Typing Characters Recognition in AOLME WebApp

Ugesh Egala, Department of ECE, UNM, Marios Pachittis, *Senior Member, IEEE*

***Abstract*—This paper aims to detect and recognize the characters that are being typed in AOLME WebApp. The frame difference technique helps us to locate regions with differences in the content between two successive frames. The regions with potential changes are fed to the CNN model which is trained on the EMNIST dataset to recognize each letter in the corresponding regions.**

***Index Terms*—Character Recognition, CNN, PyTroch, AOLME**

I. Introduction

The Advancing Out-of-School Learning in Mathematics and Engineering (AOLME) project is after school initiative which aims to support the learning and active engagement of middle school students in mathematics and engineering-related activities. AOLME WebApp is one such tool that enables kids to play interactively with Binary or Grayscale or Color images. The kid's interaction with the teacher, peers, and interactive tools are recorded. Applications of Image and video process techniques on recording videos, enable us to provide useful insights to teachers whether kids can follow the program or how they are interacting with these tools. By recognizing the characters that are being typed in AOLME WebApp, we can know what kids are typing. This has application in low bandwidth video communications. Information between sender and receiver is shared only when there is a change in content in the current frames.

II. Background

Character recognition helps us to convert handwritten documents into digital form. This has made the storage and retrieval of information easier. Machine typed characters have uniform dimensions, font, style, and positions of characters. In contrast, handwritten characters look different for different users. Character recognition is a harder problem in the handwritten case when compared to machine typed characters. Optical Character Recognition (OCR) has evolved over the years in recognizing the characters and text from images. It is hard to add customized pre-processing functions and custom trained model into OCR software. Therefore, we would be using a model trained on the Convolution Neural Network to recognize the characters.

III. Method

Read two successive images from screen recordings of AOLME Webapp, current image, and previous image. By frame difference and morphological operations like erosion and dilation with two (horizontal and vertical) rectangular structural elements, we can find a rectangular region where typing is happening. Extract character' in a rectangular region from the current and previous image, compare corresponding characters. If there is a mismatch, then recognized characters in the current image are displayed as output.

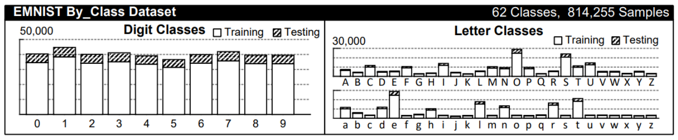


Fig. 1: Visual Breakdown of EMNIST Dataset

Algorithm 1 Character Recognition

```
while hasFrame(video) do
    curImg, prevImg = readFrame(video)
    curImgGray, prevImgGray = threshold(curImg, prevImg)
    curBImg, prevBImg = erodeDilate(curImg, prevImg)
    curImgCtr, prevImgCtr = findContours(curBImg, prevBImg)
    curImgBBox, prevImgBBox = roiHeustric(curImgCtr, prevImgCtr)
    matchedBBox = centrodDist(curImgBBox, prevImgBBox)
    for length(matchedBBox) do
        curImgPatch, preImgPatch = crop(matchedBBox)
        if nnz(curImgPatch - preImgPatch) then
            curImgBinaryPatch, prevImgBinaryPatch = threshold(curImgPatch, preImgPatch)
            curImgPatchCtr, prevImgPatchCtr = findContours(curImgBinaryPatch, prevImgBinaryPatch)
            for ctr in curImgPatchCtr do
                segChar = cropChar(curImgBinaryPatch(ctr))
                predictChar = classifyChar(segChar)
                segCharPrev = cropChar(prevImgBinaryPatch(ctr))
                predictCharPrev = classifyChar(segCharPrev)

            end
            if matchedChar(predictChar, preditCharPrev) then
                display(predictChar)
            end
        end
    end
end
```

IV. Results

A. Dataset

EMNIST is a handwritten dataset with 62 categories. They are alphabets capital and small, and numbers from 0 to 9. Fig 1 depicts the number of images per each category and the size of the dataset.

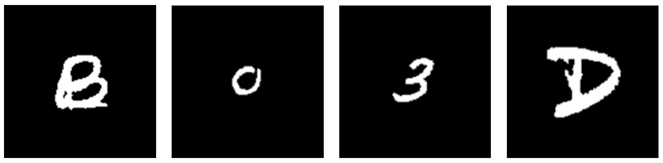
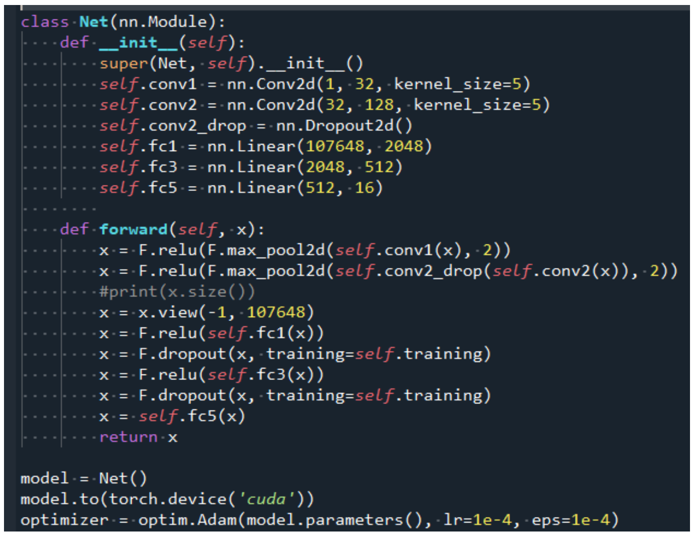

(a) Predicted: B (b) Predicted: 0 (c) Predicted: 3 (d) Predicted: D

Fig. 2: Sample Images and their Predictions

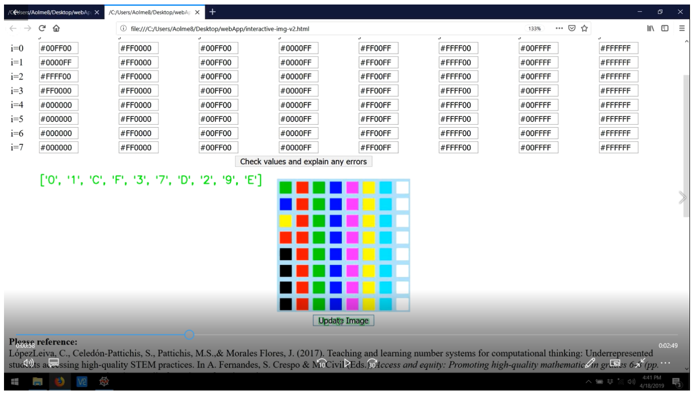


```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=5)
        self.conv2 = nn.Conv2d(32, 128, kernel_size=5)
        self.conv2_drop = nn.Dropout2d()
        self.fc1 = nn.Linear(107648, 2048)
        self.fc3 = nn.Linear(2048, 512)
        self.fc5 = nn.Linear(512, 16)

    def forward(self, x):
        x = F.relu(F.max_pool2d(self.conv1(x), 2))
        x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
        #print(x.size())
        x = x.view(-1, 107648)
        x = F.relu(self.fc1(x))
        x = F.dropout(x, training=self.training)
        x = F.relu(self.fc3(x))
        x = F.dropout(x, training=self.training)
        x = self.fc5(x)
        return x

model = Net()
model.to(torch.device('cuda'))
optimizer = optim.Adam(model.parameters(), lr=1e-4, eps=1e-4)
```

Fig. 3: Neural Network



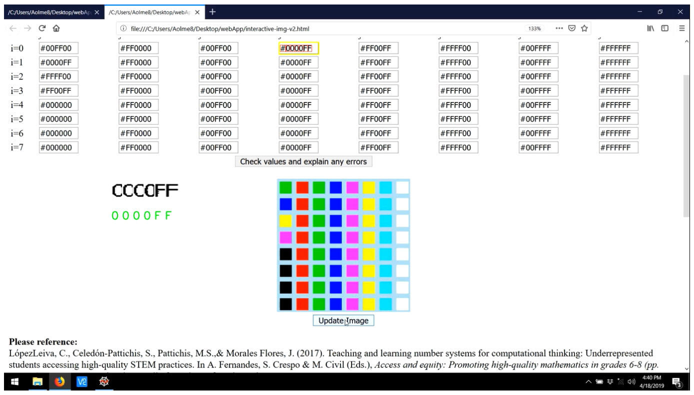
file:///C:/Users/Aolme8/Desktop/webApp/interactive-img-v2.html

i=0	#00FF00	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=1	#0000FF	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=2	#FFFF00	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=3	#FF0000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=4	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=5	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=6	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=7	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF

['0', '1', 'C', 'F', '3', '7', 'D', '2', '9', 'E']

Please reference:
LópezLeiva, C., Celedón-Pattichis, S., Pattichis, M.S.,& Morales Flores, J. (2017). Teaching and learning number systems for computational thinking: Underrepresented students accessing high-quality STEM practices. In A. Fernandes, S. Crespo & M. Civil (Eds.), *Access and equity: Promoting high-quality mathematics in grades 6-8* (pp.

Fig. 4: Prediction Failure



file:///C:/Users/Aolme8/Desktop/webApp/interactive-img-v2.html

i=0	#00FF00	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=1	#0000FF	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=2	#FFFF00	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=3	#FF00FF	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=4	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=5	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=6	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF
i=7	#000000	#FF0000	#00FF00	#0000FF	#FF00FF	#FFFF00	#00FFFF	#FFFFFF

0000FF

Please reference:
LópezLeiva, C., Celedón-Pattichis, S., Pattichis, M.S.,& Morales Flores, J. (2017). Teaching and learning number systems for computational thinking: Underrepresented students accessing high-quality STEM practices. In A. Fernandes, S. Crespo & M. Civil (Eds.), *Access and equity: Promoting high-quality mathematics in grades 6-8* (pp.

Fig. 5: Successful Prediction

B. Results

1) Training: WebApp works with hexadecimal digits. So, we made use of only 16 classes from 62 classes of EMNIST. They are numbers from 0 to 9 and capital letters from A to F. Internal structure of neural networks developed in PyTorch is depicted in Fig 3. The neural network is trained on Nvidia GeForce GTX 1080. The network operates on the image of size 128*128 pixels and outputs a hexadecimal character. Evaluation of trained model on few sample training images is displayed in Fig 2.

2) Testing: Since there is redundancy between successive frames in screen recording videos, we would be operating at 1 frame per second. The current method cannot handle if there is a drastic change in content between the frames. Fig 4 shows one of the failure cases, Inputs characters such as 'U', 'p', 'd', 'a' etc. are wrongly classified. Characters '0000FF' are accurately classified in Fig 5. Effectiveness of the trained model can be seen in the videos listed in the appendix.

V. Discussion

This approach cannot handle exceptions like errors or warnings messages because they create a large difference in content between two successive frames. The proposed method has fixed thresholds for binarizing the image, restriction on the size of the structural element, heuristics such as area, width, and height on characters bounding boxes. If these thresholds and restriction are automated based on the image, current methods work for different inputs as well. The existing neural network is trained only on hexadecimal digits. To create a generic character model, it should be trained in 62 classes.

VI. Conclusion

We were able to detect and recognize characters that are being typed in WebApp. The accuracy of a trained model could be tested based on the availability of ground truth. Model robustness can be improved by training with characters of different scaling, font sizes, and styles.

Appendix

- Detection and Recognition on WebApp:- https://unmm-my.sharepoint.com/:v:/g/personal/ugeshe_unm_edu/ESsZHRYGG6xAoCxwVJiFSasByNeJ8JEKOPJ0GhJwCyRB3w?e=G4EeRl
- Detection and Recognition on Single Image:- https://unmm-my.sharepoint.com/:v:/g/personal/ugeshe_unm_edu/ERPP8vDSHplPgHJ6DmeNvFcBcK3_deI9n_dyQgLi6pUpfA?e=qh97bt
- AOLME:- https://aolme.unm.edu/
- AOLME WebApp:- http://ivpcl.unm.edu/ivpclpages/Research/aolme/app/interactive-img-v2.html
- Pretrained Model:- https://github.com/gaurav0651/emnist/blob/master/train_emnist.ipynb
- PyTorch tutorial:- 'https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

References

[1] Gregory Cohen, Saeed Afshar, Jonathan Tapson and André van Schaik. EMNIST: an extension of MNIST to handwritten letters, 2017; arXiv:1702.05373.