method for reading digits (on 7-segment displays) which uses a VGG-16 backbone to create visual features and two layers of non-linear fully connected units and 8 categorical symbol units to generate a number in a single pass from an image. We found that predicting and training on the length of the number improved accuracy. The ability to generate a large corpus of digits with a simulator and augmentations proved to be critical to training a robust model in the absence of existing large datasets. We further determined that digit-level mode filtering could be used to improve inference over multiple frames in applications such as task monitoring. Initial results suggest good real time performance for on-line applications.

Limitations and Future Work

Currently the model's digit inferring ability is limited to 8 digits maximum. This limitation can be altered by including more or less units, but a more general solution would be to integrate a sequential model like an RNN. Also the model is trained only on the digits and a few punctuation symbols like '.' and ':' so, inferring any units of measurement on the display will not be inferred or, in worst case scenario, be inferred as an extra digit. In our observations we found that when there is text or numbers present around the display area, the model outputs extra digits or wrong digits. This can be attributed to the fact that the training data does not contain samples that have 'distractions' around the simulated display areas in the image and therefore this problem could be alleviated by improving the samples in the training data in way that simulate more real world distractions.

Acknowledgements

We would like to acknowledge the Xerox corporation for funding the development of the technology here and DARPA research contract HR001122C0009 for funding the application to the kitchen cooking domain.

References

[1] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[3] Shenoy, Varun N., and Oliver O. Aalami. "Utilizing smartphone-based machine learning in medical monitor data collection: seven segment digit recognition." AMIA Annual Symposium Proceedings. Vol. 2017. American Medical Informatics Association, 2017.

[4] Boonnag, Chiraphat, et al. "PACMAN: a framework for pulse oximeter digit detection and reading in a low-resource setting." arXiv preprint arXiv:2212.04964 (2022).

[5] Auerswald E. Seven Segment Optical Character Recognition, 2008. Available at: https://github.com/auerswal/ssocr

[6] Google's Tesseract OCR. Available at: https://github.com/tesseract-ocr/tesseract/

[7] Amazon Rekognition. Available at: https://aws.amazon.com/rekognition/

Author Biography

Shanmukha Yenneti received his MS in Computer Science from Stony Brook University (2022). His work is focused on computer vision applications in medical image analysis and assistance systems.

Yan-Ming Chiou is a member of scientific staff at the Palo Alto Research Center, where he explores augmented reality and computer vision applied to AR assistance systems. He received his Ph.D. from the University of Delaware.

Bob Price is a Principal Scientist at the Palo Alto Research Center who explores computer vision applications for real world tasks and fundamental problems in learning representations from sensor inputs. He holds a PhD from the University of British Columbia (2003) and a Post Doctoral Fellowship from the University of Alberta (2006).