FINGER PEN : AN AIR WRITING AND RECOGNITION SYSTEM

Xu Xuanbo, Ying Zhuozuo, Zhong Yuting

Institute of Systems Science, National University of Singapore, Singapore 119615

ABSTRACT

Signing and writing are inevitable events during social communication. However, this common behavior becomes high risky in public because of the pandemic. To solve this problem and decrease the spread of COVID-19, we have an idea to create a vision writing system. In this paper, we use MediaPipe hand detection API to detect people's hands in the camera. Based on finger landmarks, we can recognize users' gestures using vector included angles and achieve different control effects like writing, erasing, choosing and etc. After getting users' hand-writing, we combine trajectory-based and image-based methods to make recognition. For trajectory-based method, a double-layers stacked LSTM model trained by Char74k dataset is used for predicting. For image-based method, received trajectories from writing module are transformed to images using Bézier curve and a CNN model trained by EMNIST dataset is used for predicting. After recognition, we also use modified HMM and Bayesian model to correct recognition result based on word frequency. During all contact-less operating processes, users only need to use hand and gesture to determine when to write, erase, save and recognize based on our operating system and choose desired result within both recognition result and correction result.

Index Terms— Gesture Control, Trajectory Recognition, Image Recognition, Word Correction

1. INTRODUCTION

During this epidemic period, signing and operating the computer in public become high risky and inconvenient because of the coronavirus. It is a common choice using sanitizer to wash hands after contacting public pens or computers when necessary. But our team want to investigate whether it is possible to write or draw in the air using a camera, rather than a pen. And it will be very helpful if we can recognize what user has written and return a result. So that all procedures will be contact-less and it can help to decrease the spread of COVID-19.

The core part of writing is hand detection. It should be ultrafast and accurate so that we can get steady live stream and display handwriting in real time. We find it hard to get ideal datasets as MediaPipe uses and we are not sure about the performance of our self-designed model. So we decide to call API instead. MediaPipe hands solution uses frame as input, and output finger landmarks on it. So that we can use coordinates to calculate included vectors for each finger and recognize users' gestures based on that. Then a functional control system which uses gesture only to realize operating processes are generated. Following the instruction, users can use hand only by performing different gestures to write letters or numbers, erase wrong writing, save writing trajectory and get recognition result.

To make recognition, it is straight forward to use generated images to recognize. But since we can get the sequence of finger tip movement and use this to display handwriting, we can also use the trajectory of index finger tip to recognize what users have written. Go a step further, we decide to merge these two methods together to get a better recognition result. For trajectory-based recognition, we use trajectory numpy array as input and use a deep learning model to output a word as recognition result. Comparing the performance of CNN, single-layer LSTM, BiLSTM and double-layers stacked LSTM, we finally choose double-layers stacked LSTM to make recognition and using dataset Char74k as training data. For image-based recognition, we transform the trajectory dots to image using Bézier curve as input, and use a CNN model trained by EMNIST dataset to output a recognition result. These two recognition methods are then merged together using late fusion to make a final recognition result.

Another problem is that, there is always a gap between users' initial thought of what they want to write and what they actually wrote on the screen. Less typical writing will result in poor performance of recognition models. Then we decide to add a correction module based on common word frequency and predict what user want to write actually. We use a weighted summation of two recognition models' last layer outputs before activation function as input, and use modified HMM and Bayesian model sequentially to get a list of candidate words as output. Then we will display top 3 results together with recognition result for users to choose.

Generally, the input to our system is frames captured by camera, and then hand-writing is generated by hand detection module in the frames and recognized by trajectory-based and image-based recognition model together to get recognition result. Finally the recognition result will go through correction module to get a correction result as our output.

2. LITERATURE REVIEW

2.1. Hand Detection

Hands are what we use most frequently to interact with the world. And how to perceive the shape and motion of hands is quite a difficult problem. To focus on system design and recognition module, we decide to call API to detect and mark finger keypoints. Generally, there are two famous pre-trained skeleton-based hand detection models, one is OpenPose[1] and another is Mediapipe[2].

OpenPose uses hand keypoint detection model[3] to detect hand skeleton. The captured image is sent to the hand key point detector to obtain many rough key point detection results first. Once there are key points from different perspectives of the same hand, a key point triangulation is constructed to obtain the 3D position of the key points. But OpenPose need to detect the body region first and base on the connection to get hand region. As a webcam may only be able to capture half of the users' body, this solution does not meet our expectation and may not be so fast as wish.

As for Mediapipe, it uses a palm detection model first to operate the full image and return an oriented bounding box of hands. And then use hand landmark model to perform precise keypoint localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression. Mediapipe achieves an average precision of 95.7% in palm detection. Using a regular cross entropy loss and no decoder gives a baseline of just 86.22%. During our tests, MediaPipe offers a more fast operating on single hand, which is important for us to display real-time writing on the screen.

2.2. Trajectory-based Handwriting Recognition

Trajectory-based methods play an important role in Handwriting recognition problem. The slope of writing trajectory is considered as an effective feature for digits or letters recognition, which can explicitly display the direction and speed of writing. Based on this sight, [4] proposed a feature vector extraction method called slope variations detection with size, translation, and rotation invariance. Dynamic time warping (DTW) can measure the similarity between two time series. Therefore, the trajectory classification problem can be solved by applying distanced-based classification model, such as fast nearest neighbor [5], or clustering model, such as agglomerative hierarchical clustering [6]. Hidden Markov model (HMM) is another popular technology widely used in trajectory recognition and letter parsing tasks [7].

Despite good performance, traditional methodologies still suffer from poor generalization ability or cumbersome steps. Recently, some researchers have also turned to End-to-End Deep Learning Architecture to address these problems. Convolutional neural network (CNN) is used in 3D trajectory recognition for its different variants and robustness [8]. Another deep neural network designed to process time series data called long short-term memory (LSTM) also demonstrated good potential for trajectory recognition with the help of depth information [9]. Inspired by these previous works, we applied CNN and LSTM on 2D trajectory data instead of 3D data in this project, which would be easier to be applied and further promoted.

2.3. Image Handwritten Recognition

Image-based handwriting recognition is an essential subfield of optical character recognition (OCR) and extensive research works have been carried out with different proposed techniques. There are a significant amount of works tested over MINST database ranging from using traditional machine learning methods to deep learning methods. Since the MINST is considered to be already solved, more researches are conducted on Extended MINST database (EMINST) introduced in April 2017 constituting a more challenging benchmark. A baseline using linear classifier and OPIUM (Online Pseudo-Inverse Update Method) introduced by van Schaik and Tapson [10] was provided by the author of EMNIST paper [11]. This technique generates an analytical solution which is deterministic. Other traditional machine learning methods like k-nearest neighbors (K-NN) and support vector machine (SVM) with discrete wavelet transform and discrete cosine transform have been tested with successful results [12].

Though traditional machine learning techniques generates acceptable results, outstanding achievements are achieved by techniques belonging to the filed of deep learning. An early attempt to use deep learning was using Markov random field-based CNNs [13]. Later, CNNs with different numbers of convolution layers and dense layers were proposed. For example, a CNN with two convolution layers and one dense layers for classifying EMNIST with the purpose of developing a handwritten text classification application, obtaining an accuracy over 85% [14]. Since CNN hyper parameters take time for tuning, some researchers have made used of neural architecture search for automatically optimizing the hyper parameters of CNN classifiers. More recently, committees of neuroevolved CNNs using topology transfer learning have been used to obtain an accuracy higher than 95% on separate digits and letters data set [15]. It has complicated CNN design and high computation complexity. Considering about project scope and target, we applied CNN with convolution and dense layers in this project instead of committee CNNs, which is easier to design and has relative high accuracy.

2.4. Air-drawing Github Project

Air-Drawing is one of our initial idea source[16]. It also uses MediaPipe toolbox to perform handpose detection and customers can use index finger to draw on the canvas. What makes our project different from it is that air-drawing uses the speed and acceleration of the finger to make the prediction of

translation-invariant, and output a binary classification 'pencil up' or 'pencil down' to predict and modify user's drawing. As its deep learning predicting model does not perform well, this function can not achieve practical use. But we focus on hand-writing recognition, to recognize what user writes, then use a correction module to correct recognition result and offer more possible choices for users to choose. We trained the model to get high accuracy and this function can be used to recognize signature or some important messages. Moreover, our writing system is greatly more complex than air-drawing, which contains various gestures to implement different control laws and makes our system both interesting and practical.

3. DATASET

3.1. Trajectory-based Handwriting Recognition Dataset

We use Char74k dataset [17] as our trajectory dataset, which contains 3410 trajectories of characters written by hand using a tablet PC. Despite its trajectories amount is less than that of other datasets', it contains all commonly used characters in English including 0-9, a-z, A-Z (55 trajectory samples for each character). The trajectory is composed of a series of points and is represented by a 2-dimensional (2D) matrix. Each row in the matrix represents the coordinates of a track point on the tablet PC. During training, we randomly split the training set and the test set at a ratio of 80%-20%. Unlike air writing, the strokes of characters are not continuous because they are written on a tablet. In addition, the sample size may not be enough for deep neural network training. All of these will be further discussed in Section 4.

3.2. Image-based Handwriting Recognition Dataset

For image-based handwriting recognition, We use Extented MNIST (EMNIST) database [11], which consists on both handwirtten digits and letters. The EMNIST is drived from NIST Special Database 19 (NIST SD 19) [18], containing NIST's entire corpus of handprinted document and character recognition, including over 800,000 labelled characters. There are six datasets that compise EMNIST dataset, which are By_Class, By_Merge, Balanced, Digits, Letters and MNIST. Table 2 shows the complete structure and organization of EMNIST. We consider using the By_Class, By_Merge or Balanced for our project since they contain both digits and letters samples. The By_Merge and Balanced only have 47 classes (37 for letters and 10 for digits) by combining some letters with similar outcome in uppercase and lower case, such as letter C and c. Since we want to identify all letters in uppercase and lowercase separately, we decide to use By_Class, which has 62 different classes (52 for letters and 10 for digits).

The By_Class dataset contain 697,932 training samples and 116,323 testing samples, including 62 different categories (0-9, a-z, A-Z). The detailed information can be seen in Table 2. A sample of the letters and digits in corresponding part of EMNIST Datasets can be found in Figure 1.

Table 1. Structure and Organization of the EMNIST Datasets

Name	Classes	No.Train	No.Test	Validation	Total
By_Class	62	$697,932$	$116,323$	No	$814,255$
By_Merge	47	$697,932$	$116,323$	No	$814,255$
Balanced	47	$112,800$	$18,800$	Yes	$131,600$
Digits	10	$240,000$	$40,000$	Yes	$280,000$
Letters	37	$88,800$	$14,800$	Yes	$103,600$
MNIST	10	$60,000$	$10,000$	Yes	$70,000$

Table 2. Structure of By_Class Dataset

By_Class	Details
Class	[0-9],[a-z],[A-Z]
Color	Black and white
Position	Centered
Image Size	28×28 pixels

Fig. 1. EMNIST Datasets Sample [15].

3.3. Correction Dataset

To extract linguistic knowledge, we also introduced the Browns corpus as the corpus of the correction module. The Browns corpus is a representative balanced corpus, which contains a variety of articles with different text styles and more than one million American English words with their corresponding parts of speech. In this project, we mainly use Browns corpus to extract bi-char-gram frequency and word frequency. The bi-char-gram frequency refers to the frequency of the combination of two characters in the corpus. And the word frequency refers to the frequency of words in the corpus.

4. PROPOSED SYSTEM

4.1. System Design

Our system is an integrated vision system as shown in Figure 2. It has three major modules: gesture recognition, handwriting recognition and word correction. First, a webcam is

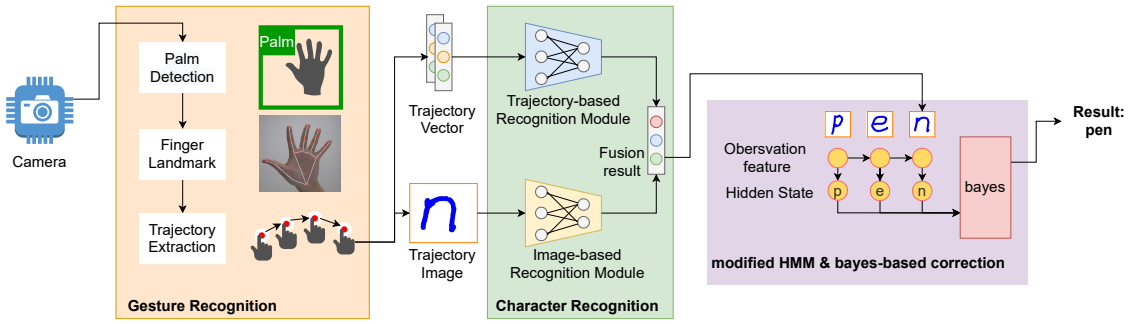


Fig. 2. System Architecture

used to capture users' hands as the input. Then we call MediaPipe API to do palm detection and finger landmarks marking. Based on the vector included angle of each finger, users' gestures can be recognized and then users can write, erase or control the system with their fingers. After acquiring users' hand-writing points coordinates, use trajectory-based recognition model to recognize the sequence of points and image-based recognition model to recognize the generated image. Then implement late fusion to merge two models together and classify what letter or number users have written. Finally our system will do correction based on common word frequency to give three possible options if users are not satisfied with the recognition result.

4.2. Gesture Recognition and Operating System

Hand Detection is realized by calling MediaPipe's Hands API, which returns fast and accurate result of finger landmarks and coordinates. Then we calculate the included angle between each finger's first knuckle vector and finger root vector to judge if a finger is straight or bent. With all fingers' included angle information gathered, we can classify which gesture users are performing as shown in Figure 3 in real time and then users can use different gestures to control the system as shown in Table 3.

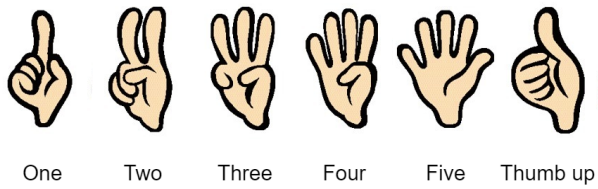


Fig. 3. Gesture Icon

As system starts, users will see themselves in the camera, once a hand is lifted up and captured by camera, it will be immediately detected and assigned with landmarks. To get more steady writing hand detection, our system will only detect one hand (whether left or right) at the same time.

Following the instruction presented on the top of the screen, users need to Gesture "4" to start writing system and four rectangles will show up. These four regions are letter regions and only the letters write inside the bounding box will be recognized finally. To distinguish valid and invalid writing, different colors are assigned to them.

Table 3. Gesture Rules.

Gesture	Rules
One	Write
Two	Erase
Three	Clear the screen
Four	Start writing
Five	End writing and Recognize
Thumb up	Save writing

Then users can use their index fingers to start writing, with the same time, their gestures will be recognized as Gesture "1". If users want to erase some wrong hand-writings, they can use index finger and middle finger together as eraser. And this gesture will be recognized as Gesture "2". If users are not satisfied with hand-writing and want to restart from start, they can use Gesture "3" to clear the screen and start writing new letters.

Once users want to save writing, they can perform Gesture "Thumb up", then all writings in the letter region will be saved and screen will be cleared for further writing. If users do not want to use all four regions to write, it is flexible to choose any combination of regions to write. Users can keep writing as wish and connect all hand-writings together as one word for recognition. Like if a user want to write word "finger", he can write "fing" first and save, then write "er" and save. All letters will be saved as a sequence, and recognized together as "finger". A combination of writing "fin" first and then "ger" later is also feasible.

If users have finished writing and want to get recognition result, they can perform Gesture "5" to end writing and start recognizing. Our recognition models are preloaded, so it will only take little time to get the result. Recognition result will be displayed to the users in green color, and three possible corrected result in indigo color will also be offered to users for

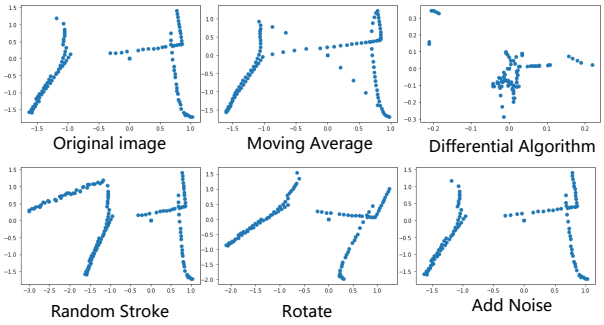


Fig. 4. trajectory data preprocessing

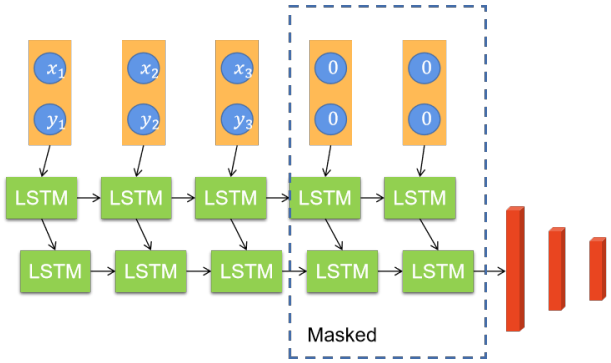


Fig. 5. stacked LSTM

choice. Users can use Gesture "Thumb up" to confirm result and not use corrected result, or use Gesture "1", "2" or "3" to choose one of the correction result. Final result will be displayed in the bottom of the screen in green color. After that, a complete process of writing and recognizing is finished. If users want to write again, they can use Gesture "4" to restart.

4.3. Trajectory-based Handwriting Recognition

The Trajectory Handwriting recognition module receives the trajectory generated by the hand detection module and gives the prediction result based on the trajectory. To achieve this, we propose a deep learning pipeline.

4.3.1. Data preprocessing

We use the trajectory data in the Char74k dataset in Section 3.1 for training. Prior to this, some necessary data transformation and data augmentation methods should be applied. We will discuss the practices and reasons of these methods in this section, and in Section 5 we will give specific experimental comparisons of these methods. Figure 4 shows the overall preprocessing results.It is worth noting that the result data in the experimental comparison shows the performance of the model on the validation set while some improvements is not obvious. However, the testing results in the real environment show that most of the preprocessing steps are indispensable. They greatly improve the anti-interference of the model.

4.3.1.1. Data Transformation

The trajectories in the Char74k dataset are written on the tablet PC, so there are discontinuities between the trajectories. (This means that the user removes the pen from the tablet). However, in a real air writing scenario, most of the user's writing trajectory is continuous. To make the training samples closer to the samples in the real application scenario, some automatically generated data points will be inserted to make the trajectory become continuous.

Specifically, we calculated the average displacement length between the entire trajectory, and divided the distance between the two discontinuous points by the average displacement length to estimate the number of points that should be inserted between the two points. The coordinates x_i, y_i of the insertion point can be calculated as follows:

$$\bar{S} = \frac{\sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{n} \tag{1}$$

$$x_i = \sum_{i=1}^{S} x_{start} + i * \frac{|x_{start} - x_{end}|}{S} + Noise \tag{2}$$

$$y_i = \sum_{i=1}^{S} y_{start} + i * \frac{|y_{start} - y_{end}|}{S} + Noise \tag{3}$$

Through experiments, we also found that models modeled on sequences using forward difference and moving average algorithms are more robust in real scenarios. Further, to enhance the generalization ability of the trajectory-based model, normalization would be applied to make the model scale-invariance.

4.3.1.2. Data Augmentation

The number of samples for each character category in the Char74k dataset is only 55, which is not enough for deep learning model training. Therefore, three data augmentation methods are designed to expand the dataset and enhance the model's anti-interference ability in the real environment. Since the characters written by the user in the air are often oblique, rotation augmentation is introduced to enhance the rotate-invariance of model. Further, to simulate the complex real scene, some random strokes are added on the starting point or ending point of the trajectory. Finally, random noise would be add on to some points of the trajectory. Through these augmentation, we enlarge our dataset to 5 times before.

4.3.2. Modeling

We have built some simple deep learning models and compared their results, including CNN model, Single-layer LSTM (Long short-term memory) model, BiLSTM (Bidirectional LSTM) model and double-layers stacked LSTM model. CNN

is often applied in signal processing tasks for its high efficiency and accuracy. In sequence classification task, we mainly use 1D (one-dimensional) convolution to extract features of the sequence. LSTM is a special kind of RNN which is designed to process long sequence data. It can alleviate the problem of gradient disappearance and gradient explosion. Since the feature transformation capability of the single-layer LSTM model may be insufficient, so we also consider multi-layers LSTM. Two multi-layers LSTM implementation are considered. One is Bidirectional LSTM inside which the sequence will be extracted from the forward and backward respectively and further be integrated. The other is double-layers stacked LSTM. The output of the previous LSTM layer will become the input of the next LSTM layer. According to the experiment results, we chose the double-layers stacked LSTM model as the trajectory-based recognition model. The model structure is shown as Figure 5 and the experimental results will be shown in Section 5.

4.4. Image-based Handwriting Recognition

The image-based handwriting recognition module obtains the trajectory generated and transferred by the hand detection module and predicts the corresponding result based on the trajectory.The complete recognition pipeline is designed as following subsections.

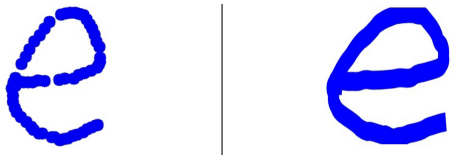
4.4.1. Data Prepossessing

After receiving trajectories composed of discrete points, we connect them into smooth curves which are the same as in training dataset. Considering the writing habit of human, it is unnatural to connect all the discrete points using straight lines since the style of daily writing is normally in curve-like way. Therefore, we decide to create natural connection for discrete trajectory points using Bézier Curve. The Bézier curve in vector graphics are generated to model smooth curves that can be scaled indefinitely based on some discrete points. In mathematics, a first-order Bezier curve is just a straight line between two given points. When there are more given points, control points are assigned to adjust the generated curves between each pair of the given points. Given the point P_0, P_1, P_2, ..., P_n, the Bézier curve is presented as follow:

$$
\begin{aligned}
B(t) &= \sum_{i=0}^{n} \begin{pmatrix} n \\ i \end{pmatrix} P_i (1-t)^{n-i} t^i \\
&= \begin{pmatrix} n \\ 0 \end{pmatrix} P_0 (1-t)^n t^0 + \begin{pmatrix} n \\ 1 \end{pmatrix} P_1 (1-t)^{n-1} t^1 + \ldots + \\
&\begin{pmatrix} n \\ n-1 \end{pmatrix} P_n - 1 (1-t)^1 t^{n-1} + \begin{pmatrix} n \\ n \end{pmatrix} P_n (1-t)^0 t^n,
\end{aligned}
$$

$$
t \in [0, 1] \tag{4}
$$

Where the curve starting at P_0 and ending at P_n is the so-called end point interpolation property. The necessary and sufficient condition for the Bézier curve to be a straight line is that the control points are collinear. The starting point (end point) of the curve is tangent to the first (last section) of the Bézier polygon. A curve can be cut into two or more sub-curves at any point, and each sub-curve is still a Bézier curve. We can use these attributes of Bézier curve to connect received trajectory points together as a smooth curve. The following Figures shows an example result.



(a) Received Discrete Points (b) Generated Smooth Curve

Fig. 6. Bézier curve Example

4.4.2. Modeling

For image-based handwriting recognition, there are many ways of extracting features from static images. The original paper has used linear classifier and OPIUM classifier, and the result is not satisfying enough for our project. After comparing other method using deep learning network model (e.g 2 conv + 1 dense), we decide to use a multi-layer CNN model to extract features. The detailed results of these methods are shown in section 5.3. There are about 70,000 images of 62 classes for training, and we found that four convolutional layers works well for this task. We design two convolutional layers, two batch normalizing layers, and one max pooling layer as one conv2d unit. There are two conv2d units in the whole image-based CNN followed by the dropout layer. Then all the extracted feature maps are flattened into single one-dimensional vector, and finally return a feature vector having 62 columns by passing through dense layers. The proposed model structure is shown below in Figure 7.

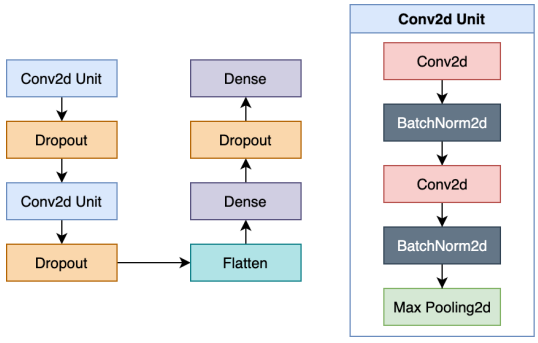


Fig. 7. image-based CNN

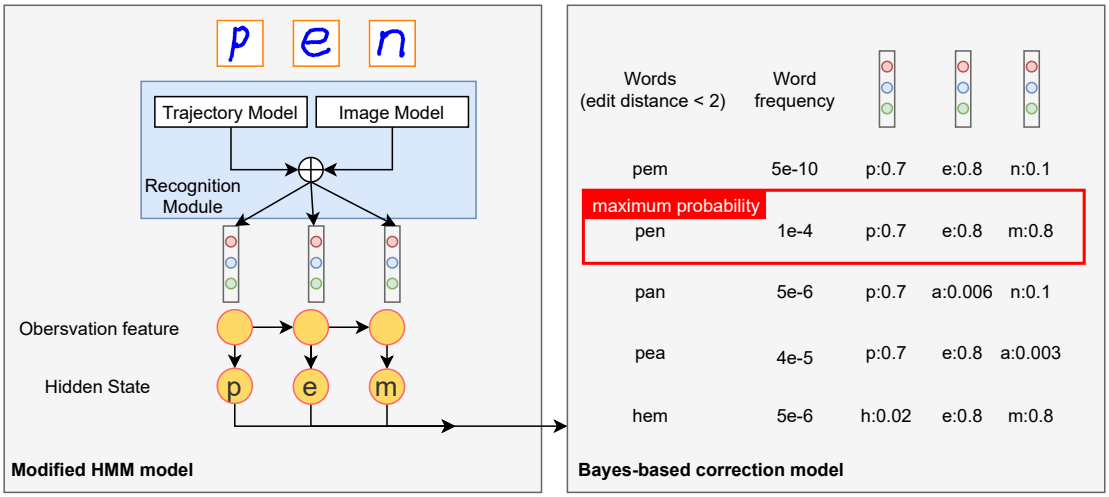


Fig. 8. Correction module

4.5. Correction Module

The entire correction module consists of two parts as shown in Figure 8. The first part is a modified HMM (Discriminant HMM), which is mainly used to correct character recognition result based on context. The second part is a Bayesian model, which is mainly for correction of common words. The correction module receives the recognition results from Trajectory-based Handwriting Recognition Module and Image-based Handwriting Recognition Module and further performs simple fusion. Specifically, we perform a weighted summation of the 62-dimension outputs of the last layers of the two model (these outputs does not activated by softmax function). This fusion result is defined as emission probability and its formal definition is as follows:

$$P_{e_i} = \frac{\lambda_1 * O_{trajectory_i} + \lambda_2 * O_{image_i}}{Z} \tag{5}$$

Where λ_1, λ_2 are the weight parameters manually set and $O_{trajectory_i}$, O_{image_i} respectively represent the $i-th$ dimension output of Trajectory-based Handwriting Recognition Model and Image-based Handwriting Recognition Model. Z is a normalization parameter to make $\sum_{i=1}^{n} P_{e_i}$

4.5.1. *Modified HMM based correction model*

The recognition module has difficulties in recognizing two situations: one is recognizing similar character pairs, such as 0 and O, 9 and g. The other is recognizing uppercase and lowercase character pairs, such as c and C, s and S. The common solution is to recognize only uppercase or lowercase characters. But capitalization is very important for some application scenarios, such as signatures. So we decided to introduce contextual information to solve this difficulty. When recognizing a character X, this X will be more likely to be o instead of O or 0 if its context is $smXoth$.

The model to achieve this goal is a modified version of HMM, which will consider the probability of the entire word. To facilitate modeling, the original HMM assumptions is changed so that the observation state (picture and trajectory) determines the hidden state (character).The modified HMM model has changed from a generative model to a discriminative model, and it is more in line with our common sense in this scenario. Given an image sequence i and trajectory sequence t, the probability that the word w was represented by these sequences can be calculated as follows:

$$\begin{aligned} P(w|i,t) &= \sum_{k=1}^{n} P(w_k|i_k,t_k) * P(w_k|w_{k-1}) \\ &= \sum_{k=1}^{n} P_k^e * P(w_k|w_{k-1}) \end{aligned} \tag{6}$$

$$P(w_k|w_{k-1}) = \frac{P(w_k, w_{k-1})}{P(w_{k-1})} \tag{7}$$

Where n represents the length of the entire sequence, and $P(w_k, w_{k-1})$ and $P(w_{k-1})$ are both empirical probabilities calculated from the browns dataset. Finally, to avoid computing the probability for all possible characters combinations, we use the Viterbi algorithm to reduce the amount of calculation with the idea of dynamic programming. The word w with the largest $P(w|i,t)$ would be the input of our next correction module.

4.5.2. *Bayes-based correction model*

Sometimes users may write words incorrectly or the model may recognize some common words incorrectly. To avoid these situations, we propose a Bayes-based correction model that considers both word frequency and the probability of

recognition results. Firstly a list of common words is constructed from Browns dataset. Then all character combinations with an edit distance less than 2 from the output word of the modified HMM model will be considered as candidate words. The words not in the common vocabulary list will be filtered. Finally, the 3 words with the highest probability among these words will be returned as the final result. The probability of a candidate word can be calculated as following:

$$\begin{aligned} P(w|i,t) &= P(w)^{\beta_1} * \sum_{k=1}^{n} P(w_k|i_k,t_k)^{\beta_2} \\ &= P(w)^{\beta_1} * \sum_{k=1}^{n} (P_k^e)^{\beta_2} \end{aligned} \tag{8}$$

Where n represents the length of the entire sequence, and $P(w)$ is empirical probabilities estimated by the occurrence probability of word w in Browns. And β_1, β_2 are the weight parameters manually set to adjust the influences of word frequency and the probability of recognition results.

4.6. Front-end Design

To make our system more straight-forward, and help users know gesture category and instruction well, we design a simple web user interface using Flask and HTML. Frames are circularly captured by camera and processed, then pushed to the web template. So we can get steady output on the localhost.

5. EXPERIMENTAL RESULTS

5.1. Experiment setting

We use three metrics to evaluate our handwriting recognition models: accuracy, macro precision, macro recall, and macro f1. The metrics could be calculated as follows:

$$accuracy = \frac{TP + TN}{|N|} \tag{9}$$

$$\begin{aligned} macro \quad precision &= \frac{1}{|C|} \sum_{c=1}^{|C|} precision_c \\ &= \frac{1}{|C|} \sum_{c=1}^{|C|} \frac{TP_c}{TP_c + FP_c} \end{aligned} \tag{10}$$

$$\begin{aligned} macro \quad recall &= \frac{1}{|C|} \sum_{c=1}^{|C|} recall_c \\ &= \frac{1}{|C|} \sum_{c=1}^{|C|} \frac{TP_c}{TP_c + TN_c} \end{aligned} \tag{11}$$

$$macro \quad f1 = \frac{1}{|C|} \sum_{c=1}^{|C|} \frac{2 * precision_c * recall_c}{precision_c + recall_c} \tag{12}$$

Where $|N|$ represents the number of total samples and $|C|$ represents the number of categories and TP_c, FP_c, TN_c represents the number of true positive samples, false positive samples, true negative samples of category c respectively.

5.2. Trajectory based

The experiment was conducted under the premise that the network parameters and data amount were relatively consistent. Totally there are two types of comparative experiments were conducted : data preprocessing methods and network structures comparison. The experiment results are shown as Table 4. In the comparative experiment of data preprocessing methods, it's shown that forward difference and moving average algorithm has greatly improved the performance of the model. This is because the forward difference explicitly extract the information of directional change along the time dimension and the moving average algorithm smooths some trembling strokes to reduce noise. Rotate and Add noise also improve the performance of the model, but it is not as obvious as the former. We believe that these two operations avoid model overfitting by increasing the diversity of training data. Although add random strokes seems to hurt the performance of the model, this operation can greatly improve the recognition rate of the model in a real scene. This is because users often accidentally add some abnormal strokes when writing in the air.

In the comparative experiment of network structure, LSTM performed better than CNN. This is mainly because the network structure of LSTM is more suitable for processing sequence data than CNN. The performance of multi-layer LSTM is better than single-layer LSTM, because multi-layer LSTM has stronger nonlinear transformation ability. We also compared the performance of the BiLSTM model and double-layers stacked LSTM. Double-layers stacked LSTM is significantly better than the BiLSTM model. In this regard, we have such an assumption: for the trajectory, there is no obvious difference between forward learning and reverse learning because they are both trying to learn a certain sequence. In BiLSTM, the features of each time step are actually transformed by a different nonlinear layer in two directions. Therefore, a feature is actually only transformed once. In stacked LSTM, the features of each time step will undergo two-layer transformation, so the model expression ability will be stronger.

5.3. Image based

The experiment was performed on the By Class dataset (62 classes comprising 10 digit classes, 26 lowercase letter classes, and 26 uppercase letter classes) using proposed image-based CNN model, and the dataset contains vastly more digit samples than letter samples. We conduct the experiments on own CNN model and compare our result with

Table 4. Trajectory-based model comparison

Approach	Data-preprocess methods				Metrics		
	Rotate	Random Strokes	Add Noise	Forward difference	Precision	Recall	F1
Stacked LSTM					0.725	0.713	0.708
Stacked LSTM	√				0.731	0.728	0.721
Stacked LSTM		√			0.718	0.715	0.697
Stacked LSTM			√		0.726	0.715	0.710
Stacked LSTM				√	0.740	0.738	0.728
Stacked LSTM	√	√	√	√	**0.747**	**0.743**	**0.731**
CNN	√	√	√	√	0.687	0.695	0.673
LSTM	√	√	√	√	0.722	0.721	0.718
BiLSTM	√	√	√	√	0.730	0.727	0.720

methods proposed by other researchers carrying recognition task on the same EMNIST dataset. The original methods used in EMNIST published paper [11] are linear classifier and OPIUM classifer. They performed the handwriting recognition task and the accuracy is 51.80% and 69.70% respectively. These two methods are generate ananalytical solutions and have relatively low accuracy due to the limitation of classical machine learning techniques. Compared to linear classifier and OPIUM classifer, methods invlving CNNs have better performance with higher accuracy above 85%. The CNN with two convolutional layers and one dense layer reaches the accuracy of 87.1%, and it performs worse than our model due to the shallow network design than ours. The committee of 7 CNNs has the highest performance among all the models listed in Table 5. Since it is an ensemble model and uses multiple CNNs to make the final decision, it makes the model more robust while increasing the computation complexity. Though the accuracy of our model is a litter lower than committee of 7 CNNs, the computation complexity and time are much less than committee of 7 CNNs with an relatively high accuracy of 87.5%, precision of 87.6%, recall of 88.0%, and f1 score of 0.868.

Table 5. Image-based model comparision

Approach	Metrics			
	Accuracy	Precision	Recall	F1
Linear classifier	0.518	–	–	–
OPIUM	0.697	–	–	–
CNN(2 conv + 1 dense)	0.871	–	–	–
Committee of 7 CNNs	0.881	–	–	–
Our model	**0.875**	**0.876**	**0.880**	**0.868**

6. CONCLUSIONS AND FUTURE WORK

We build a functional air writing system, which can be operated totally contact-less. Users can enjoy writing on screen using their fingers and perform different gestures to control the system. Compared with other works, our system is more functional and intuitive with clear guidance. And it is not only interesting, but also useful because we are able to recognize hand-writing. To get accurate recognition result, we creatively combine trajectory-based method and image-based method together, which is rarely implemented before. And our system is user-friendly, because correction module will provide 3 potential prediction of writing for users to choose their desired result.

For future work, we will try to increase the accuracy of recognition models and correction models. And then add more functions to the writing system to make it more interesting. We will also try to reduce loading time and computational resource requirement so that our system can be more widely deployed on microcomputers.

7. CONTRIBUTIONS

Xu Xuanbo's contributions are mainly in Trajectory-based Recognition Module and Correction Module. For Trajectory-based Recognition Module, he tried several models including LSTM, bi-LSTM, stacked-LSTM and 1D-CNN to solved the problems. For Correction Module, he proposed a modified HMM model and a Bayse-based correction model to do the correction.

Ying Zhuozuo contributed to build writing system and user interface. He called MediaPipe API to do hand detection on frames captured by camera. Based on generated finger landmarks, he used finger vector included angles to recognize users' gestures. Then he wrote logical rules of operating system using gestures and generated trajectory list as recognition models' input. He also used Flask and HTML to design front-end user interface.

Zhong Yuting mainly contributed to Image-based Recognition Module. After comparing advantages and disadvantages of mainstream techniques for handwritten digit and letter recognition, she made a final decision on using 2D-CNN model after the input (discrete points) converted into continuous curve by applying Bezier curve to solve the problem.

8. REFERENCES

[1] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[2] Fangfang Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann, "Mediapipe hands: On-device real-time hand tracking," *ArXiv*, vol. abs/2006.10214, 2020.

[3] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[4] S. Mohammadi and R. Maleki, "Air-writing recognition system for persian numbers with a novel classifier," *The Visual Computer*, vol. 36, no. 1, 2020.

[5] S. Poularakis and I. Katsavounidis, "Low-complexity hand gesture recognition system for continuous streams of digits and letters," *IEEE Trans Cybern*, vol. 46, no. 9, pp. 2094–2108, 2016.

[6] "Person authentication by air-writing using 3d sensor and time order stroke context," *National Central University, Taoyuan City 32001, Taiwan;National Taiwan Ocean University, Keelung City 202, Taiwan;National Central University, Taoyuan City 32001, Taiwan;National Taiwan Ocean University, Keelung City 202, Taiwan;National Taiwan Ocean Univ*, 2018.

[7] C. Amma, M. Georgi, and T. Schultz, "Airwriting: A wearable handwriting recognition system," *Personal and Ubiquitous Computing*, vol. 18, no. 1, 2014.

[8] M. S. Alam, K. C. Kwon, M. A. Alam, M. Y. Abbass, S. M. Imtiaz, and N. Kim, "Trajectory-based air-writing recognition using deep neural network and depth sensor," *Sensors (Basel, Switzerland)*, vol. 20, no. 2, 2020.

[9] P. Kumar, R. Saini, P. P. Roy, and D. P. Dogra, "Study of text segmentation and recognition using leap motion sensor," *IEEE Sensors Journal*, vol. PP, no. 99, pp. 1–1, 2016.

[10] "Online and adaptive pseudoinverse solutions for elm weights," *Neurocomputing*, vol. 149, pp. 233–238, 2015, Advances in neural networks Advances in Extreme Learning Machines.

[11] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik, "Emnist: an extension of mnist to handwritten letters," 02 2017.

[12] Premanand Ghadekar, Shubham Ingole, and Dhruv Sonone, "Handwritten digit and letter recognition using hybrid dwt-dct with knn and svm classifier," 08 2018, pp. 1–6.

[13] Yao Peng and Hujun Yin, "Markov random field based convolutional neural networks for image classification," 10 2017, pp. 387–396.

[14] Thi Thi Zin, Shin Thant, Moe Pwint, and Tsugunobu Ogino, "Handwritten character recognition on android for basic education using convolutional neural network," *Electronics*, vol. 10, pp. 904, 04 2021.

[15] Alejandro Baldominos, Yago Sáez, and Pedro Isasi, "Hybridizing evolutionary computation and deep neural networks: An approach to handwriting recognition using committees and transfer learning," *Complexity*, vol. 2019, pp. 2952304, 03 2019.

[16] "Air-drawing," https://https://github.com/loicmagne/air-drawing, 2021.

[17] Ted Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *VISAPP 2009 - Proceedings of the Fourth International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, February 5-8, 2009 - Volume 2*, 2009.

[18] "Nist," https://www.nist.gov/srd/nist-special-database-19, Accessed: 2021-10-25.