Lightweight CNN Architecture for IoT: Enhancing Character Recognition in Multiple Fonts

Chung-Hsing Chen[1,2], Ko-Wei Huang[1]*

[1] Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Taiwan
[2] Plustek Inc., Kaohsiung City, Taiwan
I110154101@nkust.edu.tw, elone.huang@nkust.edu.tw

Abstract

Existing approaches to English character recognition generally ignore font differences, and those based on deep learning are often trained on only one font owing to computational constraints. To address this problem, we propose the Very Lightweight Network (VLNet), a lightweight convolutional neural network. First, we decompose the characters of a given font into their constituent strokes. Subsequently, we pass the stroke information as input to a simple convolutional neural network. Stroke-based feature extraction reduces the requirement for graphics data and training input size. The network is small and efficient; thus, it is suitable for edge computing and Internet of Things applications. In the experimental comparison with the standard character-recognition systems LeNet, AlexNet and MobileNet V3, VLNet demonstrated superior accuracy for known and unknown fonts and a reasonable prediction time per character. The results of this study were also implemented in Plustek Inc.'s Q30 network scanner, which enabled direct document content recognition and transfer to various cloud services.

Keywords: Feature extraction, CNN, Internet of things, Deep learning, Edge computing

1 Introduction

Recently, the field of automatic document processing has seen significant advancements. The demand for data extraction from documents has increased due to their digitalization. Data extraction and recognition tasks for paper-based documents pose significant challenges. Although these documents can easily be converted into digital formats using a scanner and computer via traditional scanning, after this procedure, the documents go through a storage phase that requires manual classification and creation of media files from images. With the increasing prevalence of artificial intelligence (AI), there has been a rise in the demand for internet-connected network scanners that have taken on the role of data extraction for cloud storage. Data sent to the cloud are not limited to image files but consist of media files containing document content, which can be used for automatic classification, archiving, and transfer of diverse types of document to cloud platforms. For example, accounting, education, and government documents can be loaded into accounting, education, and government clouds, respectively. For this, network scanners need to be upgraded to perform AI edge computing for document pre-recognition and deliver the necessary document content to cloud storage. Consequently, network scanners have evolved into Internet-of-Things (IoT) devices. To achieve this functionality, we propose a neural network that performs edge computing and connects to cloud services, constituting a complete IoT device. The network can run smoothly on the Raspberry Pi 4B platform.

The automatic recognition of English text is a subject of extensive research. The process begins with the document layout analysis and continues with preprocessing, text segmentation, optical character recognition (OCR), and postprocessing. However, the English text is printed in different fonts, and an effective text-recognition system must handle this diversity, particularly during the OCR stage. Because a neural network cannot effectively generate predictions, we developed a model comprising feature extraction methods and a neural network which can accurately learn character strokes and features, effectively predict unknown fonts, and readily obtain results without requiring a graphics processing unit (GPU).

Recent studies have employed deep learning to improve recognition accuracy rates. [1-3] However, this method cannot be suitably implemented in general Internet-of-Things (IoT) settings and was therefore not used in this study. Furthermore, we utilized the more representative physical features of the recognition targets and did not adopt automatic feature extraction in deep learning.

Although many character recognition methods have been developed [4-12], these methods use the same fonts for training and verification. However, fonts affect the prediction results of deep learning. If a font is absent from the training data, its recognizability may be low. This is similar to the classic problem of binarization: most methods are employed in all situations because they cannot separate the foreground from the background in all images. The first step in learning to write English letters involves mimicking the printed writing in books. The 52 capital and lowercase characters of the English alphabet are composed of strokes [13] which have an associated direction relative to the center of the

*Corresponding Author: Ko-Wei Huang; E-mail: elone.huang@nkust.edu.tw
DOI: https://doi.org/10.70003/160792642024122507004

character. In principle, this direction may be arbitrary. If only horizontal or vertical strokes are considered, the directions are limited to 0, 90, 180, and 270° [14]. Many fonts also include 125° strokes from the upper left to the lower right. In this study, we used strokes to characterize letters in different fonts.

Traditional AI networks are trained using preprocessed images to produce a desired model. However, different fonts that do not have clear features may negatively impact learning efficiency, affecting prediction accuracy. Because a neural network cannot effectively predict which fonts it will encounter, we developed a set of feature extraction methods and a neural network that can accurately learn character strokes and features. Our Very Lightweight Network (VLNet) employs two-layer one-dimensional convolutions and a three-layer fully-connected layer and accepts our artificially induced stroke features as input. Thus, VLNet effectively predicts unknown fonts and can deliver immediate results without needing a graphics processing unit (GPU). The primary distinction between VLNet and State of the Art networks in the field of artificial intelligence lies in their architectural approach. Contemporary network designs prioritize parameter simplification within each layer to reduce computational overhead. In contrast, VLNet is characterized by its inherent simplicity, comprising only two one-dimensional convolutional layers, resulting in fewer parameters and layers. Its uniqueness stems from its utilization of image feature data for training, deviating from the prevalent paradigm of employing networks with numerous layers and parameters. VLNet's innovative training methodology harnesses image features to replace the need for multi-layer convolutions, achieving comparable precision and producing equivalent outcomes. This novel approach demonstrates the evolution of network design within the AI domain.

We conducted comprehensive experiments to assess VLNet's ability to recognize both the same letter in different fonts and different letters in the same font. Additionally, we compared VLNet with existing networks such as LeNet [15], AlexNet [16], and MobileNet V3 [17] utilizing the same training data for a fair evaluation.

We considered physical characteristics that are more indicative of the recognition targets in the feature extraction process and did not use the automated feature extraction of deep learning. As a result, we could learn with less data and produce more precise features. The quantity of tensors employed also has an impact on computation. LeNet and AlexNet both employ 784 and 154,587 tensors, compared to 135 used by VLNet. In the IoT environment, the use of features is vital. This study provides experimental evidence that character recognition can be integrated with the IoT. In our innovative method, the architecture rather than the computer capacity determines how well a system can recognize objects.

Existing approaches to English character recognition generally ignore font differences, and those based on deep learning are often trained on few fonts owing to computing constraints. To address this problem, we first decompose the characters of a given font into their constituent strokes (directed line segments).

The performance of VLNet on unknowed fonts demonstrates an impressive accuracy of 97.81%, surpassing that of LeNet (96.67%), AlexNet (91.15%), and MobileNet V3 (95.77%). Additionally, for knowed fonts, VLNet achieves an even higher accuracy of 99.1%, outperforming LeNet (98.59%), AlexNet (95.48%), and MobileNet V3 (98.31%). The predicSubsequently, we use the stroke information as the input to a simple convolutional neural network, namely the Very Lightweight Network (VLNet). Stroke-based feature extraction reduces the requirement for graphics data and thus the training input size. The network is suitable for edge computing and use in the Internet of Things as it is small and efficient. An experimental comparison with the standard character-recognition systems LeNet and AlexNet demonstrate that ttion time per character is between 4.44 and 4.61 ms when the system is implemented on Raspberry Pi 4B.

The results of this study were also implemented in Plustek Inc.'s Q30 network scanner. The Q30 scanner is capable of performing document content recognition and uploading different types of documents to various cloud services directly from the machine.

The remainder of this paper is organized as follows. Section 2 presents a review of relevant literature. The proposed model is outlined in Section 3. The experimental results and performance are evaluated in Section 4. Finally, conclusions and the scope for future study are presented in Section 5.

2 Related Works

In this section, we review prior studies on the process of converting a paper document into digital text through image classification using OCR and convolutional neural networks (CNNs). In general, OCR operates through image acquisition, whereby an image is transformed into a digital file and preprocessed [18]. The subsequent algorithm is impacted by the preprocessing quality. When dealing with split characters during feature extraction, major issues may arise if the preprocessing does not effectively separate the character foreground from the background. File splitting may lead to errors in target recognition; for example, sticky or broken text may cause recognition failure. Preprocessing is followed by image segmentation [19], wherein each character to be recognized is segmented from the original image. As mentioned previously, low-quality foreground/background separation may lead to characters being missed or split characters being merged incorrectly. Thereafter, feature extraction [20-21] is applied, followed by classification [22]. The UCI capitalized English alphabet database is a machine learning database available for research [23]. It provides the features of 16 characters for training. This database is not used to train artificial intelligence (AI) networks as this would require the network to learn character properties by itself; however, this easily affects induction and convergence owing to variability among fonts. In other words, the use of character features for training can eliminate the problems associated with mutually different fonts.

Existing deep-learning classification methods are based on prior feature extraction. In CNNs, feature extraction is contained within the convolutional layers, whereas classification occurs in the fully-connected layers.

CNNs have been used for tasks such as word recognition [24], handwriting recognition [25], intrusion detection [26], sign-language recognition [27], gesture recognition [28], automatic modulation classification [29], and parasitic egg detection [30]. LeNet-5 was the first CNN employed for handwritten digits recognition. It uses two layers of two-dimensional convolutions and two layers of maxpooling, followed by a three-layer fully-connected layers. AlexNet was used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) on September 20, 2012. It comprises eight layers, of which the first five are convolutional layers (some of them followed by max-pooling layers), and the final three are fully connected layers. Subsequently, VGGNet [31], GoogLeNet [32], ResNet [33], and other variants have been proposed, expanding the width and depth of network models and setting new ILSVRC accuracy records.

However, these increasingly deep network architectures require large amounts of data. Although feature extraction is performed through the convolutional layer, the neural network obtains features by calculating image pixels and then using supervised learning with a large amount of training data to obtain a suitable model for classification. The greatest advantage of this method is that it does not require manual feature extraction: data of the same type only require appropriate labeling for use in deep learning. However, the disadvantage is that, owing to a higher accuracy and greater number of categories (i.e., greater network depth), the computation must be handled by a GPU. In contrast, we employed artificial feature extraction and a shallow neural network to achieve text classification. Each character is represented by its geometrical and statistical features and subsequently used to train the neural network.

MobileNet [34] was proposed by Andrew G. Howard from Google Inc. in 2017. It is a lightweight CNN image classification model designed for mobile and embedded devices. The key technique used in MobileNet is depthwise separable convolution. In this approach, depthwise convolution applies separate convolutions to each channel to reduce computational complexity, whereas pointwise convolution learns relationships between different channels by transforming the traditional convolutional multiplications into additions. This reduces the number of convolution operations and significantly decreases the number of network parameters, thereby improving computational efficiency.

In 2018, Howard introduced MobileNet V2 [35]. The motivation behind this version was the presence of many empty convolutional kernels in depthwise separable convolution. The application of a rectified linear unit (ReLU) in low-dimensional space was found to cause information loss, while it did not have the same effect in high-dimensional space. To address this issue, MobileNet V2 employs linear bottlenecks to increase the input dimensionality and replace ReLU before the operation. Thus, the characteristics of MobileNet V1 were retained while resolving the problem associated with ReLU in low-dimensional space.

MobileNet V3 further improved upon the previous versions by incorporating a neural architecture search (NAS) and the Squeeze and Excitation (SENet) framework. The SENet module employs global average pooling (GAP) to calculate importance weights for each feature map, thereby strengthening the influence of important feature maps while reducing the impact of less significant ones. The upgrade in MobileNet V3 was implemented to enhance accuracy without compromising image processing time.

In summary, the MobileNet series has successfully achieved efficient image classification on lightweight devices through techniques such as depthwise separable convolution. The improvements introduced in MobileNet V2 and V3 have enhanced the models' accuracy and performance while maintaining their efficiency.

3 Proposed Method

Data collection and preprocessing are crucial in the field of document processing because adequate typographical training datasets do not exist. As with other open-source OCRs, our training data consist of images produced by the program. These images tend to be excessively clean and tidy and may lack factors that affect recognizability. Therefore, we placed significant emphasis on data collection and preprocessing. In addition, we used stroke features to enhance the recognizability of different fonts. Because strokes have different directions, we specifically considered vector features in this study. The vectors are oriented line segments that are directed outward from the center of gravity of each character. Strokes may be long or short. Moreover, we added edge features for horizontal and portrait strokes because a very long lateral or lengthwise stroke extends to the edge of the bounding box of the character. Thus, the long stroke features can be accentuated, and the features of the lateral and lengthwise strokes become more comprehensive. Each unique feature can be induced more accurately irrespective of font variability because the character images are not used for deep-learning training.

3.1 Data Collection Phase

In this study, we employed twenty distinct fonts, as enumerated in Table 1, sourced from the Microsoft Windows 10 operating system. To obtain the requisite dataset, text was printed in these fonts and subsequently scanned from printed pages. Throughout the scanning process, deliberate angle rotations were introduced to augment the diversity of letter shapes.

Initially, the selected fonts were printed on A4-sized paper using Microsoft Word, with five samples generated for each font. Subsequently, each of these five samples underwent ten scans. Following scanning, preprocessing steps were executed to isolate individual character images. The resulting images were meticulously categorized and sequentially encoded to represent the fifty-two English letters. Uppercase letters A-Z were encoded from class 0 to 25, while lowercase letters a-z were assigned class numbers from 26 to 51. This intricate process formed the foundation for both the training and testing datasets in this study.

Table 1. Font list of the training set

No	Font type
0	Arial Unicode MS
1	Malgun Gothic Semilight
2	MS Gothic
3	MS Mincho
4	MS PGothic
5	SimSun
6	Arial Narrow
7	Bahnschrift Condensed
8	Baskerville Old Face
9	Microsoft YaHei
10	Bookman Old Style
11	Candara Light
12	Comic Sans MS
13	COPPERPLATE GOTHIC BOLD
14	Courier New
15	Maiandra GD
16	Rockwell
17	Verdana
18	Mongolian Baiti
19	Leelawadee UI Semilight

3.2 Data Preprocessing Phase

Because each sample encompassed 20 fonts of English characters, each character corresponded to 20 images with a resolution of 300 DPI. The average variant method was used to calculate the binarization threshold. The average jump method was used to calculate the threshold, and the prediction equation was defined as follows:

HTGS(k) is a histogram statistical map with k ranging from 0 to 255, and it measures the average value of each pixel p. Gray(p) measures the gray value, which was substituted with the RGB three-channel average value in the present study. LastGray(p) represents the gray value of the previous pixel, and Mean is the gray value of the overall image. HTGS(k) can be calculated as

$$\begin{aligned} HTGS(k) &= \left|\{p|k \right. \\ &= \left\lfloor \frac{Gray(p)+LastGray(p)}{2} \right\rfloor, \\ &50 < Gray(p) < Mean, \\ &\left|Gray(p)-LastGray(p)\right| > 40\} \end{aligned} \quad (1)$$

By applying statistics, we obtain kmax as the global binarization threshold. Following binarization, the character is labeled and cut by the connected-component labeling method [36].

As illustrated in Figure 1, each uppercase character has a single connected component and does not require the merging of separate parts; however, certain lowercase symbols (i and j) have two connected components and therefore necessitate the merging of rectangles that contain each component.

The merge principle is depicted in Figure 2. If yt2 (yb2) is the top (bottom) point of the small rectangle, and yt1 (yb1) is the top (bottom) point of the large rectangle, the top (bottom) point of the merged rectangle, yt' (yb'), is given by yt' = yt2 (yb' = yb1). Similarly, if xl2 (xr2) is the leftmost (rightmost) point of the small rectangle, xl1 (xr1) is the leftmost (rightmost) point of the large rectangle, and xl' (xr') is the leftmost (rightmost) point of the merged rectangle, xl' = min (xl1, xl2) and xr' = max (xr1, xr2).

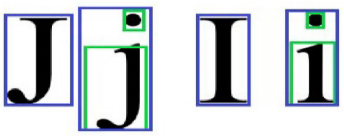


Figure 1. Character cutting and labelling

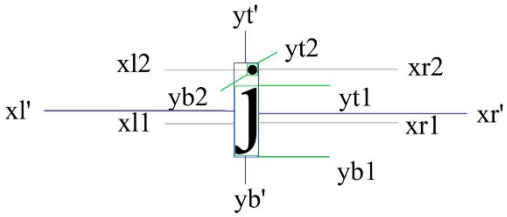


Figure 2. Terminology for boundary boxes edges of connected components before and after merging

3.3 Vector Features

As illustrated in Figure 3, vectors are used to represent the 12 directions radiating from the center of gravity. Given a character of width w and height h, its center of gravity in Cartesian coordinates is (w/2, h/2). (If the coordinates are not integers, we round them down.) Starting at this center of gravity, we plotted a straight line every 30° from the horizontal. Three points on this straight line represent eigenvalues of the vector. Thus, the 12 lines correspond to 36 vector features. The algorithm for the stroke direction feature method and selection of sampling point coordinates is expressed in Eqs. 2–9 [37]:

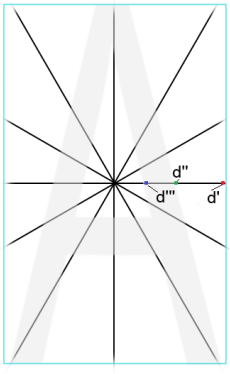


Figure 3. Vector-feature association schematic

$$\text{Short Edge} = \begin{cases} Width, \ if \ \ Height > Width \\ Height, \ if \ \ Width > Height \end{cases} \quad (2)$$

$$nCenterX = nWidth / 2 \quad (3)$$

$$nCenterY = \frac{nHeight}{2} \quad (4)$$

$$C = \left(nCenterX, nCenterY\right) \quad (5)$$

$$\text{nRadius} = \text{shortEdge} / 2 \quad (6)$$

$$\text{nRadiusHalf} = \text{nRadius} / 2 \quad (7)$$

$$\text{nRadiusQuarter} = \text{nRadiusHalf} / 2 \quad (8)$$

$$d_1 = (nCenterX + nRadius \times \cos(30^\circ \times i \times \frac{\pi}{180}), nCenterY + nRadius \times \sin(30^\circ \times i \times \frac{\pi}{180}))),$$

$$d_2 = (nCenterX + nRadiusHalf \times \cos(30^\circ \times i \times \frac{\pi}{180}), nCenterY + nRadiusHalf \times \sin(30^\circ \times i \times \frac{\pi}{180}))),$$

$$d_3 = (nCenterX + nRadius\text{Quarter} \times \cos(30^\circ \times i \times \frac{\pi}{180}), nCenterY + nRadius\text{Quarter} \times \sin(30^\circ \times i \times \frac{\pi}{180}))),$$

$$where\ i \in R,\ 0 \le i \le 11 \quad (9)$$

3.4 Edge Feature

A 360° range of vector features can be used to obtain stroke features in all directions. However, because the stroke length cannot be represented by vector features, we added edge features to reflect this information.

As illustrated in Figure 4, we can select five points on the top and bottom graph edges that are separated by quarters of the width and six points on the left and right graph edges that are separated by fifths of the height. (For most characters, the height is greater than the width.) No corner points are counted twice, and there are 18 points for the edge features. With the addition of the 36 vector-feature points, we obtain 54 feature points.

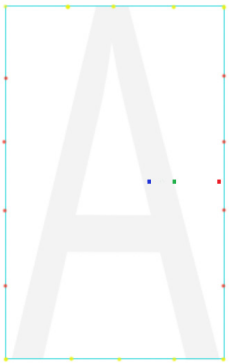


Figure 4. Schematic of edge features

3.5 Density Feature

Density is a stroke feature that indicates the manner in which a word is written. The character image is projected onto a grid, and the ratio of strokes to this grid is the average density. We used a 9 × 9 grid in this study. Thus, the width and height of each grid cell were W/9 and H/9, respectively.

The positional feature of the stroke is the set of intersections between the stroke and the 81 cells. As illustrated in Figure 5, the density D is the ratio of the number of black pixels in each cell to the area of the grid. BlackPixel/(W*H). If a stroke completely covers a cell, its density is 1; if the stroke does not pass through the cell at all, its density is 0. Thus, we used density to represent the positional features of strokes in this study.

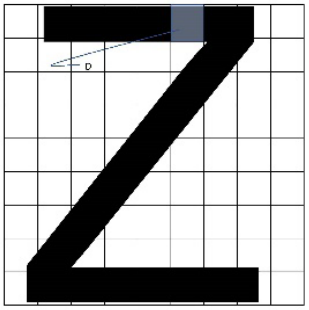


Figure 5. Positional features of a character

3.6 The Proposed VLNET Model

We designed the VLNet for edge computing. Because the stroke features are extracted as described previously, the 135 vector-feature, edge-feature, and density-feature inputs do not have two-dimensional graphics features, and the output has only 52 categories (the uppercase and lowercase letters of the alphabet).

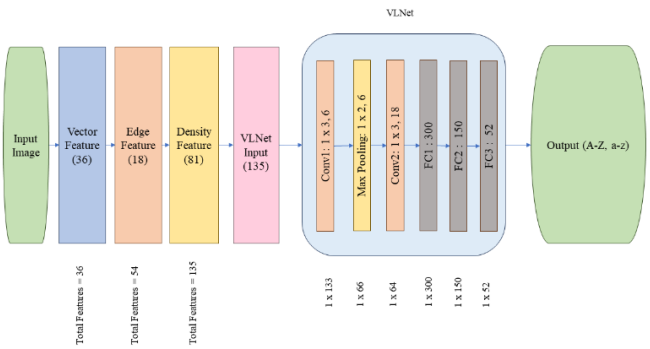


Figure 6. VLNet workflow diagram

In accordance with the graphical representation in Figure 6, the following workflow is executed:

1. **Feature Extraction:** A total of 36 vector features, 18 edge features, and 81 density features are extracted from the input image.
2. **Feature Concatenation**: All extracted features are concatenated into a 1D array of size 1 x 135, serving as the input to subsequent layers.
3. **Convolutional Layers 1:** A one-dimensional convolution operation is applied using a 1 x 3 kernel, resulting in the generation of six 1 x 133 feature maps.

4. **Max-Pooling:** Max-pooling is performed with a 1 x 2 kernel, producing six 1 x 66 feature maps.

5. **Convolutional Layers 2:** Another one-dimensional convolutional layer, utilizing a 1 x 3 kernel, refines the six 1 x 66 feature maps, resulting in 18 feature maps.

6. **Fully Connected Layers:** The 18 resulting 1 x 64 feature maps are fed into a fully connected layer, which reduces the dimensionality from 18 x 1 x 64 = 1152 features to 300 features, and subsequently to 150 features.

7. **Classification:** The final fully connected layer classifies the 150 features into one of 52 categories, producing 52 probability results."

The maximum probability according to softmax is generated as the output.

Table 2 presents a comparison between the neural network architecture proposed in this study and the comparison group architectures. The output is in k categories, which in this study represents the 52 uppercase and lowercase English characters.

Table 2. Neural network architectures comparison

	VLNet	LeNet	AlexNet	MobileNet V3 Small
Input	135 x 1	32 x 32 x 1	224 x 224 x 3	224 x 224 x 3
L1	1 x 3 Conv1d, 6	5x5 Conv2d, 6	11x11 Conv2d, 96	3 x 3 Conv 2d, 16
L2	Pool	Pool	5x5 Conv2d, 256	3 x 3 Bneck, 16
L3	1 x 3 Conv1d, 18	5x5 Conv2d, 16	Pool	3 x 3 Bneck, 24
L4	FC 400	Pool	3x3 Conv2d, 384	3 x 3 Bneck, 24
L5	FC 300	FC 400	Pool	5 x 5 Bneck, 40
L6	FC 150	FC 120	3x3 Conv2d, 384	5 x 5 Bneck, 40
L7		FC 84	3x3 Conv2d, 256	5 x 5 Bneck, 40
L8			Pool	5 x 5 Bneck, 48
L9			FC 4096	5 x 5 Bneck, 48
L10			FC 4096	5 x 5 Bneck, 96
L11			FC 1000	5 x 5 Bneck, 96
L12				5 x 5 Bneck, 96
L13				1 x 1 Conv, 576
L14				GAP
L15				1 x 1 Conv, 1280
Output	Softmax, k	Softmax, k	Softmax, k	1 x 1 Conv, k

4 Results and Discussion

In this study, 20 fonts were used for implementation (Figure 7).

Figure 7. Letter A in 20 English fonts

The training set spanned five samples of each font for 100 samples. There are 52 characters in the alphabet (upper and lower case); hence, 5200 characters were considered. Samples were not repeated between the training and test sets. Each test set comprised 260 characters, with five samples per character for a single font. The experimental objective was to evaluate the ability of the proposed model to recognize different fonts. Accordingly, each test set used a font that was absent from the corresponding training set; i.e., each model was trained with 19 fonts and tested with the remaining font. The result of each experiment was the average of the test results of 20 models.

The experimental results are presented in Table 3. We used two different calculation methods: case sensitive and case insensitive. In some fonts, the uppercase and lowercase letters were almost identical for certain characters; in all fonts, C, O, S, V, W, X, and Z were nearly identical in both cases, whereas P, U, and Y were similar in both cases. VLNet exhibited a fairly high error rate for characters with similar uppercase and lowercase forms, with a 6.41% difference between the case-insensitive and case-sensitive accuracies.

Table 3. Experiment results for unknown fonts

Model	Test target	Case-sensitive errors	Case-insensitive errors	Case-sensitive accuracy	Case-insensitive accuracy
VLNet	5200	425	110	91.40%	97.81%
AlexNet	5200	786	469	84.88%	91.15%
LeNet	5200	623	121	88.02%	97.67%
MobileNet V3	5200	336	220	93.54%	95.77%

Regardless of the image used for training, LeNet, AlexNet, and MobileNet V3 could not distinguish between uppercase and lowercase characters. These results suggest that differentiating between uppercase and lowercase characters solely based on the same strokes is impossible. Therefore, the proposed method has no disadvantage relative to other AI recognition methods. The proposed method obtained the second best results in the final test for the case-sensitive case, with 198 fewer errors (3.38%) than LeNet, 361 fewer errors (also 6.25%) than AlexNet, and 89 more errors (2.14%) than MobileNet V3. For the case-insensitive case, VLNet yielded the best results among all architectures, with 11 fewer errors (0.14%) than LeNet, 359 fewer errors (6.66%) than AlexNet, and 110 fewer errors (2.04%) than

MobileNet V3. We also selected five characters in each font from the test set to demonstrate the ability of the network to recognize known fonts better than the comparison networks. Because each training set represented 19 fonts, each test set had 52 × 19 × 5 = 4940 characters for validation. Each test set also had different samples from those in its corresponding training set; only the fonts were the same.

Table 4. Experimental results for known font

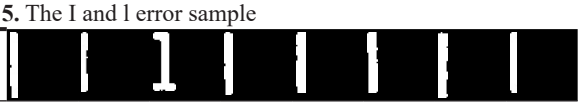
Model	Test target	Case-sensitive errors	Case-insensitive errors	Case-sensitive accuracy	Case-insensitive accuracy
VLNet	98800	1650	835	98.24%	99.11%
AlexNet	98800	6454	4470	93.47%	95.48%
LeNet	98800	7252	1392	92.66%	98.59%
MobileNet V3	98800	2327	1670	97.64	98.31%

The experimental results are listed in Table 4. We calculated separate accuracy rates for the uppercase and lowercase sets. The test results for same-font recognizability indicate that VLNet performs better than the comparison group. For the case-sensitive case, it exhibited 5602 fewer errors (5.58%) than LeNet, 4804 fewer errors (4.77%) than AlexNet, and 677 fewer errors (0.60%) than MobileNet V3. For the case-insensitive case, the proposed architecture yielded 557 fewer errors (0.52%) than LeNet, 3635 fewer errors (3.63%) than AlexNet, and 835 fewer errors (0.80%) than MobileNet V3.

In summary, the four neural networks were not as effective in recognizing unknown fonts as known fonts; however, VLNet exhibited overall superior performance for both known and unknown fonts.

Moreover, when analyzing the experimental results, we identified another special case wherein it was not easy to distinguish between uppercase and lowercase characters, particularly, uppercase I and lowercase l, which are difficult to distinguish even with the human eye in some fonts. We have provided a detailed error sample in Table 5 to illustrate instances of such challenges.

Table 5. The I and l error sample

Image								
Ans	l	l	l	l	I	I	I	I
Predict	I	I	I	I	l	l	l	l

In these specific fonts, humans can correctly distinguish the two characters through context and prior knowledge. However, as this study did not include sequence-to-sequence training, VLNet could not distinguish these characters in certain fonts. The results obtained by VLNet for unknown fonts when ignoring mismatches between I and l are presented in Table 6. In the case-sensitive experiment, accuracy reached 92.40%, which was an increase of 1.00%; in the case-insensitive experiment, accuracy reached 98.46%, which was an increase of 0.65%.

Table 6. Experimental result for unknown fonts ignoring I and 1

Model	Test samples	Error in case sensitive	Error in case insensitive	Accuracy for case sensitive	Accuracy for case insensitive
VLNet	5200	395	80	92.40%	98.46%

Experimental results for the known-fonts model are displayed in Table 7. Experimental result for known fonts ignoring I and 1. The accuracy rates of the case-sensitive and case-insensitive experiments exceeded the original results by 0.43% and 0.38%, respectively. Although the recognizability of I and l was lower with unknown fonts than with known fonts, the two characters were not easy to distinguish in either case.

Table 7. Experimental result for known fonts ignoring I and 1

Model	Test samples	Error in case sensitive	Error in case insensitive	Accuracy for case sensitive	Accuracy for case insensitive
VLNet	98800	1316	501	98.67%	99.49%

The objective of this study was to produce a network architecture that can run on a GPU or neural processing unit (NPU) platform without AI computing capabilities. Therefore, for verification, we opted to use Raspberry Pi 4B, which is easily obtained in the market. The experiment was conducted using a CPU to perform the CNN operation. The average prediction time per character ranged between 4.44 and 4.61 ms (Table 8 and Table 9), which confirms our assumption that character recognition can be performed by VLNet on an IoT device without AI computing capabilities.

Table 8. Model prediction time for unknown fonts

Model	Test sample	Time (s)	Time (ms/per character)
VLNet	5200	23.10	4.44
LeNet	5200	46.70	8.98
AlexNet	5200	1279.11	245.98
MobileNet V3	5200	853.55	164.14

Table 9. Model prediction time for known fonts

Model	Test sample	Times (s)	Time (ms/per character)
VLNet	98800	455.32	4.61
LeNet	98800	878.14	8.89
AlexNet	98800	22996.23	232.76
MobileNet V3	98800	16091.16	162.87

The experimental results presented in this study indicate that the character recognition time on Pi 4 ranged from 4.4 to 4.6 ms. General documents, such as receipts or bills, typically contain fewer than 200 characters, resulting in a

maximum processing time of approximately 0.92 s. These findings successfully meet our pre-recognition requirements, prompting us to integrate the research outcomes into the Plustek Q30 system. The Q30 system enables the automatic transmission of various images and metadata to predefined clouds tailored to specific characteristics based on different keywords. This advanced functionality eliminates the need for manual classification and uploading. For further reference, please refer to Figure 8, which displays the architecture diagram presented by Plustek at Computex 2023.

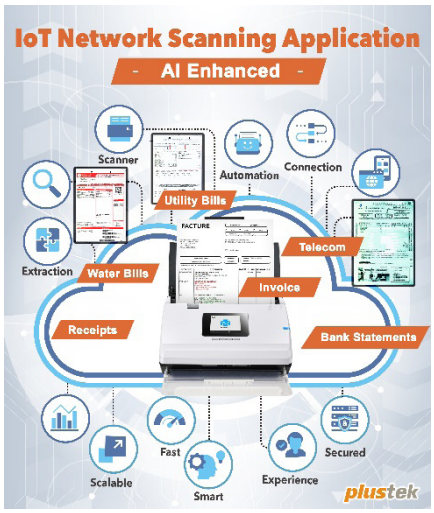


Figure 8. Plustek Q30 cloud application architecture diagram

The recognition process within Q30 follows a standard methodology, consistent with the majority of recognition approaches, as depicted in Figure 9. The process involves scanning data into image files through the Q30 system. After undergoing image segmentation, the target characters are extracted. Subsequently, 135 features of these characters are computed. These 135 features are subjected to classification utilizing VLNnet, resulting in the identification of the character in question.

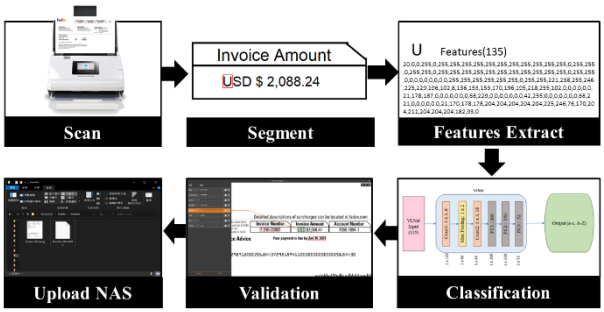


Figure 9. Plustek Q30 classification flow

Naturally, it is important to acknowledge that predictions generated by artificial intelligence systems are susceptible to errors. Therefore, a manual verification step is incorporated to ensure the accuracy of the obtained data. Depending on the specific requirements of the application domain, the manual verification step can be optionally deactivated. The data acquired through this process is employed for predictive data distribution, as illustrated in the example. For instance, if the system identifies a document as a FedEx Invoice, Q30 will autonomously allocate it to a designated temporary location. In this example, the document is stored within the "Invoice" directory on the NAS, encompassing both image files and associated media data.

Although MobileNet V3 demonstrated a comparable accuracy to VLNet, it exhibited an inferior execution speed on Pi 4. Moreover, although MobileNet V3 is intended for mobile devices, it still relies on an NPU to attain satisfactory performance. When running on a CPU, MobileNet V3 operates at least 36 times slower than VLNet. This reinforces the benefits of adopting the VLNet design when implementing neural networks on IoT devices.

5 Conclusion and Future Work

This study experimentally demonstrates that stroke features can be used to recognize unknown fonts. Our proposed VLNet employs two-layer two-dimensional convolutions and a three-layer fully connected layer and accepts our artificially induced stroke features as input. Thus, VLNet is effective for unknown font prediction; it can obtain results immediately without a graphics processing unit (GPU). We conducted comprehensive experiments to assess the ability of VLNet to recognize both the same letter in different fonts and different letters in the same font. Additionally, we compared VLNet with existing networks such as LeNet, AlexNet, and MobileNet V3, utilizing the same training data for a fair evaluation. The known-font recognizability of VLNet was 99.11%, whereas its unknown-font recognizability was 97.81% under the best conditions. Several factors affected the recognition results. The uppercase and lowercase forms of the letters C, O, P, S, U, V, W, X, Y, and Z have identical or very similar strokes; similarly, the characters I and l have the same strokes and are difficult to distinguish even with the human eye. These cases are also problematic in traditional image recognition methods. In the future, we plan to use layer analysis or sequence-to-sequence training to resolve this issue.

Acknowledgement

The authors wish to acknowledge the help of Plustek Inc. in data collection and the full support of the General Manager Lin of Plustek Inc. The authors extend their gratitude to Advanced View Inc. for providing the NVIDIA GPU that was used in the training model, as well as aiding in the successful completion of this study. The research production has been imported into Plustek's iKnow and SmartZone software, as well as the DOCapture cloud recognition platform. The method proposed in this study obtained an invention patent of the Republic of China with the patent number I775634. In addition, this work was supported in part by the Ministry of Science and Technology, Taiwan, R.O.C., under grants MOST 110-2222-E-992 -006 -.

This work was also supported in part by the Ministry of Science and Technology, Taiwan, R.O.C., under grants MOST 110-2222-E-992 -006 -.

Permission of Data and Material

Samples used in this study were obtained by printing and scanning the 20 fonts built into the Microsoft Windows 10 operating system. During the scanning process, angular rotation was applied to increase the variability between sample characters. In addition, according to the font redistribution FAQ for Windows, Microsoft does not place any restrictions on print output using these fonts unless using an application that is specifically licensed for home, student, or noncommercial use.

References

[1] A. Aberdam, R. Litman, S. Tsiper, O. Anschel, R. Slossberg, S. Mazor, Sequence-to-Sequence Contrastive Learning for Text Recognition, *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021, pp. 15297-15307.

[2] T. T. H. Nguyen, A. Jatowt, M. Coustaty, A. Doucet, Survey of post-OCR processing approaches, *ACM Computing Surveys (CSUR)*, Vol. 54, No. 6, pp. 1-37, July, 2022.

[3] X. Chen, L. Jin, Y. Zhu, C. Luo, T. Wang, Text recognition in the wild: A survey, *ACM Computing Surveys (CSUR)*, Vol. 54, No. 2, pp. 1–35, March, 2022.

[4] Y. Yang, L. Xu, C. Chen, English character recognition based on feature combination, *Procedia Engineering*, Vol. 24, pp. 159-164, 2011.

[5] S.-W. Lee, D.-J. Lee, H.-S. Park, A new methodology for gray-scale character segmentation and recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 10, pp. 1045-1050, October, 1996.

[6] I. B. Cruz, A. D. Sardiñas, R. B. Pérez, Y. S. Oliva, Learning optimization in a MLP Neural Network Applied to OCR, *MICAI 2002: Advances in Artificial Intelligence: Second Mexican International Conference on Artificial Intelligence Mérida*, Yucatán, Mexico, 2002, pp. 292-300.

[7] N. Sharma, B. Kumar, V. Singh, Recognition of off-line hand printed English Characters, Numerals and Special Symbols, *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, Noida, India, 2014, pp. 640-645.

[8] J. R. Quinlan, Induction of decision trees, *Machine learning*, Vol. 1, No. 1, pp. 81–106, March, 1986.

[9] Y. Li, J. Li, M. Li, Character Recognition Based on Hierarchical RBF Neural Networks, *Sixth International Conference on Intelligent Systems Design and Applications*, Jian, China, 2006, pp. 127-132.

[10] R. Arnold, P. Miklós, Character recognition using neural networks, *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, *Budapest, Hungary*, 2010, pp. 311-314.

[11] J. Bai, Z. Chen, B. Feng, B. Xu, Image character recognition using deep convolutional neural network learned from different languages, *2014 IEEE International Conference on Image Processing (ICIP)*, Paris, France, 2014, pp. 2560-2564.

[12] R. Ptucha, F. P. Such, S. Pillai, F. Brockler, V. Singh, P. Hutkowski, Intelligent character recognition using fully convolutional neural networks, *Pattern recognition*, Vol. 88, pp. 604-613, April, 2019.

[13] J. Parkinson, B. Khurana, Temporal order of strokes primes letter recognition, *The Quarterly Journal of Experimental Psychology,* Vol. 60, No. 9, pp. 1265-1274, September, 2007.

[14] Y. Y. Tang, B. F. Li, H. Ma, J. Lin, Ring-projection-wavelet-fractal signatures: a novel approach to feature extraction, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 45, No. 8, pp. 1130-1134, August, 1998.

[15] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE*, Vol. 86, No. 11, pp. 2278-2324, November, 1998.

[16] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Communications of the ACM*, Vol. 60, No. 6, pp. 84-90, June, 2017.

[17] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, Q. Le, Searching for mobilenetv3, *Proceedings of the IEEE/CVF international conference on computer vision*, Seoul, Korea (South), 2019, pp. 1314-1324.

[18] W. B. Lund, D. J. Kennard, E. K. Ringger, Combining multiple thresholding binarization values to improve OCR output, *Document Recognition and Retrieval XX*, February, 2013.

[19] N. A. Shaikh, Z. A. Shaikh, G. Ali, Segmentation of Arabic text into characters for recognition, *Wireless Networks, Information Processing and Systems: International Multi Topic Conference, IMTIC 2008*, Jamshoro, Pakistan, 2008, pp. 11-18.

[20] S. Khalid, T. Khalil, S. Nasreen, A survey of feature selection and feature extraction techniques in *machine learning*, 2014 Science and Information Conference, London, UK, 2014, pp. 372-378.

[21] K. K. M. Shreyas, S. Rajeev, K. Panetta, S. S. Agaian, Fingerprint authentication using geometric features, *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, Waltham, MA, USA, 2017, pp. 1-7.

[22] A. S. Tarawneh, A. B. Hassanat, D. Chetverikov, I. Lendak, C. Verma, Invoice Classification Using Deep Features and Machine Learning Techniques, *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, Amman, Jordan, 2019, pp. 855-859.

[23] E. M. d. A. Neves, A. Gonzaga, A. F. F. Slaets, A multi-font character recognition based on its fundamental features by artificial neural networks, Proceedings II Workshop on Cybernetic Vision, Sao Carlos, Brazil, 1996, pp. 196-201.

[24] H. Hosseini, B. Xiao, R. Poovendran, Google's cloud vision api is not robust to noise, *2017 16th IEEE*

international conference on machine learning and applications (ICMLA), Cancun, Mexico, 2017, pp. 101-105.

[25] J. Memon, M. Sami, R. A. Khan, M. Uddin, Handwritten Optical Character Recognition (OCR): A Comprehensive Systematic Literature Review (SLR), *IEEE Access*, Vol. 8, pp. 142642-142668, July, 2020.

[26] J. Kim, J. Kim, H. Kim, M. Shim, E. Choi, CNN-based network intrusion detection against denial-of-service attacks, *Electronics*, Vol. 9, Article No. 916, June, 2020.

[27] A. A. Barbhuiya, R. K. Karsh, R. Jain, CNN based feature extraction and classification for sign language, *Multimedia Tools and Applications*, Vol. 80, No. 2, pp. 3051-3069, January, 2021.

[28] W. Cheng, Y. Sun, G. Li, G. Jiang, H. Liu, Jointly network: a network based on CNN and RBM for gesture recognition, *Neural Computing and Applications*, Vol. 31, No. 1 supplement, pp. 309-323, January, 2019.

[29] J. H. Lee, K.-Y. Kim, Y. Shin, Feature Image-Based Automatic Modulation Classification Method Using CNN Algorithm, *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, Okinawa, Japan, 2019, pp. 1-4.

[30] V. Savitha, M. Karthick, T. Kalaikumaran, Parasitic Egg detection from Microscopic images using Convolutional Neural Networks, *Tamjeed Journal of Healthcare Engineering and Science Technology*, Vol. 1, No. 1, pp. 24-34, April, 2023.

[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*, April, 2015. https://arxiv.org/abs/1409.1556

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015, pp. 1-9.

[33] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778.

[34] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, *arXiv preprint arXiv:1704.04861*, April, 2017. https://arxiv.org/abs/1704.04861

[35] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, 2018, pp. 4510-4520.

[36] J.-M. Park, C. G. Looney, H.-C. Chen, Fast connected component labeling algorithm using a divide and conquer technique, *15th International Conference on Computers and their Applications (CATA)*, New Orleans, LA, USA, 2000, pp. 373-376.

[37] C.-H. Chen, Z.-H. Huang, K.-W. Huang, Recognition of Handwritten English and Digits Using Stroke Features and MLP, *2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCIS&ISIS)*, Ise, Japan, 2022, pp. 1-5.

Biographies

Chung-Hsing Chen received his master's degree at the Department of Information Management, National Sun Yat-Sen University, in 2006. Currently, he is the Director of Research and Development Department of Plustek Inc. His current research interests mainly include, network applications, embedded systems and AI image recognition.

Ko-Wei Huang received his PhD from the Institute of Computer and Communication Engineering, Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, in 2015. He is currently an Associate Professor at the Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Taiwan. His current research interests mainly include data mining, deep learning, evolutionary computing, and medical image processing.