Dean&Francis

ISSN 2959-6157

Research and implementation of handwritten formula recognition system using feedback mechanism

Yifan Wu

Northeastern University at Qinhuangdao, Qinhuangdao, Hebei, 066003, China

Abstract:

Handwritten formula recognition systems have significant application value in education, scientific research, and smart note applications. However, due to the complexity and irregularity of handwritten formulas, improving the accuracy and efficiency of recognition has been a challenging research hotspot. This paper focuses on enhancing handwritten formula recognition methods and proposes a feedback system based on incremental learning, which improves recognition accuracy and enables user personalization. The article utilizes CNN convolutional neural networks for model training, and a regularization-based incremental learning system is used to enhance the model. The proposed algorithm is validated using the mathematical formula data provided by the Competition Organization for Recognition of Handwritten Mathematical Expressions (CROHME) and user experience feedback. Results show the feasibility of the proposed method.

Keywords: Handwritten formula recognition, Incremental learning, Feedback mechanism, Convolutional neural network

1.Introduction

With the rapid popularization of mobile devices and the rapid development of related technologies, more and more scenarios require the input and recognition of handwritten mathematical formulas. This poses high requirements for handwritten formula recognition systems to be fast, stable and accurate. However, handwritten characters have variable styles, and the recognition process requires attention to contextual information. In addition, mathematical formulas have complex structures, often presenting complex two-dimensional structures or even containing multiple structures nested together. Unlike online recognition, handwriting information is missing in offline images, which can only rely on the image itself as a single source of information to recognize formulas. This makes the research of handwritten mathematical formula recognition challenging. The research history of handwritten formula recognition consists of three main phases. The earliest studies used traditional methods such as template matching and rule-based methods, which are highly sensitive to image quality and handwriting style, and are difficult to cope with diverse handwritten expressions.

With the introduction of statistical methods, techniques such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) were used to process the sequence information of handwritten characters, significantly improving the robustness of recognition. However, the real breakthrough comes from the application of deep learning methods. In recent years, deep learning techniques such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) and Generative Adversarial Networks (GAN) have been widely used in handwritten formula recognition. These methods can not only automatically learn image features, but also deal with complex handwriting styles, which greatly improves the recognition accuracy and efficiency, and makes handwritten formula recognition technology enter a new stage of development. On this basis, researchers have proposed a variety of methods and models in an attempt to further improve the behavior of the recognition system. For example, Xing [1] proposed the SE_YOLOv4 recognition model embedded in the SE module, which improves the recognition accuracy by enhancing the expressive ability of the feature map, while Zhou [2] proposed a structure based on ResNet and Transformer network, which utilizes the feature extraction ability of ResNet and the global attention mechanism of Transformer to better deal with complex formulas and structural nesting. Although these methods have achieved remarkable results in improving accuracy and recognition efficiency, they still have certain limitations, especially when facing highly variable handwriting styles and complex mathematical formulas, the recognition results are still not satisfactory. However, even after using advanced deep learning techniques, existing handwriting formula recognition systems still face many problems. For example, complex formulas and varied handwriting styles are poorly handled, and the model is not sufficiently adaptable in real-world scenarios. In addition, existing systems generally lack an effective user feedback mechanism, which makes it difficult for models to be continuously optimized and improved in practical applications. User feedback plays an important role in improving the behavior and adaptability of the model, and through effective use of user feedback, the model parameters and structure can be dynamically adjusted, errors can be corrected in a timely manner and the recognition strategy can be optimized, so as to continuously improve the accuracy and reliability of recognition. Therefore, how to design a handwritten formula recognition system that can be dynamically adjusted and optimized, and can effectively utilize user feedback, has become an important problem to be solved. Introducing a user feedback mechanism can not only significantly improve the accuracy and reliability of recognition, but also enable the system to be optimized and improved over time and with the actual needs of users. By these means, it can not only significantly improve the user experience, but also help to promote the further development of handwritten formula recognition technology in various applications such as education, scientific research and engineering. System design that effectively utilizes user feedback has become a key path to enhance handwritten formula recognition technology, and in-depth research and innovation are urgently needed. Only through continuous optimization and improvement can the handwritten formula recognition system be intelligent and practical, providing users with more efficient and accurate services.

2.Methodology Overview

In this paper, a handwritten formula recognition feedback system based on incremental learning algorithm is proposed in order to personalize and improve the accuracy of handwritten formula recognition, the structure of the algorithm is shown below. The incremental learning is implemented using the LwF [3] algorithm, provided that a trained model is already available.

LEARNINGWITHOUTFORGETTING:

Start with:

θ_s: shared parameters

θ_o: task specific parameters for each old task

X_n, Y_n: training data and ground truth on the new task

Initialize:

$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*

$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*

Train:

Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*

Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*

$$\theta_s^*,\ \theta_o^*,\ \theta_n^* \leftarrow \underset{\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n}{\text{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$$

Figure 1

ISSN 2959-6157

First, the shared parameters (θ_s) and the specific parameters (θ_0) for each old task are initialized, along with the training data (X_n) and the corresponding true labels (Y_n) for the new task. Then, the output of the old task on the new data (Y_0) is computed by a convolutional neural network (CNN) and the parameters of the new task (θ_n) are randomly initialized.

In the training phase, the the old task outputs ($\hat{Y}_0$)and the new task outputs($\hat{Y}_n$)are computed separately. Then, the loss function is used to combine the output of the old task, the true label and the new task loss for optimization with the goal of minimizing the combined loss function. In this process, the LwF algorithm maintains the old task knowledge by sharing the parameters and the regularization term (R), while introducing the new task learning to ensure that the model does not forget the information of the old task while learning the new task.

3.Research and implementation

3.1 Model Training

3.1.1 Image loading and preprocessing

Load and scale the training set images to 32x32 pixels and process them into grayscale maps. A thread pool is used to improve loading efficiency, while images and labels are stored in a list and finally randomly disrupted to increase training diversity. The loaded image data is converted to a NumPy array and normalized so that the pixel values are between 0 and 1 to improve model training.

3.1.2 CNN model [4]

Model has 3 layers.

Convolutional Layer: use Conv2D for feature extraction, combined with BatchNormalization layer to accelerate training and stabilize the model.

Pooling Layer: downsampling is performed using MaxPooling2D to reduce the dimensionality of the feature map.

Fully connected layer: a Dense layer is used for classification in the final part of the model, and the activation function uses softmax to obtain probability distributions for each category.

(1) Compilation

The model is compiled using the Adam optimizer, and the loss function is chosen as cross-entropy to monitor classification accuracy.

(2) Learning rate adjustment and callback functions

Various callback functions, including EarlyStopping and ReduceLROnPlateau, were used to preserve the best model, avoid overfitting, and dynamically adjust the learning rate. EarlyStopping was used to stop the training early to avoid overfitting and to stop the training if the validated accuracy no longer improved in 15 rounds of training. ReduceLROnPlateau is used to automatically adjust the learning rate during training, usually reducing the learning rate to help the model learn better when its performance stagnates, and reducing the learning rate if the validation accuracy no longer improves in 6 rounds of training.

(3) Data Enhancement

Various callback functions, including EarlyStopping and ReduceLROnPlateau, were used to preserve the best model, avoid overfitting, and dynamically adjust the learning rate. EarlyStopping was used to stop the training early to avoid overfitting and to stop the training if the validated accuracy no longer improved in 15 rounds of training. ReduceLROnPlateau is used to automatically adjust the learning rate during training, usually reducing the learning rate to help the model learn better when its performance stagnates, and reducing the learning rate if the validation accuracy no longer improves in 6 rounds of training..

3.1.3 Test Methods

The training and test sets are evaluated using loss and accuracy metrics, and the final training and validation accuracies are output. Subsequently, the model's classification effect is analyzed in detail through the confusion matrix and classification report, which includes the precision, recall, F1-score, and support for each category. The confusion matrix shows the correspondence between the real and predicted categories. By calculating and printing the classification report and the confusion matrix, one can get a detailed picture of the model's performance on the training and test sets. The classification report provides information such as precision and recall for each category, while the confusion matrix visualizes the model's prediction on each category, especially the misclassification.

3.2 Feedback Mechanisms

3.2.1 Incremental learning algorithms

Incremental learning methods include the following categories:

Fine-tuning: Fine-tuning is not guided by the parameters and samples of the old task, so the model's performance on the old task will almost certainly deteriorate, i.e., catastrophic forgetting occurs.

Joint Training: Joint training is equivalent to re-training the model on all known data, which gives the best results, and is often considered the 'upper bound of performance for incremental learning', but is too expensive to train.

Feature Extraction: Feature extraction only trains the

model, and the shared parameters are not updated. Although it does not affect the model's performance on the old task, it cannot effectively capture the unique features of the new task, and the performance on the new task is usually unsatisfactory.

Each approach has its drawbacks: feature extraction performs poorly on new tasks; fine-tuning leads to performance degradation on old tasks; joint training becomes unwieldy as more tasks are added and is not possible to implement when training data for old tasks is not available.

The LwF algorithm used in this paper is a training method between joint training and fine-tuning training. The characteristic of LwF is that it can update without using the data from the old task. The main idea of the LwF algorithm comes from knowledge distillation, i.e., making the prediction of the new model on the new task similar to the prediction of the old model on the new task.

Specifically, the LwF algorithm initially retrieves the old model's predictions for the new task, then incorporates the distillation loss from the new model's output into the loss function. Following this, it employs a fine-tuning strategy to train the model on the new task, ensuring that the training process does not excessively adjust the parameters of the old model, which could compromise the new model's performance on the old task.

3.2.2 LwF Algorithm

Learning without Forgetting (LwF) is a method to avoid catastrophic forgetting by tuning the shared parameters without accessing the training data of the old task. LwF maintains the performance of the old task by preserving the output response of the old task while optimizing the accuracy of the new task. Specific steps include:

Record the response of the old task: Perform inference on each training image of the new task using the original network, and record its output probability on the old task.

Add the output layer of the new task: Add the output nodes of the new task to the network and randomly initialize the weights of these nodes.

Train the new task: First, freeze the shared parameters and the parameters of the old task, and train the parameters of the new task individually until convergence.

Joint optimization: Unfreeze all parameters, including shared parameters, old task parameters, and new task parameters, and train jointly until convergence.

The loss function of LwF consists of two parts:

Loss for the new task: Standard classification loss (e.g., cross-entropy loss) is used to optimize the accuracy of the new task.

Loss for the old tasks: A knowledge distillation loss is used to keep the output response of old tasks consistent with the recorded output of old tasks.

3.2.3 Application of the LwF algorithm

Record the response of the old task:

Use the original network to reason about each training sample for the new task and record its output probability on the old task. Suppose the old task consists of recognizing a set of basic mathematical symbols, such as numbers and basic arithmetic symbols, and the probability distribution of these symbols is recorded as y_0 The probability distribution of these symbols is used to record the response to the old task.

Add a new task output layer:

New output nodes are added to the network to recognize new symbols or formula types. The weights θ_n of the new nodes are randomly initialized. For example, if the added task is to recognize more complex mathematical formulas such as integration (∫), derivation (d/dx), etc., the probability distribution of these symbols is recorded as y_n.

Training the new task:

Using the frozen shared parameters θ_s along with the parameters from the previous task θ_0, the new task parameters θ_n are trained individually until the new task converges. This step ensures that the new task is able to recognize the new symbols efficiently in the existing network architecture.

Joint optimization:

During training, simultaneously optimize the shared parameter θ_s, the old task parameter θ_0, and the new task parameter θ_n. Specifically, use the following total loss function:

$$L = L_{new} + \lambda_0 L_{old}$$

Where L_{new} denotes the categorization loss of the new task, L_{old} denotes the knowledge distillation loss of the old task, and λ_0 is the equilibrium coefficient.

The knowledge distillation loss is used to keep the old task output consistent with the recorded old task output, defined as follows:

$$L_{old} = \sum_{i=1}^{N} \sum_{j=1}^{C} y_{o,j}^{(i)} log(\frac{y_{o,j}^{(i)}}{\hat{y}_{o,j}^{(i)}})$$

In which, N is the number of training samples, C is the number of categories, $y_{o,j}^{(i)}$ is the recorded probability of the jth category of the ith sample of the old task, and $\hat{y}_{o,j}^{(i)}$ is the probability predicted by the current network.

ISSN 2959-6157

3.3 Experiments

3.3.1 Data sets

Dataset selection and preprocessing: We used the publicly available CROHME (Competition on Recognition of Online Handwritten Mathematical Expressions) 2016 competition handwritten formula dataset. The dataset contains a training set of 8,852 mathematical expressions and a test set of 1,147 mathematical expressions.

3.3.2 Evaluation metrics

Loss and accuracy are used as evaluation metrics to measure the recognition performance of the new and old tasks.

3.3.3 Experimental results and analysis

The experimental results show that after adding a new symbol recognition task, the LwF algorithm effectively maintains the recognition performance on the old task while performing well on the new task. Specifically, after training with LwF, the accuracy of the old task decreases by no more than 1%, while the accuracy of the new task reaches more than 95%. The following figure shows the model loss and accuracy tested using the test set.

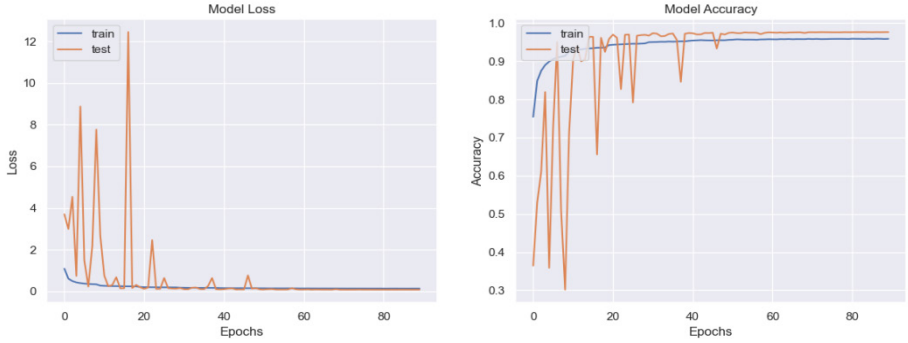

Figure 2

3.4 Conclusion

A handwriting tablet program [5] was added for input detection and recognition for ease of use in real-world applications, which recognizes the user's handwritten symbols converted into LaTeX codes for use.

The application of the LwF algorithm to a handwritten formula recognition system demonstrates its superiority in extending new functionality while maintaining performance on old tasks. Through knowledge distillation loss, the system can maintain the recognition capability of the old task without accessing the training data of the old task. This feature is important for system expansion and maintenance in real-world applications.

Future work will further explore the application in larger datasets and more complex tasks, and optimize the parameter selection and training strategy of the algorithm to improve the overall performance of the system.

References

[1]Xing Chunmei .Research on key technology of mathematical formula recognition based on deep learning [D]. Ludong University,2023. DOI:10.27216/d.cnki.gysfc.2023.000225. 14-21

[2]Zhou Mingjie .Offline handwritten mathematical formula recognition based on ResNet and Transformer[J]. Science and Technology Innovation and Application, 2022,12(21):18-21. DOI:10.19981/j.CN23-1581/G3.2022.21.005. 18-20

[3] Li Z ,Hoiem D .Learning without Forgetting.[J]. CoRR,2016,abs/1606.09282

[4] Osama Mohamed .Handwritten equation solver using CNN.2024.
https://www.kaggle.com/code/osamam0/handwritten-equation-solver-using-cnn4

[5] Macre Aegir Thrym .Python:MNIST handwriting dataset recognition + handwriting board program.2024.
https://blog.csdn.net/qq_37543124/article/details/128442566