Received September 13, 2018, accepted October 4, 2018, date of publication October 15, 2018, date of current version November 8, 2018.

*Digital Object Identifier 10.1109/ACCESS.2018.2876035*

# Baidu Meizu Deep Learning Competition: Arithmetic Operation Recognition Using End-to-End Learning OCR Technologies

**YUXIANG JIANG, HAIWEI DONG, (Senior Member, IEEE), AND ABDULMOTALEB EL SADDIK, (Fellow, IEEE)**
Multimedia Computing Research Laboratory, School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada
Corresponding author: Haiwei Dong (hdong@uottawa.ca)

**ABSTRACT** The end-to-end learning approaches were proposed for an arithmetic expression recognition task in the Baidu Meizu Deep Learning Competition by a deep convolutional neural network (DCNN) with parallel dense layers and component-connection-based detection pipeline with the convolutional recurrent neural network (CRNN) model. Two effective pipelines for DCNN and CRNN to identify long and complex expressions are presented and compared. In the first task, a DCNN connected to parallel dense layers for digital arithmetic operations was developed, which achieves 99.985% accuracy. In the second task, the CRNN with connectionist temporal classification was adopted, combined with the text region detection technique to recognize more complex pictures with both assignment operations and calculation formulas, which achieves 98.087% accuracy.

**INDEX TERMS** Optical character recognition, end-to-end learning, convolutional recurrent neural network.

## I. INTRODUCTION

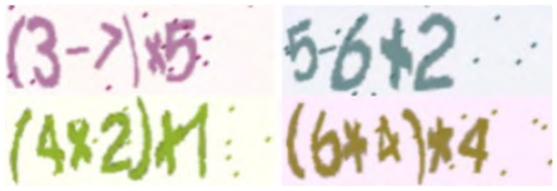
Optical character recognition (OCR) is a long-term research topic in the computer vision area, where a large number of applications have appeared to improve visual recognition efficiency. Parking lots need OCR to recognize vehicle license plate. Scanned documents need to be transformed from image text to editable text to build digital libraries. Even certain verification pictures can be cracked easily when an OCR algorithm is applied. An arithmetic operation-type verification picture begins to appear gradually since a simple character sequence is often easily identified. Although researches have been done and many methods have been widely used in the OCR, arithmetic operation recognition is different from the typical OCR task that has not been completely resolved [1], [2]. Online OCR cloud services include an online service by Baidu that has poor performance when processing images with handwritten-style arithmetic operations. Therefore, Baidu cloud service and Meizu jointly hosted the "Baidu Meizu Deep Learning Application Competition," aiming at improving their OCR system with the solutions presented in this competition. The competition provided two different datasets and was divided into two stages. The datasets are CAPTCHA-style, which is similar to handwritten style [3]. This CAPTCHA-style is a type of challenge-response test used in computing to determine whether or not the user is human. Thus, letters in the images are twisted, and interfering lines and spot noise are added in order to increase the recognition difficulty for the OCR system.

The main contributions of this paper are two pipelines of neural network models, whose network architectures are specifically designed for tagging sequence-like arithmetic operation in images with different complexity. Both pipelines are segmentation-free of characters. The DCNN with parallel dense layers is designed for short sequence-like operations with the improved structure of the convolutional neural network. The CRNN pipeline is designed for long and complex sequence-like operations. The datasets and the implementation of the pipelines are also presented in details. The complexity of these two models are compared, showing the scenes to be used.

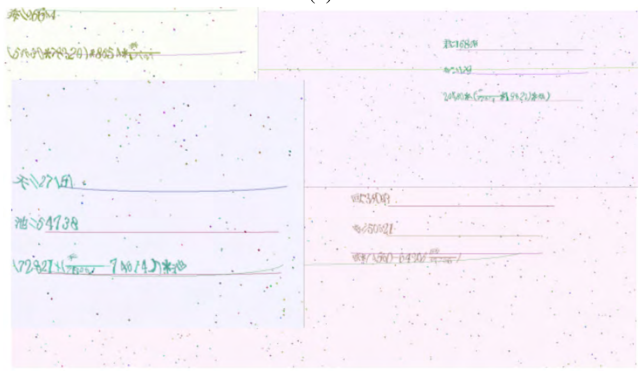
### A. STAGE LEVEL 1

In level 1,[1] 100000 data images have a fixed size (width: 180, height: 60, channel: RGB). Every image has one mathematical expression that contains 3 digits and 2 operators with

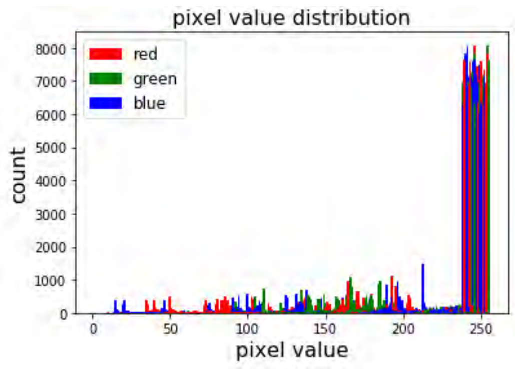
[1]Baidu Meizu Deep Learning Competition dataset level 1: https://www.kaggle.com/yuxiangjohn/simplearithmeticoperation/data



(a)



(b)



(c)

**FIGURE 1. Sample data images from the two stages are different. Pixel value are analyzed. (a) Level-1 images have fixed size and less margin. (b) Level-2 images are more complex than level-1 images. The image size varies with more noise added, and margins are different as well. (c) 30 randomly sampled images from the level-2 dataset where the pixel values are captured on RGB channels.**

or without a pair of the brackets. As a result, the expression lengths are different. The total number of classes that need to be differentiated is 15. The main noise in the images is spot noise. As the data images (Fig.1 (a)) have a neat format, there is no need to apply any pre-processing before fitting into a classifier.

The arithmetic operation recognition task can be considered as a multi-class and multi-label classification problem [4]. Each operation has several digits and operators between these digits such as the plus sign, minus sign, times sign, etc. Our task is similar to a mapping process that includes mapping the sequence in the image to editable texts. Commonly, characters in arithmetic equations have a strong contextual relationship. For example, brackets should appear as a pair in an operation; an operator cannot be followed by another operator. These kinds of contextual relationships ought to be utilized as useful information. Thus, the chosen classifier should have the ability to make use of th contextual relationships. Feature extraction from images is also very important. The mixed recognition task of digits and signs is not as simple as the task with only one kind of character; therefore, classifiers with strong ability to extract features of characters are able to solve the problem.

A few segmentation-based traditional approaches recognize arithmetic operation very well only under the condition that characters are neatly printed. Typical segmentation methods separate each character by the sliding window and then fit the segmented characters individually into a classifier to recognize the formula without using a large training set [5]. However, their drawbacks are obvious. Segmentation-based methods may fail when the arithmetic operations are written by hand. Firstly, segmentation between characters in these handwritten expressions is difficult because of joint letters. Secondly, the formula recognition is obviously more complex than only digit recognition or character recognition. Thirdly, for handwritten style arithmetic expression images, brackets and arithmetic operators occupy various spaces on an image. Lastly, random rotation of the characters and background noise also make the recognition difficult. Not only computers but also humans feel confused with recognizing handwritten characters. In practice, segmentation approaches do not always avoid the limitations of characters styles [6]. In our task, the type of images is the style on which segmentation is not capable of working perfectly. In the segmentation approaches, separating every character and then fitting every character into the classifier is inefficient and inaccurate in this competition.

## B. STAGE LEVEL 2

The level-2 task[2] is a more complex arithmetic operation recognition task than the level-1 task. In addition to the features in the level-1 dataset, the following changes were made to increase difficulty: image sizes are varied; the region position of formulas in the images is random; 27 Chinese characters are added to form the assignment formula; the total number of classes is increased to 43; and there are either two lines (one assignment formula and one arithmetic expression) or three lines (two assignment formulas and one arithmetic expression) in an image. The Chinese characters in the arithmetic expression represent a sequence of digits (no more than 6 digits) that were assigned to a Chinese character in the assignment lines. The divisions in the equation are in terms of fractions. As discussed for the level-1 task, the segmentation-based method still does not function well here.

As the regions of expression only appear in part of the image in the level-2 data (Fig.1 (b)) and the position of the regions is not fixed, region detection from a text image is a crucial step before recognition. The original text images are so large with a small region of text inside; thus, region detection is a good way to reduce the image size, reducing the training consumption simultaneously.

[2]Baidu Meizu Deep Learning Competition dataset level 2: https://www.kaggle.com/yuxiangjohn/complexarithmeticoperation/data

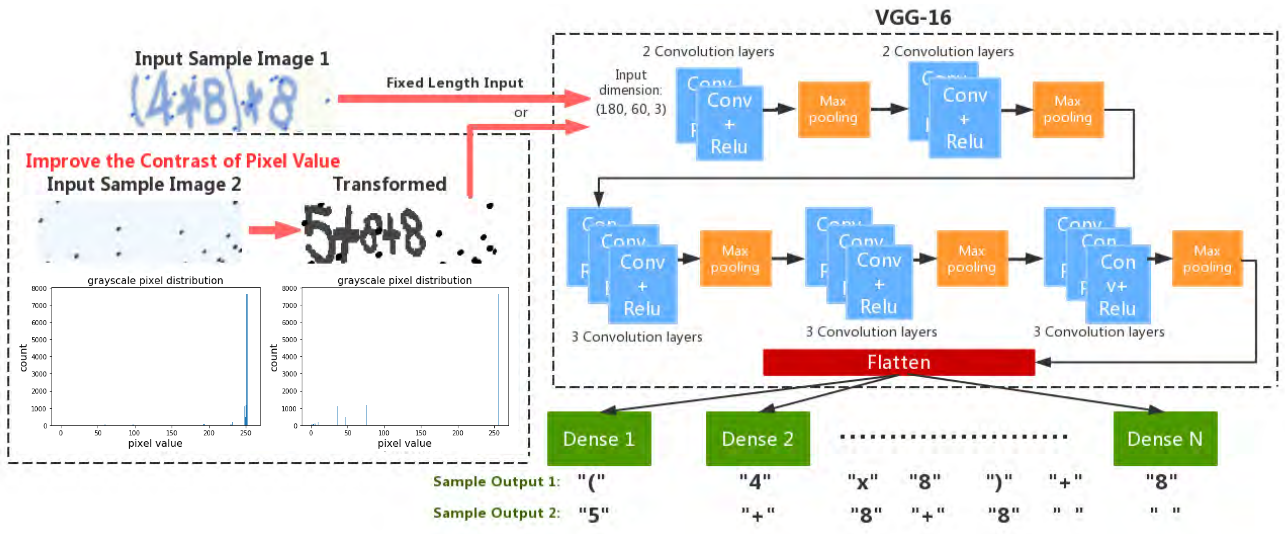


**FIGURE 2.** The preprocessing and DCNN with parallel dense layers. The left side is the exception handling process for images with little contrast of pixel values. The right side is DCNN referred to VGG-16. Each dense layer output takes responsibility for a single character where N is the max length of the sequence. Relu is rectified Linear Unit.

The main detection methods are typically connected-component (CC)-based [7], texture-based [8] and neural-network-based [9]. The connected component-based method detects the connected region of the compact characters on the image after noise reduction and obscuring process, and then the method finds the block of the connected region as the text region. In the texture-based method, edge detection methods such as canny and gradient are applied to determine the distinction between the background and the text area. In the neural-network-based method, part of the images is labeled in the text area. These images are trained by the neural network and generalized as a region detection tool. The popular networks we use today are Faster-RCNN [10] and YOLO [11].

To solve different detection tasks, the approaches have different performance; thus, the decision should be made according to the data features. In our dataset, pixel distribution is displayed as a histogram in Fig.1 (c). Dominating pixel values with the most counts are from the background (240-255). Characters only appear in small regions, and the counts of pixel values for the characters are very little. As a result, the distinction between the background and characters is the most significant feature to make detection easier. The CC-based method should be considered ahead of other detection methods.

### C. METHOD OVERVIEW

Here, we applied two segmentation-free approaches to recognize the arithmetic operation with a limited number of handwritten-style characters. To recognize the simple calculation formula in level 1, a deep convolutional neural network (DCNN) with parallel dense layers are applied. Each dense layer takes the responsibility of classifying each character in the order from left to right. For more complex and long equations with the assignment formula in the level-2 task, the recurrent neural network and DCNN were combined to make a classifier called a convolutional recurrent neural network (CRNN).

Firstly, the detected regions of interest were extracted from the processed images after thresholding and morphological image processing. This region detection method is categorized as connected component (CC)-based. This method was adopted to detect operations that only appear on part of an image. Extracted parts of images were recombined into a horizontal picture, and the sequence feature was used by the classifier to improve accuracy. In the following sections, the level-1 DCNN solution is briefly summarized, and we devote the main paragraphs to the level-2 CRNN solution that can solve both stages very well.

## II. LEVEL-1 SOLUTION: DCNN WITH PARALLEL DENSE LAYERS

Before predicting the expressions in images, a small number of images that cannot be identified (seem empty) with the naked eye need special processing. Pixel values of the background and expression from these images are confined to some specific range of values only. Humans are not capable of perceiving the contrast between the background and the expression. In this situation, histogram equalization is applied to improve the contrast of these images. In the histogram equalization (on the left side of Fig.2), the probability of an occurrence of a pixel of $l$ in the image is

$$p_l = \frac{n_l}{n}, 0 \leq l < L-1 \tag{1}$$

L is the total number of gray levels in the image (256), n is the total number of pixels, and $n_l$ is the number of pixels with level $l$.

The histogram equalized transform is defined by

$$T(k) = floor((L-1)\sum_{l=0}^{k} p_l) \tag{2}$$

where $floor(\cdot)$ rounds down to the nearest integer of $(\cdot)$, and $k$ is the transforming pixel intensity. $\sum_{l=0}^{k} p_l$ is the cumulative distribution value at k. The original histogram of the image is expanded. The conversion map the minimum value $l_{min}$ to 0 and the maximum value $l_{max}$ to $L-1$. This transform stretch the gray levels of the output image to occupy the entire dynamic range $0 \leq l \leq L-1$ (Fig.2).

The preprocessed images and normal images are then inputted to the DCNN network. The network architecture (Fig.2) of DCNN with parallel dense layers has convolutional layers, pooling layers, and flatten layers, identical with common DCNN. The processed images with a fixed shape are fit into the input layer. In the middle of the network, the structure of the model is referred to as VGG-16 [12], which extracts features from the input images. In the output layer, the single dense layer is replaced with several parallel dense layers of which the quantity is equal to the maximum sequence length that appears in the dataset. Each dense layer makes use of the features extracted from the convolutional layers to output the probability of each character from the sequence. For each dense layer, multi-class cross-entropy is adopted as the loss function,

$$e_1 = -\sum_{i \in n} \sum_{j \in m} y_{ij} log(p_{ij}) \tag{3}$$

where $e_1$ is the multi-class cross entropy, $n$ is the class set, and $m$ is the objects set to classification, $y_{ij} = 1$ if $j$ belongs to class $i$; otherwise, $y_{ij} = 0$, and $p_{ij}$ is the probability of $j$ belongs to class $i$.

The entire model loss is the sum of every multi-class cross entropy in each dense layer ($loss = e_{11} + e_{12} + ... + e_{1N}$, where N is the number of dense layers). The sequence order is from left to right. The training goal is to minimize the model loss. The model extracts the feature information deeper with the increase of convolution filters. The abstract information going through the flatten layer becomes a one-dimensional vector, which contains all feature information. This information is sent to different dense layers, which obtain the desired information with high-weighted features. Every dense layer focuses on the features that are considered the most important related to the output. Actually, each dense layer pays the most attention to the certain area related to the characters that appear most frequently in this area. As the image sequences are trained as a whole, the contextual relationships such as order are also learned by the DCNN. This model eliminates error during the cropping process, which leads to inaccurate output from the model. As a result, it can be directly learned from sequence labels with small length, avoiding segmentation.

The DCNN pipeline in Algorithm 1 is suitable for images with small length sequences. The longer sequence means only small areas of the image are considered to have related information for a single dense layer, and the other area would be likely considered as background noise. If the number of dense layers is large, it is more difficult for the model to converge. Thus, the CRNN method is used in the level-2 task to improve the performance of longer sequence recognition by using more characteristics of the end-to-end sequence.

**Algorithm 1** DCNN pipeline

```
Input:
    image $x$ with width:160, height:60, depth:3, maximum
    sequence length: $n$, threshold: $T$
Output:
    Probability set of sequence labels $P : (P_1, P_2, ..., P_n)$
    Initialization: histogram transform
1: Calculate the difference of range $R$ from histogram
2: if distribution is narrow ($R < T$) then
3:     Do histogram equalization to obtain image $x_e$
4:     Replace the original image $x = x_e$
5: end if
6: Input $x$ to DCNN network
7: return $P$
```

## III. LEVEL-2 SOLUTION: CRNN FOR END-TO-END LEARNING

As previously discussed, complex arithmetic expression images have sequence different sizes and locations inside images. The number of Chinese characters for assignment varies as well. For better accuracy performance and less GPU consumption, a pipeline needs to be built. The CRNN model in the pipeline can also solve the task in the level-1 task. The entire pipeline (Fig.3) is presented in Algorithm 2, which consists of the following two parts: *operation region detection* and *operation recognition*.

### A. OPERATION REGION DETECTION

The proposed region detection method is based on the sequence connectivity. This method is a practical approach to find the region of interest as preprocessing [13], which is capable of locating the position of the sequence in different lines from an image, which extracts the rectangular region containing the sequence. The flow process and the corresponding output of each step with the related technique of the proposed detection algorithm are demonstrated in the left part of the pipeline in Fig.3. This presented method is especially suitable for the chosen dataset and is available to the sequence with obvious block-shaped distribution in the image.

1) *Apply Sobel filter:* Original images are transformed from RGB channel to grayscale images, which only capture the brightness information. The interfering lines beside or under each sequence text confuse the actual regions. Thus, a horizontal Sobel operator filter calculates the gradient in the horizontal direction, which eliminates the interfering lines.
2) *Binarization:* Threshold the filtered image so that the contrast of the gradient in the image is more sharp. The binary image only contains black and white.
3) *Median blur:* Replace each pixel with a regional average in a small area. After this step, the text region becomes smooth, and the spot noise is eliminated.

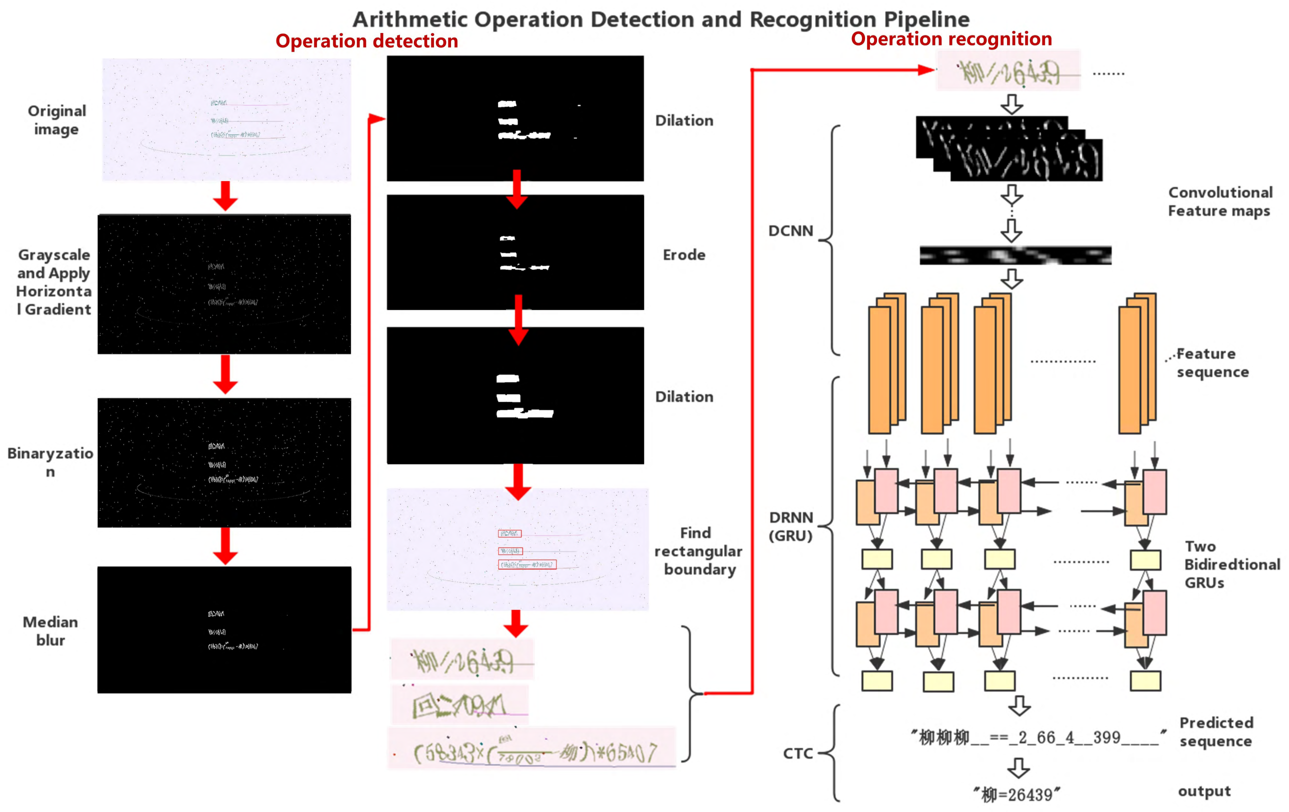


**FIGURE 3.** Full pipeline for CRNN end-to-end learning includes operation detection (left side) and operation recognition (right side).

**Algorithm 2** CRNN pipeline

```
Input: original image $x$, rectangle threshold $T$
Output: sequence labels set $P$
 1: Convert $x$ to grayscale image $x_g$
 2: Apply horizontal gradient to $x_g$ to obtain $x_h$
 3: Binary $x_h$ to obtain $x_b$
 4: Dilate $x_b$ to obtain $x_{d1}$
 5: Erode $x_{d1}$ to obtain $x_e$
 6: Dilate $x_e$ to obtain $x_{d2}$
 7: Find rectangles set $r$ in $x_{d2}$ and calculate areas set $a$
 8: $i = 0$
 9: while $r$ is not empty and $a_i > T$ do
10:     Extract rectangle as $x_i$, $i++$
11: end while
12: Input $x_i$ to CRNN network
13: return $P$
```

4) *Dilation:* The dilation process is a category of morphological transformation. Since the image have been binarized, we can directly consider the binary dilation. The binary image is A, and B is a structuring element. The dilation of A by B is defined by the following:

$$A \oplus B = \bigcup_{b \in B} A_b \tag{4}$$

where $A_b$ is the translation of A by b. The value b is one of the elements in kernel B. B can be understood as a sliding window applied to A, in which the locus of the points is covered by B when the center of B moves inside A. For each pixel in A, superimpose the center of B. Each pixel of every superimposed B is included in the dilation of A by B. Intuitively, the white pixels of the image are dilated. Thus, the sequence regions forms a connected region.

5) *Erode:* The erosion process is also a kind of morphological transformation, which can be considered as the opposite of dilation. In intuition, the region of the connected region is sharpened, and the remaining noise regions are removed. This step ensures the small spot noise regions are removed.

6) *Dilation:* The same as previous dilation step. Dilate the image again to ensure the connectivity of each sequence region.

7) *Find the rectangular region:* Find a curve joining all continuous points along the boundary of each sequence region. To obtain a rectangular region, four contours of the curve are examined. Then, the masked image with the rectangular region of interest is detected.

The masked image is obtained after completion of the above procedure, and each rectangular region is cropped as the input image to the model. The input images have the proper size to fill into the network, and the arithmetic operation fills an image.

### *B. OPERATION RECOGNITION*

The CRNN (convolutional recurrent neural network) model is a combination of DCNN and deep recurrent neural network (DRNN). As the combination of two different types of networks, CRNN makes use of the property of DCNN on the information extraction from image data. At the same time, CRNN inherits the ability of DRNN, which uses internal memory to process contextual relationships. It is unlimited to the lengths of the sequence-like data, which have been widely used in the application of speech recognition and music classification.

The CRNN model structure (shown in the right part of Fig.3) consists of three portions, including the DCNN component, the DRNN component and the CTC component with the flow from top to the bottom. On the top, a DCNN net with the structure modified from VGG-16 excluding the dense layer is utilized as the DCNN component. DCNN extracts the feature sequence from each input image automatically.

#### 1) DCNN COMPONENT FOR SEQUENCE FEATURE EXTRACTION

In the CRNN model, a standard CNN model referring to VGG-16 was applied without any fully connected layer. Two-dimension convolution and max-pooling were taken as the components of DCNN. These two components were used to extract the sequential features representation from the input image. After reshaping the original image, all input images have the same size. The feature sequence was extracted from the feature maps as the input of the DRNN layer. In the feature sequence, every feature vector was generated from left to right in the feature and associated with the certain receptive field on the image.

#### 2) DRNN COMPONENT FOR SEQUENCE LEARNING

Two deep bidirectional RNNs were established after the DCNN component. In the proposed approach, the gated recurrent unit (GRU) was used in the DRNN layer. The GRU used in the DRNN layer is easier to train and avoid the gradient vanish problem than a simple RNN. A GRU has two gates that are a reset gate and update gate. Compared to the long-short-term memory (LSTM) which has tree gates, the GRU is less complex and more efficient computationally [14]. GRUs make each recurrent unit adaptively capture dependencies of different time scales. As a recurrent layer, the GRU layer has a strong ability to capture the contextual information in the sequence. It labels each frame in feature sequence $X = (x_1, x_2, ..., x_T)$, where T is the length of the sequence. Then, it predicts the probability of a character appear in each frame as output $Y = (y_1, y_2, ..., y_T)$.

In this arithmetic recognition task, the mathematical structure of each equation is not fixed. It means the sequential information does not only rely on the past contexts but also relies on the future contexts. Reflecting a text image, each character has a relation on feature sequences from both left and right. A single GRU layer is directional, which means only the past features play a role in the prediction. Therefore, two GRU layers were combined as bidirectional GRU (BiGRU) with forward and backward directions to complement each other. This bidirectional structure is often used by sequence speech recognition [15], [16]. To make up for the lack of a single layer, the stack of several combined forward and backward GRUs allows for more abstracted features to have a deep fitting. In our network, we stacked two BiGRU layers. Every GRU has 128 units. The first BiGRU output values from forward and backward GRUs are added as the input of the second BiGRU. The outputs from the second BiGRU are concatenated as the input to the dense layer.

#### 3) CTC COMPONENT FOR INTERPRETATION

Connectionist temporal classification (CTC) is a specifically designed technique for the RNN output layer proposed by Graves *et al.* [17]. A CTC can map the RNN outputs to labels. The CTC has a Softmax output layer with one more unit than the labels in the lexicon (digits, signs, Chinese). $L$ is an extra unit that represents the observation of a 'blank' that means no label. Thus, we merged the 'blank' in the full lexicon L. The input from the GRU layer to the CTC layer is $Y = (y_1, y_2, ..., y_T)$. $y_t$ is the probability distribution over the characters in L. $\pi$ is the prediction sequence according to each $Y$. The mapping from $X$ to the actual output sequence $Z = (z_1, z_2, ..., z_U)$ (i.e., $U \leq T$) is processed by first removing the repeated labels and then removing the blanks. The entire recognition process can be understood as follows: firstly, DCNN perceives the receptive field on an image and extracts the feature sequence according to the receptive fields; secondly, the GRU makes use of the feature sequence to output the raw sequence probability distribution; and finally, CTC translates the results into the actual sequence output. This process is shown in Fig.4. The loss function can be described as follows:

$$e_2 = -\sum_{(X,Z)\in S} log(p_{i(Z|X)}) \tag{5}$$

where $S$ is the pair set of (X, Z). The training goal is to minimize this loss function. Thus the DCNN and the BiGRU network can be jointly trained.

## IV. TRAINING AND RESULTS

### *A. DATA AND SETTING*

The datasets for each level are randomly partitioned into 90% (90000 images) for training and the rest (10000 images) for validation, respectively. To train the DCNN and CRNN models for level-1 dataset, training images were directly fit in two neural network models. Before training the CRNN model for level-2 dataset, we adopted the pipeline for sequence region detection. The images of the detected regions then fed the CRNN model. During the training process, the mini-batch size was set to 128, and the optimizer was "Adam" [18]. Both DCNN and CRNN were trained on the Amazon EC2 cloud

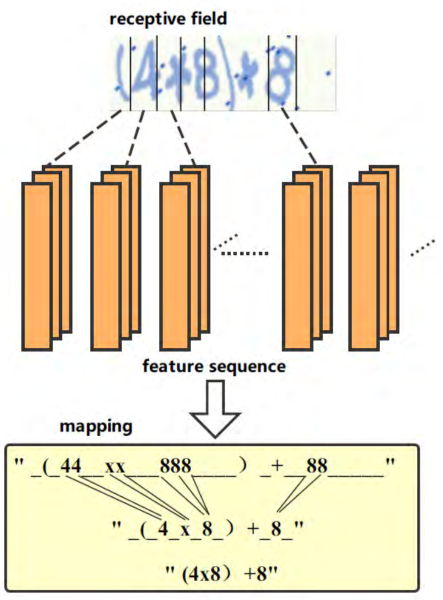


**FIGURE 4.** Visualizing the mapping process, which also explains the feature relationship from the receptive field to the final output.

server with a Tesla K80 GPU. The framework employed was Keras, a high-level neural networks API running on the top of Tensorflow (a framework maintained by Google).

### B. PARAMETERS AND SENSITIVITY

The important configurations for the network are shown in Table I. The convolution structures are similar except for the number of filters. We only show the top 2 or 3 convolution layers where the similar structures are presented as "Repeat" layer. The filter quantity in the convolution layer is 2 times more than in the last convolution layer before max-pooling. The total number of trainable parameters of DCNN model is 15,001,520, which mainly includes 14,714,688 from convolution layer and 40,976 from each parallel dense layer. The total number of trainable parameters of CRNN model is 1,797,165, which mainly includes 583,680 from BiGRU and 925,440 from convolution layers. The difference between these two model in the number of parameters is mainly reflected in the number of convolution layers. The CRNN has fewer convolution layers than DCNN. That means the CRNN doesn't rely too much on deeper and more abstract features from the images.

The number of filters affects the depth of the feature maps in convolution layers. The more filters it has, the more features it obtains from the convolution. The batch normalization is following each convolution layer. The output unit of BiGRU is set as 128, which should be longer feature sequences than the number of symbols to input to the CTC layer. To avoid the overfitting problem, L2 regularization was applied to the convolution layer [19], and the dropout was set to 0.25 to randomly drop 25% parameters before the dense layer [20]. The L2 regularization adds a regular term to the error function. It limits the size of parameters to limit the space of the model, reducing the impact of excessive enhancement from the weights of some features. At each gradient descent, the corresponding activation is normalized by the mini-batch, so that the mean is 0 and the variance is 1, which allows a higher learning rate in order to accelerate the training.

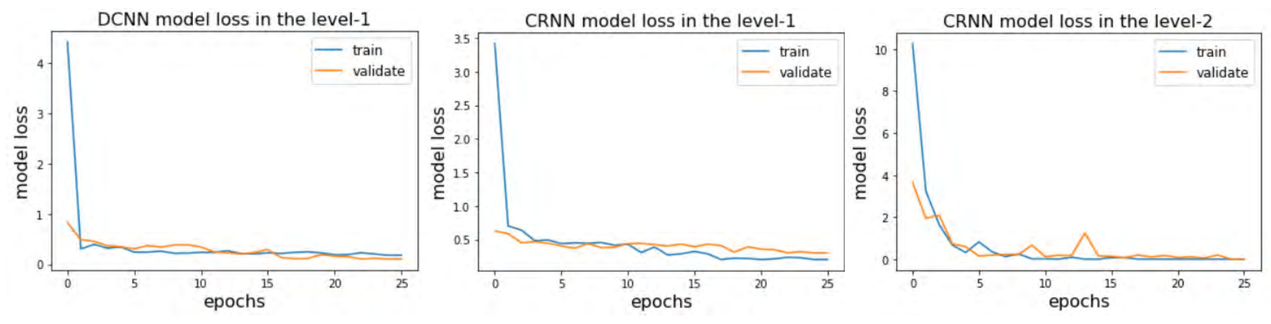
**TABLE 1.** Important configurations of the two networks. (a) DCNN configuration. (b) CRNN configuration.

(a)

| Component | Configurations#1 |
|---|---|
| Input layer | input shape: 180×60×3 |
| Convolution2D | #filters: 32, kernel: 3×3, same padding, $L2 : 5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Convolution2D | #filters: 32, kernel: 3×3, same padding, $L2 : 5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Maxpooling | pool size: 2×2, same padding |
| Repeat | Repeat the above layers 1 time<br>#filter is twice the previous. |
| Convolution2D | #filters: 128, kernel: 3×3, same padding, $L2 : 5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Convolution2D | #filters: 128, kernel: 3×3, same padding, $L2 : 5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Convolution2D | #filters: 128, kernel: 3×3, same padding, $L2 : 5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Maxpooling | pool size: 2×2, same padding |
| Repeat | Repeat the above layers 2 times<br>#filter is twice the previous each time. |
| Dropout | ratio: 25% |
| Dense | units:7; 7 parellel dense layers |

(b)

| Component | Configurations#2 |
|---|---|
| Input layer | input shape: 400×100×3 |
| Convolution2D | #filters: 32, kernel: 3×3, same padding, L2: $5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Convolution2D | #filters: 32, kernel: 3×3, same padding, L2: $5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Convolution2D | #filters: 32, kernel: 3×3, same padding, L2: $5 \times 10^{-5}$ |
| BatchNormalization | - |
| Activation | relu |
| Maxpooling | pool size: 2×2, same padding |
| Repeat | Repeat the above layers two times<br>#filter is twice the previous each time. |
| Reshape | input:(w, h, c), output:(w, h×c) |
| Dense | units: 50 |
| BiGRU | hidden uints: 128 |
| Merge | add |
| BiGRU | hidden uints: 128 |
| Merge | concetenate |
| Dropout | ratio: 25% |
| Dense | units: 44 |
| CTC | Input: dense; labels;<br>feature sequence length; label length |



(a) DCNN loss in level 1. (b) CRNN loss in level 1. (c) CRNN loss in level 2.

**FIGURE 5. The training loss converged in a few epochs. (a) and (b) are the training process in level 1. (c) is the training process in level 2.**

### C. RESULT

Only when all the symbols in a picture are recognized correctly, and the order of the symbols in each formula is correct, this picture is correctly recognized. The performance of the result is The proposed DCNN and CRNN approaches[3] worked well in the level-1 stage. The benchmark of level-1 is the accuracy of 90%. Our best accuracy achieved 99.985%. In the level-2 stage, the benchmark is accuracy of 65%. Only the CRNN method was used in this stage. The best accuracy was 98.087%. The presented pipelines ranked 2% of all teams in the competition. Compared to the benchmark, the proposed pipelines have significant improvement. The loss from the training process is visualized in Fig.5. The loss was decreasing to almost 0. Therefore, DCNN and CRNN fit well to the level-1 dataset. The training converged quickly in a few epochs. The CRNN model spent more epochs for fitting than the DCNN model. In level 2, the CRNN model needed more epochs to converge than the models in level 1. The results show that the CRNN still performed well on the more complex sequence even when fractions appeared in the equation.

## V. CONCLUSION

In this paper, the recognition problem of CAPTCHA-style arithmetic operation is solved by two DCNN and CRNN pipelines in the competition. Both pipelines performance well beyond the benchmark. In practical use, the pipelines can be used to recognize the CAPTCHA-style verification images in arithmetic operation format. The CRNN pipeline above provides a practical end-to-end approach to recognize arithmetic operations. To find the region of interest in the sequential text region, a component connection-based approach was adopted, which crops the text region from the original image by morphological processing. For the short equation, both DCNN with parallel dense layers and CRNN can solve the problem well. It is noted that the computation consumption of CRNN is lower than DCNN when achieving similar accuracy because CRNN needs fewer convolution layers. Thus, CRNN is a better choice when encountering operation recognition tasks. For longer equations, CRNN definitely would be a more appropriate choice than DCNN due to its strong ability to capture the contextual information.

[3] Link to code: https://github.com/YuxiangJohn/Arithmetic_Breaker_Baidu

## REFERENCES

[1] U. Garain and B. B. Chaudhuri, “Recognition of online handwritten mathematical expressions,” *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 6, pp. 2366–2376, Dec. 2004.

[2] S. A. Naik, P. S. Metkewar, and S. A. Mapari, “Recognition of ambiguous mathematical characters within mathematical expressions,” in *Proc. Conf. Elect., Comput. Commun. Technol. (ICECCT)*, Feb. 2017, pp 1–4.

[3] L. Ahn, M. Blum, and J. Langford, “Telling humans and computers apart automatically,” CMU, Mount Pleasant, MI, USA, Tech Rep. CMU-CS-02-117, 2004.

[4] F. A. Thabtah, P. Cowling, and Y. Peng, “MMAC: A new multi-class, multi-label associative classification approach,” in *Proc. 4th IEEE Int. Conf. Data Mining*, 2004, pp. 217–224.

[5] N. Zheng, Q. You, G. Meng, J. Zhu, S. Du, and J. Liu, “50 years of image processing and pattern recognition in China,” *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 33–41, Nov. 2008.

[6] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 11, pp. 2298–2304, Nov. 2017.

[7] H. I. Koo and D. H. Kim, “Scene text detection via connected component clustering and nontext filtering,” *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2296–2305, Jun. 2013.

[8] K. I. Kim, K. Jung, and J. H. Kim, “Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 12, pp. 1631–1639, Dec. 2003.

[9] R. Girshick, J. Donahue, and T. Darrell, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.

[12] K. Simonyan and A. Zisserman. (2014). “Very deep convolutional networks for large-scale image recognition.” [Online]. Available: https://arxiv.org/abs/1409.1556

[13] C. Scharfenberger, A. G. Chung, A. Wong, and D. A. Clausi, “Salient region detection using self-guided statistical non-redundancy in natural images,” *IEEE Access*, vol. 4, pp. 48–60, 2016.

[14] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. (2014). “Empirical evaluation of gated recurrent neural networks on sequence modeling.” [Online]. Available: https://arxiv.org/abs/1412.3555

[15] K. Choi, G. Fazekas, M. Sandler, and K. Cho, “Convolutional recurrent neural networks for music classification,” in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2017, pp. 2392–2396.

[16] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 6645–6649.

[17] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 369–376.

[18] D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[19] A. Ng, "Feature selection L1 vs. L2 regularization, and rotational invariance," in *Proc. Int. Conf. Mach. Learn.*, 2004, p. 78.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

**YUXIANG JIANG** received the B.Eng. degree in Internet of Things engineering from Southwest Jiaotong University, China, in 2016, and the M.Sc. degree in electrical and computer engineering from the University of Ottawa, Canada, in 2018. His research interests include Internet of Things, computer vision, and deep learning.

**HAIWEI DONG** (M'12–SM'16) received the Dr. Eng. degree in computer science and systems engineering from Kobe University, Japan, in 2008, and the M.Eng. degree in control theory and control engineering from Shanghai Jiao Tong University, China, in 2010. He was a Post-Doctoral Fellow with New York University, a Research Associate with the University of Toronto, and a Research Fellow (PD) with the Japan Society for the Promotion of Science. He is currently a Research Scientist with the University of Ottawa and a Senior Research Engineer with Huawei Technologies Canada. He is a licensed Professional Engineer. His research interests include robotics, multimedia, and artificial intelligence.

**ABDULMOTALEB EL SADDIK** (M'01–SM'04–F'09) is currently a Distinguished University Professor and the University Research Chair with the School of Electrical Engineering and Computer Science, University of Ottawa. He has authored or co-authored four books and over 550 publications. His research focus is on multimodal interactions with sensory information in smart cities. He received research grants and contracts totaling more than $18 M. He was a recipient of the IEEE I&M Technical Achievement Award and the IEEE Canada Computer Medal. He chaired more than 40 conferences and workshops. He has supervised over 120 researchers and received several international awards, among others are the ACM Distinguished Scientist, a fellow of The Engineering Institute of Canada, and a fellow of the Canadian Academy of Engineers.