Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 4, No. 3, 2023

Real-time Classifier of Multilingual Font Styles based on ResNet, SwordNet, Logistic Regression and Random Forest Algorithms

Yue Wu *

School of AI and Advanced Computing, Xi'an Jiaotong-liverpool University, Jiangsu 215123, China
*** Corresponding author Email:** wuyue20022021@163.com

Abstract: Different languages have different characters. At the same time, each character has a lot of font styles. This makes it difficult for humans to recognize different font styles for different characters. However, being able to detect and identify these font styles quickly and accurately has many important application use cases in different fields. At the same time, a large number of Internet users use web pages to query font styles. Therefore, I choose to make this real-time multilingual font style recognition algorithm. In this paper, I propose an algorithm that recognizes the input text and pictures in real time to judge the language and style of the text. It includes ResNet, SwordNet, logistic regression and random forest algorithms. The whole algorithm also calls pytesseract and Google Tesseract to realize text recognition and text positioning. I used Font Datasets used in "Font and Calligraphy Style Recognition Using Complex Wavelet Transform" for training. At the same time, I also built an image text recognition algorithm and generated various font styles as a data source. Based on this data, we adjusted the parameters and finally achieved an accuracy rate higher than 90%.

Keywords: Font Style Recognition; Real-time Font Identification; Image Text Recognition; Language Identification; ResNet; Random Forest; Logistic Regression; Pytesseract.

1. Dataset

1.1. Real-time Font Style Dataset

This dataset was created by me. The data set is generated in real-time, and its content is very varied. It mainly uses Google's Tesseract and various font packages for generation. I will describe how the entire database is generated and used in real time.

1.1.1. Data Establishment Process

First, the user needs to find a picture of some text as shown in Figure 1. The selected picture should be clear and the text in the picture should be parallel to the picture. Text and pictures in multiple languages can be input and the input picture should be placed in the 'test' folder.

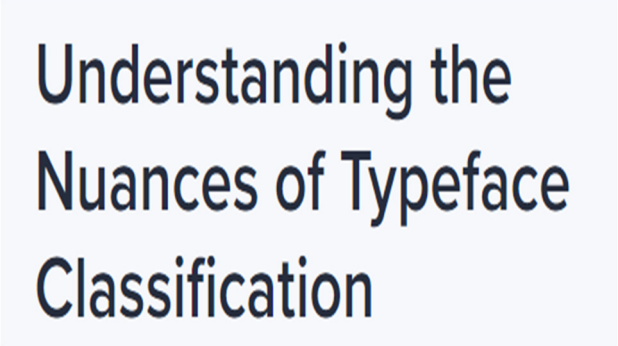


Fig 1. Sentence image

Second, I call pytesseract and google Tesseract to perform recognition on the input image. In this way, I can identify the text content and text position in the image. Based on the position and content of the image, it allows the output of the box-selected image as shown in Figure 2. The output image size is unified to 220*100.

Fig 2. picture of single font

Third, the algorithm calls the font package to generate images with different font styles of the same text as shown in Figure 3. Fonts are all stored in the 'fonts' folder and the number of datasets generated is affected by the number of fonts. The output image size is also unified to 220*100. All generated text images will be stored in the 'train' folder.

Fig 3. pictures of fonts

1.1.2. Data Description and Feature Analysis

The text in the images is black and the background is white. The images are arranged in BGR and they are of the same size. All images are of size 220*100. The data is displayed as a picture and stored in '.png' format. The image name is the label of the image. The format is 'text_font style.png'. All fonts are fully displayed and marked.

Fig 4. pictures of fonts

```
train//wave_AGENCYB.png
(100, 220, 3)
train//wave_ALGER.png
(100, 220, 3)
train//wave_ANTQUAB.png
(100, 220, 3)
train//wave_ANTQUABI.png
(100, 220, 3)
train//wave_ARLRDBD.png
(100, 220, 3)
train//wave_calibri.png
(100, 220, 3)
train//wave_calibrii.png
(100, 220, 3)
train//wave_calibril.png
(100, 220, 3)
```

Fig 5. pictures information **Fig 6.** pictures of single font

Fig 7. Pictures of fonts

Pros:

The entire dataset is generated in real-time, which means that the training data can be modified. So, the user can generate training data based on the text he wants to query.

This allows the user to choose the language they want to use and the font package they own to generate the data.

it also means that we can select the same text as the query text during the model learning process to improve the accuracy and reduce the model learning time.

Cons:

The amount of data generated is relatively small. It takes more time to generate a large amount of data.

The results of training using this dataset are for specific texts and have no great generality.

1.1.3. Data Pre-processing

1.1.3.1 Pictures Imported and Converted to Black and White Images

I use the CV2 package to import the images from the folder and convert the images to black and white. At the same time, the algorithm also reads the image name and enters it into 'words' and 'labels'.

1.1.3.2 Image Feature Transformation

The algorithm calculates the histogram after dividing each channel into 8 groups using the feature transformation function. Finally, it flattens the 8x8x8 multidimensional array. In this way, I get all the image features.

1.2. Font Datasets used in "Font and Calligraphy Style Recognition Using Complex Wavelet Transform"

1.2.1. Data Description

The database is from Font Datasets used in "Font and Calligraphy Style Recognition Using Complex Wavelet Transform" shared by Alican Bozkurt [1]. It includes multiple languages, such as Arabic, Chinese, and Latin-based languages. It has a total of 158 fonts and each font supplies rich font images. All labels are stored in the 'font datasets.csv' file and hold 'ID', 'Alphabet', 'Font', 'Emphasis' and 'Noise Level' as shown in Figure 9.

Arabic Texts	Paragraph images for Arabic, Latin and Chinese fonts add...
Chinese Texts	Paragraph images for Arabic, Latin and Chinese fonts add...
Latin Texts	Paragraph images for Arabic, Latin and Chinese fonts add...
Ottoman Texts/Cropped Ott...	added cropped ottoman dataset

Fig 8. different languages of fonts

ID	Alphabet	Font	Emphasis	Noise Level
0001	Chinese	FangSong	Bold italic	None
0002	Chinese	FangSong	Bold	None
0003	Chinese	FangSong	Italic	None
0004	Chinese	FangSong	Regular	None
0005	Chinese	HeiTi	Bold italic	None
0006	Chinese	HeiTi	Bold	None

Fig 9. fonts information

1.2.2. Feature Analysis

Pros:

This dataset has font data in multiple languages and the data is categorized in detail.

The number of samples of each font is large, which is conducive to model learning and generalization using the model results.

The dataset annotates font details such as italics and bold.

Cons:

Font styles other than those in the dataset are not recognized. At the same time, the text that is not in the dataset may not be recognized.

All text is concentrated on a single image. This requires splitting the text on the image.

1.2.3. Data Pre-processing

1.2.3.1 Character Recognition and Location Recognition

I use the Chinese traditional recognition module of Google Tesseract to recognize the dataset images. The algorithm will locate all the recognized text on the entire image. At the same time, it will also provide the height, width and text content information of the text.

1.2.3.2 Text Image Segmentation

The algorithm uses the location information of the text in

the image to segment the image. It segments the image and sets the size to 64*64. Finally, the algorithm stores all the generated images in a folder.

2. Model

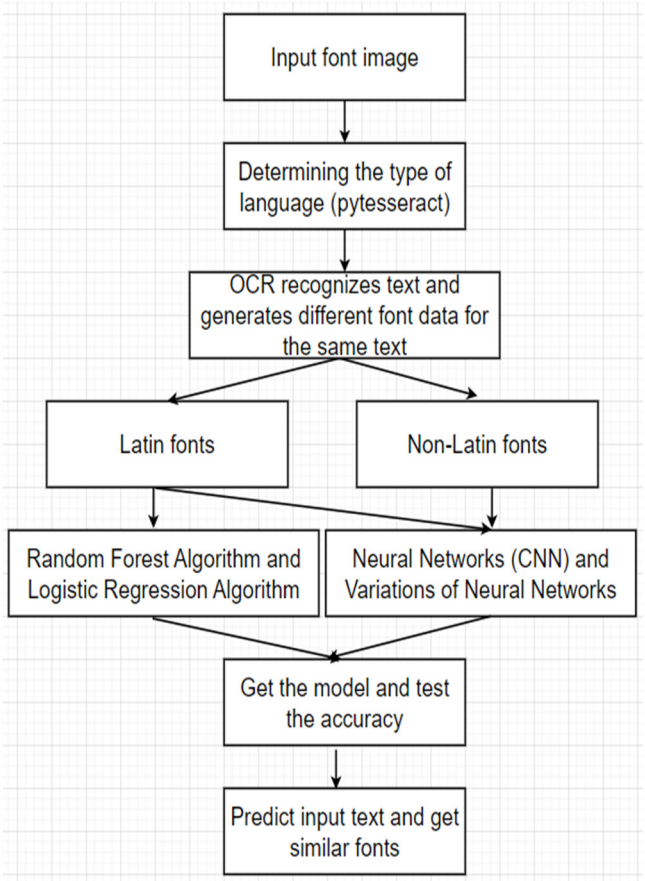


Fig 10. workflow

2.1. Random Forest Algorithm

2.1.1. Workflow

The algorithm flow of random forest:

(1) The algorithm uses bootstrap to extract N samples with replacement from the training set R and Rk is a training subset of R.

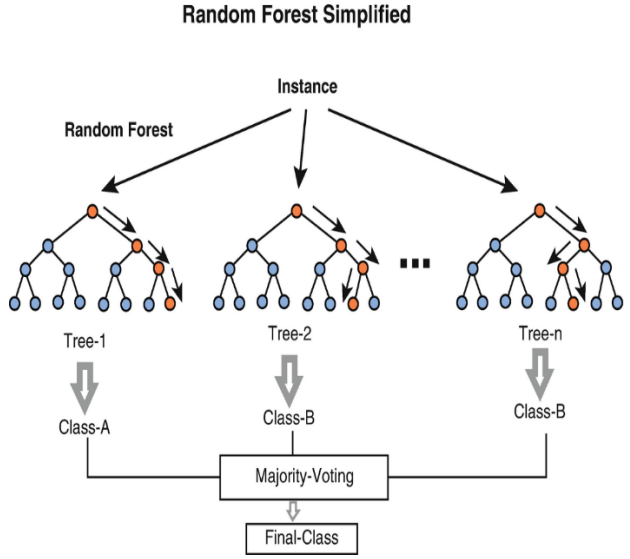


Fig 11. Workflow of Random Forest Algorithm [2]

(2) For the training subset Rk, the algorithm randomly extracts m features from the feature set F without replacement. Among them, m=log2^M. It is used as the basis for the split of each node on the decision tree to generate a complete decision tree Sk from the root node from top to bottom without pruning.

(3) The algorithm repeats steps 1 and 2 n times to get n training subsets R1, R2…Rn. At the same time, it generates decision trees S1, S2…Sn and combines n decision trees to form a random forest.

(4) The algorithm inputs the sample dμ of the test set D into the random forest and lets each decision tree make a decision on dμ. Then, it uses the majority voting method to vote on the decision result and finally decides the classification of dμ.

(5) The algorithm repeats step 4 for λ times until the classification of test set D is completed.

2.1.2. Hardware Usage

The random forest algorithm uses CPU to perform operations. There is no error during the whole running process and the operation speed is fast.

Hardware list:

Device name: LAPTOP-9LIJ8TIU

Processor: 11th Gen Intel(R) Core (TM) i9-11900H @ 2.50GHz 2.50 GHz

On-board RAM: 16.0 GB (15.7 GB available)

System Type: 64-bit operating system, x64-based processor

2.1.3. Theory

An ensemble model like Random Forest is composed of a bunch of decision trees. There are multiple algorithms for decision trees, the most commonly used ones are ID3 and CART. ID3 trees use information gain, while CART trees use the Gini index to decide when to split. Random Forest is designed to reduce overfitting and variance by using a bagging algorithm.

2.1.3.1 ID3 Tree and Information Gain

The whole point of an ID3 tree is to maximize information gain and information gain is a measure of impurity using entropy. Entropy is a measure of (dis)order, which is able to represent the missing flow of information, or the degree of confusion in the data.

Entry Publicity:

$$Entropy(s) = \sum_{i=1}^{c} -p_i \log_2 (p_i)$$

2.1.3.2 CART Tree and Gini Index

The decision tree of the CART algorithm is designed to minimize the Gini index. The Gini index can represent how often randomly selected data points in a dataset may be misclassified.

The Gini index is calculated as:

$$G(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

Among them, pk represents the probability that the sample belongs to the kth category.

2.1.3.3 Bagging Algorithm

The bagging algorithm is mainly used to deal with weak learners. The accuracy of the weak learning algorithm is not high, but the accuracy of the prediction function sequence is improved after repeated use [3]. The random forest uses the bagging algorithm to avoid the defects of high variance and

overfitting.

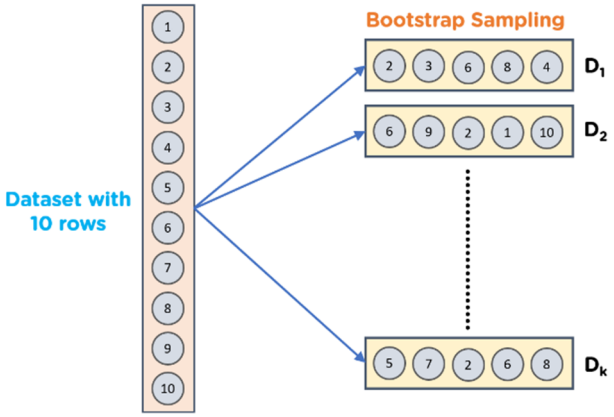


Fig 12. Bootstrapping [3]

2.1.4. Implementation

2.1.4.1 Import All Required Libraries

The random forest algorithm mainly uses the sklearn platform for model building and training. The algorithm also calls pandas for reading files and cv2 for image processing. Some basic data processing packages are also called.

2.1.4.2 Modeling

2.1.4.2.1 Random Forest Model Parameter Optimization

In the whole modeling process, the model parameters are tested to determine the optimal parameters.

n_estimators: the number of decision tree models included in the random forest model.

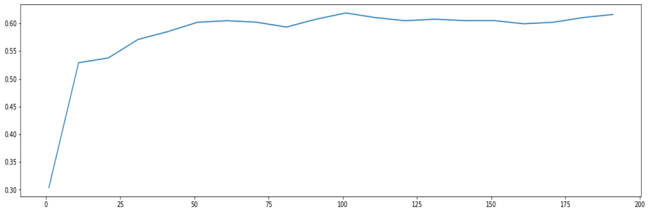


Fig 13. 0-200 estimators' result

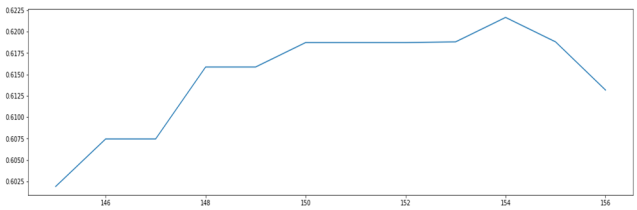


Fig 14. 146-156 estimators' result

max_depth: the maximum depth of the decision tree model.

max_features: The maximum number of features to select when building a decision tree.

min_samples_leaf: the minimum number of samples for leaf nodes.

min_samples_split: the minimum number of samples allowed to split the current node.

criterion: Node splitting basis.

2.1.4.2.2 The Final Model after Parameter Optimization

The final model is modified according to the parameter tuning test and the final model parameters are obtained.

2.1.4.3 Training and Results

The training model is performed using the input x values (font images) and y values (labels). The whole training process takes a short time and generates the training result model.

```
RandomForestClassifier
RandomForestClassifier(max_depth=14, max_features=20, min_samples_split=6,
                       n_estimators=104, random_state=1234)
```

Fig 15. Random forest classifier

The font recognition results of the random forest algorithm for the test image. It runs fast and gives no error messages.

2.2. Logistic Regression

2.2.1. Workflow

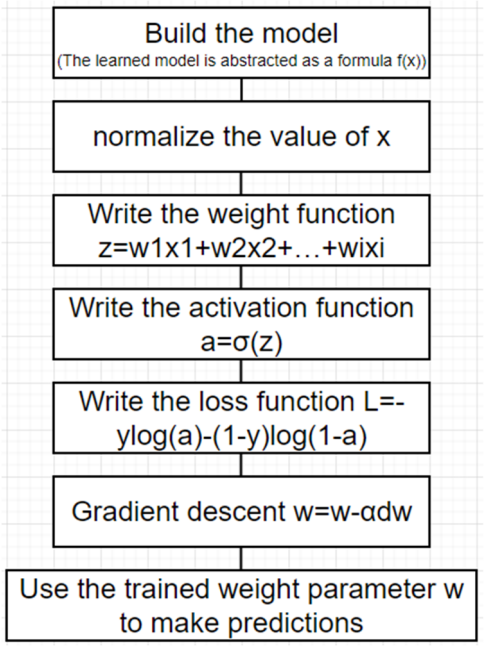


Fig 16. Workflow of Logistic regression

2.2.2. Hardware Usage

The Logistic regression algorithm uses CPU to perform operations. There is no error during the whole running process and the operation speed is fast.

Hardware list:

Device name: LAPTOP-9LIJ8TIU

Processor: 11th Gen Intel(R) Core (TM) i9-11900H @ 2.50GHz 2.50 GHz

On-board RAM: 16.0 GB (15.7 GB available)

System Type: 64-bit operating system, x64-based processor

2.2.3. Theory

The principle of logistic regression is to use the logistic function to map the results of linear regression (-∞, ∞) to (0,1). Logistic regression assumes that the data follow this distribution, and then uses maximum likelihood estimation to estimate parameters.

2.2.3.1 Logistic Distribution

Logistic distribution is a continuous probability distribution, and its distribution function and density function are:

$$F(x) = P(X \le x) = \frac{1}{1 + e^{-(x-\mu)/\gamma}}$$

$$f(x) = F'(X \le x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma(1 + e^{-(x-\mu)/\gamma})^2}$$

Among them, μ represents the position parameter, and y>0 is the shape parameter [4].

2.2.3.2 Logistic Regression

Mathematical expression of linear regression function:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n = \theta^T x$$

where xi is the independent variable, y is the dependent variable, the range of y is (-∞, ∞), θ0 is a constant term, θi(i=1,2,...,n) is the coefficient to be calculated, different weights θi reflects the different contribution of the independent variable to the dependent variable [4].

2.2.3.3 Cost Function

Likelihood function:

$$L(w) = \prod [p(x_i)]^{y_i} [1 - p(x_i)]^{1-y_i}$$

In machine learning we have the concept of a loss function, which measures how wrong a model's predictions are. If we take the average log-likelihood loss over the entire dataset, we can get:

$$J(\theta) = -\frac{1}{m} \ln(L(\theta))$$

2.2.3.4 Gradient Descent and Newton's Method

Stochastic Gradient Descent

Gradient descent is to find the descending direction through the first derivative of J(w) to w and update the parameters in an iterative manner. The update method is:

$$g_i = \frac{\partial J(w)}{\partial w_i} = (p(x_i) - y_i)x_i$$
$$w_i^{k+1} = w_i^k - \alpha g_i$$

where k is the number of iterations. After each parameter update, you can compare the $||J(w^{k+1}) - J(w^k)||$ to stop the iteration when it is less than the threshold or when the maximum number of iterations is reached.

The idea of Newton's method is to perform a second-order Taylor expansion of f(x) in the vicinity of the existing minimum point estimate, and then find the next estimate of the minimum point. Assuming w^k is the current minimum estimate, then there are:

$$\varphi(w) = J(w^k) + J'(w^k)(w - w^k) + \frac{1}{2}J''(w^k)(w - w^k)^2$$

2.2.3.5 Regularization

Objective function:

$$-\ln L(w) = -\sum_i [y_i \ln p(x_i) + (1 - y_i) ln(1 - p(x_i))] + \frac{1}{2\sigma^2} w^T w$$

2.2.4. Implementation

2.2.4.1 Import all Required Libraries

The logistic regression algorithm calls Logistic Regression in sklearn to build the model

2.2.4.2 Modeling

2.2.4.2.1 Parameter Tuning

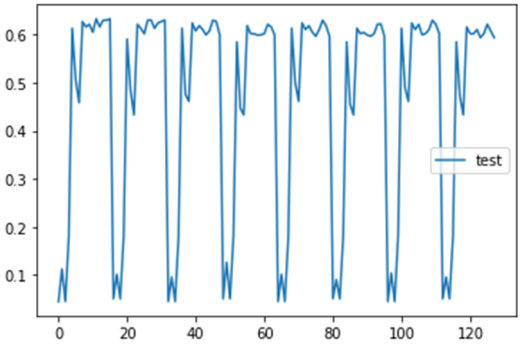


Fig 17. result chart

I use GridSearchCV in sklearn to optimize the parameters of the model. The function of GridSearchCV is to automatically adjust parameters. The user only needs to input the parameters, and the optimized results and parameters can be given.

2.2.4.2.2 Final Model

I build a model based on the results of parameter optimization.

2.2.4.3 Training and Results

The training model is performed using the input x values (font images) and y values (labels). The whole training process takes a short time and generates the training result model. The font recognition results of the random forest algorithm for the test image are shown in the Figure. It runs fast and gives no error messages.

LogisticRegression

```
LogisticRegression(C=0.0001, multi_class='ovr', n_jobs=1, random_state=1234,
solver='newton-cg')
```

Fig 18. Logistic Regression

2.3. ResNet and SwordNet

2.3.1. Workflow

ResNet:

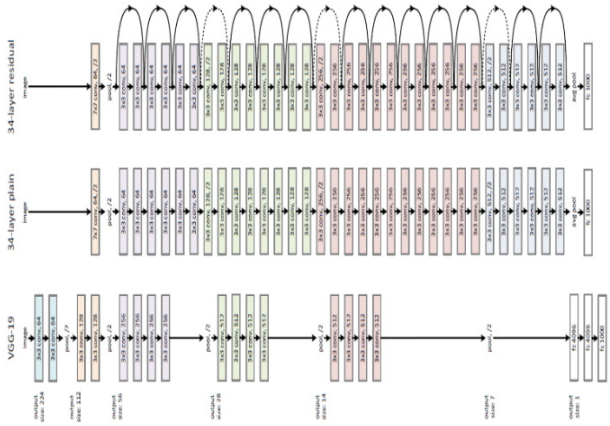


Fig 19. workflow of ResNet [5]

SwordNet:

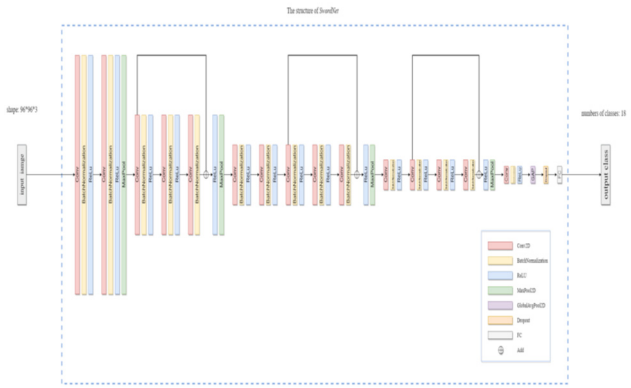


Fig 20. SwordNet structure diagram [6]

2.3.2. Hardware Usage

The neural network algorithm can use GPU to accelerate the model operation process. Therefore, I choose GPU1 (RTX3060 90W) to build and operate the entire neural network model.

Hardware list:

Device name: LAPTOP-9LIJ8TIU

Processor: 11th Gen Intel(R) Core (TM) i9-11900H @ 2.50GHz 2.50 GHz

On-board RAM: 16.0 GB (15.7 GB available)

System Type: 64-bit operating system, x64-based processor

2.3.3. Theory

2.3.3.1 ResNet

2.3.3.1.1 Residual Learning

ResNet has many bypass branches to directly connect the input to the following layers, so that the latter layers can directly learn the residuals. This structure is also called shortcut or skip connections.

2.3.3.1.2 Equivalent Mapping Via Shortcut

When the input and output are the same, the identity shortcut connection is used:

$$y = F(x, W_i) + x$$

When the input and output dimensions are different, a projection shortcut connection and padding with zeros are used:

$$y = F(x, W_i) + W_s x.$$

2.3.3.1.3 Network Structure

ResNet mainly has five main forms: Res18, Res34, Res50, Res101, Res152. Each network includes three main parts: input part, output part and intermediate convolution part

2.3.3.2 SwordNet

SwordNet consists of 15 convolutional layers. The kernel size of each layer is 3*3 and the stride is 2.

2.3.3.2.1 MaxPooling Layer

The maximum subsampling function takes the maximum value of all neurons in the area (max-pooling). The algorithm not only calculates the maximum value in the pool area, but also records the position of the maximum value in the input data.

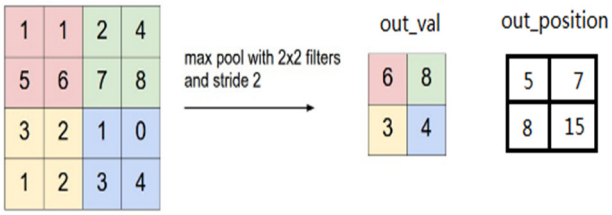


Fig 21. MaxPooling layer [7]

2.3.3.2.2 Skip Connection

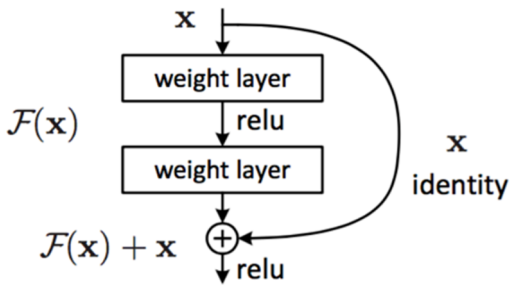


Fig 22. Skip connection [8]

Make a residual block (RB): This block provides shortcuts in the form of an "identity function".

2.3.3.2.3 Global Average Pool

The Network in Network work uses GAP to replace the last fully connected layer, which directly realizes dimensionality reduction, and more importantly, greatly reduces the parameters of the network.

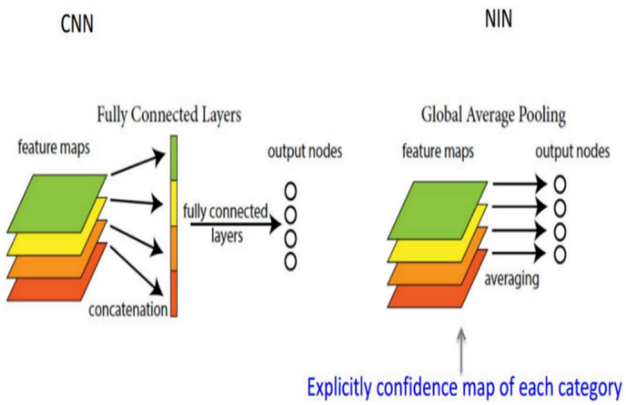


Fig 23. Global average pool [9]

2.3.4. Implementation

2.3.4.1 Import all Required Libraries

ResNet and SwordNet mainly call the ptorch package for model building. It includes the necessary models for neural network construction.

2.3.4.2 Modeling

ResNet:

```
ResNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (layer1): Sequential(
    (0): BasicBlock(
      (features): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (shortcut): Sequential()
    )
    (1): BasicBlock(
      (features): Sequential(
        (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (2): ReLU(inplace=True)
        (3): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
...
  )
  (avg_pool): AvgPool2d(kernel_size=4, stride=4, padding=0)
  (classifer): Linear(in_features=512, out_features=10, bias=True)
)
```

Fig 24. ResNet detail [6]

Different ResNet architectures are unified with one layer of feature extraction and four layers of residuals, and the difference lies in the depth of each layer of residuals. For images, the feature map size changes as follows: (32, 32, 3) -> [Conv2d] -> (32, 32, 64) -> [Res1] -> (32, 32, 64) -> [Res2] -> (16, 16, 128) -> [Res3] -> (8, 8, 256) ->[Res4] -> (4, 4, 512) -> [AvgPool] -> (1, 1, 512) -> [Reshape] -> (512) -> [Linear] -> (10).

SwordNet:

The input image of SwordNet is in 96*96*3 format. The algorithm has a total of 15 convolutional blocks. Each block consists of convolutional layer, BN layer, ReLU activation function layer and MaxPooling layer. SwordNet also includes 3 skip connections. At the end of SwordNet GAP and

SwordNet will output the predicted font style.

Layer Name	Layer	Output Shape
Input	96 × 96 RGB image	(96,96,3)
Conv_Block × 2	3 × 3 Conv2D, 64. BN, ReLU	(96,96,64)
	3 × 3 Conv2D, 128. BN, ReLU	(96,96,128)
MaxPool	MaxPool2D, stride 2	(48,48,128)
Conv_Block × 3	3 × 3 Conv2D, 128. BN, ReLU	(48,48,128)
	3 × 3 Conv2D, 128. BN, ReLU	(48,48,128)
	3 × 3 Conv2D, 128. BN, ReLU	(48,48,128)
MaxPool	MaxPool2D, stride 2	(24,24,128)
Conv_Block × 5	3 × 3 Conv2D, 256. BN, ReLU	(24,24,256)
	3 × 3 Conv2D, 256. BN, ReLU	(24,24,256)
	3 × 3 Conv2D, 256. BN, ReLU	(24,24,256)
	3 × 3 Conv2D, 256. BN, ReLU	(24,24,256)
	3 × 3 Conv2D, 256. BN, ReLU	(24,24,256)
MaxPool	MaxPool2D, stride 2	(12,12,256)
Conv_Block × 4	3 × 3 Conv2D, 512. BN, ReLU	(12,12,512)
	3 × 3 Conv2D, 512. BN, ReLU	(12,12,512)
	3 × 3 Conv2D, 512. BN, ReLU	(12,12,512)
	3 × 3 Conv2D, 512. BN, ReLU	(12,12,512)
MaxPool	MaxPool2D, stride 2	(6,6,512)
Conv_Block × 1	3 × 3 Conv2D, 1024. BN, ReLU	(6,6,1024)
GAP	-	(1024)
Dropout	0.5	(1024)
Output	18 Softmax	18

Fig 25. SwordNet detail [6]

2.3.4.3 Training and Results

Both ResNet and SwordNet have good accuracy after training. And according to my speculation, the accuracy rate of ResNet and SwordNet will reach more than 0.90. SwordNet will have better prediction effect than ResNet.

3. Evaluation

3.1. Random Forest Algorithm

3.1.1. Results

Accuracy:

```
from sklearn.metrics import precision_score


print(precision_score(y, y_pred, average='macro'))
print(precision_score(y, y_pred, average='micro'))
print(precision_score(y, y_pred, average='weighted'))
print(precision_score(y, y_pred, average=None))
```

✓ 0.4s

```
0.9945588235294117
0.9943661971830986
0.9946727423363713
[1.         1.         1.         1.         1.         1.
 1.         1.         1.         1.         1.         1.
 1.         0.94117647 1.         1.         1.         1.
 1.         0.95       ]
```

Recall:

```
from sklearn.metrics import recall_score


print(recall_score(y, y_pred, average='macro'))
print(recall_score(y, y_pred, average='micro'))
print(recall_score(y, y_pred, average='weighted'))
print(recall_score(y, y_pred, average=None))
```

✓ 0.3s

```
0.9944272445820432
0.9943661971830986
0.9943661971830986
[1.         1.         1.         1.         1.         1.
 1.         1.         0.94117647 1.         1.         1.
 1.         1.         1.         1.         1.         0.94736842
 1.         1.        ]
```

Confusion matrix:

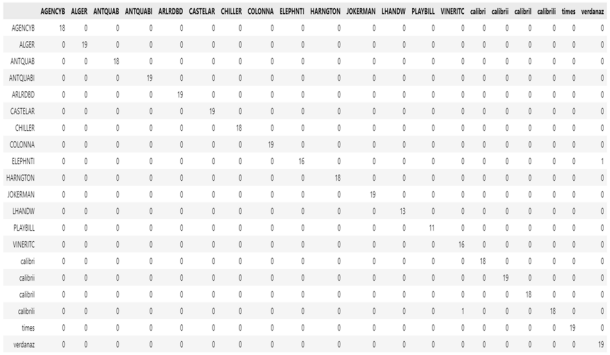


Fig 26. Confusion matrix

3.1.2. Cross Validation

	Label	Precision	Recall	F1	Support
0	AGENCYB	1.000000	1.000000	1.000000	18
1	ALGER	1.000000	1.000000	1.000000	19
2	ANTQUAB	1.000000	1.000000	1.000000	18
3	ANTQUABI	1.000000	1.000000	1.000000	19
4	ARLRDBD	1.000000	1.000000	1.000000	19
5	CASTELAR	1.000000	1.000000	1.000000	19
6	CHILLER	1.000000	1.000000	1.000000	18
7	COLONNA	1.000000	1.000000	1.000000	19
8	ELEPHNTI	1.000000	0.941176	0.969697	17
9	HARNGTON	1.000000	1.000000	1.000000	18
10	JOKERMAN	1.000000	1.000000	1.000000	19
11	LHANDW	1.000000	1.000000	1.000000	13
12	PLAYBILL	1.000000	1.000000	1.000000	11
13	VINERITC	0.941176	1.000000	0.969697	16
14	calibri	1.000000	1.000000	1.000000	18
15	calibrii	1.000000	1.000000	1.000000	19
16	calibril	1.000000	1.000000	1.000000	18
17	calibrili	1.000000	0.947368	0.972973	19
18	times	1.000000	1.000000	1.000000	19
19	verdanaz	0.950000	1.000000	0.974359	19

Fig 27. Cross validation

3.1.3. Analysis

I get the accuracy of random forest algorithm on the training set is 0.994 and the recall is 0.994. This means that

random forest algorithm can get good prediction performance with parameter optimization. We can see that random forest predicts the wrong font style as shown in Figure. From visual observation, we can clearly see that these misidentified fonts have some similarities. It may be an error due to high font similarity. The accuracy rate, recall rate and F1 value of the entire model are all high. This means that the model is successfully established and the parameter optimization effect is good.

Fig 28. The result fonts

3.2. Logistic Regression

3.2.1. Results

Accuracy:

```
from sklearn.metrics import precision_score


print(precision_score(y, y_pred1, average='macro'))
print(precision_score(y, y_pred1, average='micro'))
print(precision_score(y, y_pred1, average='weighted'))
print(precision_score(y, y_pred1, average=None))
```

✓ 0.4s

```
0.716430853615283
0.704225352112676
0.7131374223715334
[1.         0.83333333 0.61904762 0.42857143 0.88235294 0.7826087
 0.6875     0.95       0.6        0.77777778 0.53333333 0.6
 1.         0.52631579 0.75       0.52941176 0.86666667 0.65217391
 0.5        0.80952381]
```

Recall:

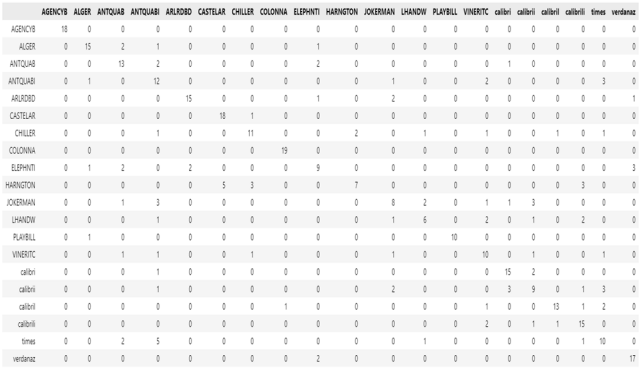
```
from sklearn.metrics import recall_score


print(recall_score(y, y_pred1, average='macro'))
print(recall_score(y, y_pred1, average='micro'))
print(recall_score(y, y_pred1, average='weighted'))
print(recall_score(y, y_pred1, average=None))
```

✓ 0.5s

```
0.7032988403924937
0.704225352112676
0.704225352112676
[1.         0.78947368 0.72222222 0.63157895 0.78947368 0.94736842
 0.61111111 1.         0.52941176 0.38888889 0.42105263 0.46153846
 0.90909091 0.625      0.83333333 0.47368421 0.72222222 0.78947368
 0.52631579 0.89473684]
```

Confusion matrix:



	AGENCYB	ALGER	ANTQUAB	ANTQUABI	ARLRDBD	CASTELAR	CHILLER	COLONNA	ELEPHNTI	HARNGTON	JOKERMAN	LHANDW	PLAYBILL	VINERITC	calibri	calibrii	calibril	calibrili	times	verdanaz
AGENCYB	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ALGER	0	15	2	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ANTQUAB	0	0	13	2	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0
ANTQUABI	0	1	0	12	0	0	0	0	0	0	1	0	0	2	0	0	0	0	3	0
ARLRDBD	0	0	0	0	15	0	0	0	1	0	2	0	0	0	0	0	0	0	0	1
CASTELAR	0	0	0	0	0	18	1	0	0	0	0	0	0	0	0	0	0	0	0	0
CHILLER	0	0	0	1	0	0	11	0	0	2	0	1	0	1	0	0	1	0	1	0
COLONNA	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0
ELEPHNTI	0	1	2	0	2	0	0	0	9	0	0	0	0	0	0	0	0	0	0	3
HARNGTON	0	0	0	0	0	5	3	0	0	7	0	0	0	0	0	0	0	3	0	0
JOKERMAN	0	0	1	3	0	0	0	0	0	0	8	2	0	1	1	3	0	0	0	0
LHANDW	0	0	0	1	0	0	0	0	0	0	1	6	0	2	0	1	0	2	0	0
PLAYBILL	0	1	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0
VINERITC	0	0	1	1	0	0	1	0	0	0	1	0	0	10	0	1	0	0	1	0
calibri	0	0	0	1	0	0	0	0	0	0	0	0	0	0	15	2	0	0	0	0
calibrii	0	0	0	1	0	0	0	0	0	0	2	0	0	0	3	9	0	1	3	0
calibril	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	13	1	2	0
calibrili	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	1	15	0	0
times	0	0	2	5	0	0	0	0	0	0	0	1	0	0	0	0	0	1	10	0
verdanaz	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	17

Fig 29. Confusion matrix

3.2.2. Cross Validation

	Label	Precision	Recall	F1	Support
0	AGENCYB	1.000000	1.000000	1.000000	18
1	ALGER	0.833333	0.789474	0.810811	19
2	ANTQUAB	0.619048	0.722222	0.666667	18
3	ANTQUABI	0.428571	0.631579	0.510638	19
4	ARLRDBD	0.882353	0.789474	0.833333	19
5	CASTELAR	0.782609	0.947368	0.857143	19
6	CHILLER	0.687500	0.611111	0.647059	18
7	COLONNA	0.950000	1.000000	0.974359	19
8	ELEPHNTI	0.600000	0.529412	0.562500	17
9	HARNGTON	0.777778	0.388889	0.518519	18
10	JOKERMAN	0.533333	0.421053	0.470588	19
11	LHANDW	0.600000	0.461538	0.521739	13
12	PLAYBILL	1.000000	0.909091	0.952381	11
13	VINERITC	0.526316	0.625000	0.571429	16
14	calibri	0.750000	0.833333	0.789474	18
15	calibrii	0.529412	0.473684	0.500000	19
16	calibril	0.866667	0.722222	0.787879	18
17	calibrili	0.652174	0.789474	0.714286	19
18	times	0.500000	0.526316	0.512821	19
19	verdanaz	0.809524	0.894737	0.850000	19

Fig 30. Cross validation

3.2.3. Analysis

The final prediction result of the logistic regression model has an accuracy of 0.71 and a recall of 0.70. the prediction accuracy of this model exceeds 0.6, which means that the entire model can make accurate predictions but the final result may still have errors. According to the results of Cross validation and Confusion matrix I can conclude that this model has the lowest prediction accuracy for 'ANTQUABI' font it is only 0.428. At the same time, this model has the lowest recall rate for 'HARNGTON' font which is only 0.388. This means that the model's learning of some font images is still incomplete and the prediction results may be reduced due to the high similarity of the overall image set.

3.3. ResNet and SwordNet

3.3.1. Results

According to the Figure 32, it can be concluded that the prediction results of ResNet and SwordNet are higher than

other models. At the same time, SwordNet has better prediction accuracy than ResNet.

Model	Total_Params	Test_Accuracy	Ref.
EfficientNet	4,072,629	0.8989	[36]
ShuffleNet	1,288,234	0.9071	[37]
AlexNet	6,229,714	0.9097	[34]
GoogleNet	5,992,002	0.9392	[35]
Vgg16Net	28,387,154	0.9457	[20]
ResNet	23,598,034	0.9491	[29]
SwordNet	16,186,322	**0.9903**	**Ours**

Fig 31. Accuracy results [6]

3.3.2. Analysis

The prediction accuracy of ResNet and SwordNet is very high. The model can also be generalized. However, the training of the entire model requires a long time and a large amount of data.

4. Discussion

4.1. Random Forest Algorithm

Pros:

Random forest can produce very high dimensional data. It does not need to reduce dimensionality and it does not need to do feature selection. Therefore, random forest algorithm can also be used for image classification.

Random forest is not easy to overfit. Therefore, users can continuously optimize the parameters of random forest.

If a large part of the features is missing, the random forest can still maintain the accuracy. At the same time, the random forest algorithm can balance the error for the data imbalance.

Random Forest is simple and easy to use. It doesn't require building a model to run and it trains well.

Cons:

Random forest will overfit on some noisy classification or regression problems. Therefore, when inputting images, it is necessary to preprocess the images and analyze the input data.

For datasets with a large amount of data, the random forest algorithm takes a long time for model learning and may not output results for datasets larger than expected.

4.2. Logistic Regression

Pros:

Logistic regression classification is very computationally small and fast. It uses less storage resources.

Logistic regression is computationally inexpensive and easy to implement. It also has a handy observation sample probability score.

Logistic regression is simple to implement and widely used. It has numerous improvements.

Cons:

Logistic regression is prone to underfitting, and the accuracy is generally not very high. It may not perform well in using model predictions.

Logistic regression cannot handle a large number of multi-class features or variables very well. For multi-classification problems, softmax is required.

When using logistic regression, a transformation is required for nonlinear features.

4.3. ResNet and SwordNet

4.3.1. ResNet

Pros:

Even in the most extreme case (where the optimal function is equal to the identity function), the residual function does not produce a "visible" degradation. Since the weights of the residual function part are initialized close to zero, they do not get a large gradient (compared to 1). The residual function will only undergo extremely limited changes when it is updated, and the impact on the identity function is almost negligible.

With the shortcut connection, the signal can be easily propagated throughout the network, thus avoiding "vanishing gradient".

Global average pooling of ResNet can suppress overfitting and make input size more flexible.

ResNet solves the gradient problem to build a deeper neural network model. Therefore, the model can learn more abstract features, and the prediction results will be better.

Cons:

ResNet training is slow and training time is long

With the same number of layers, ResNet training requires more computing power than other models.

ResNet helps us avoid the problem that deep models are difficult to optimize without really solving it. Although ResNet stacks many network layers, the effective depth is small.

4.3.2. SwordNet

Pros:

Global average pooling of SwordNet can suppress overfitting and make input size more flexible.

Its model accuracy is high and higher than that of similar algorithms.

Cons:

SwordNet model training takes a long time. The entire SwordNet model has more layers, which means more time for model training.

4.4. Pros&Cons of the Whole Algorithm

Pros:

The whole algorithm has a fast operation speed and a high prediction accuracy. It can perform real-time text style inference on the text in the input image.

The amount of data required by the algorithm is small. This means that the algorithm is also very adaptable to the lack of data.

The algorithm can not only recognize the font style but also recognize the text content.

Cons:

The algorithm may not have good prediction results for some languages such as Traditional Chinese.

The whole algorithm uses three different methods which may result in more models.

4.5. Problems

4.5.1. Data Processing Issues

Improperly formatted input of data in data processing may cause training results not to be as expected.

4.5.2. Modification and Optimization of Model Parameters

During parameter tuning, each model has many parameters. Optimizing parameters requires code to traverse the possible situations in the parametric model. This will consume a lot of

time, and some parameters have less influence on model optimization.

5. Conclusion

I built this real-time font style recognition algorithm based on ResNet, SwordNet, logistic regression and random forest. Users can input multiple images of any size and get a final font style prediction result. In the end, the accuracy of the prediction results of ResNet and SwordNet is above 0.93. The accuracy of the prediction results of logistic regression is 0.71 and the prediction result of random forestd is 0.99. Finally, the accuracy of the prediction results of the entire model is above 0.90.

However, this font style recognizer is not finalized. I hope to build a model that can classify all font styles. At the same time, although ResNet and SwordNet have completed the deep neural network, their learning ability is not high. I hope that the entire neural network can be further optimized to find a neural network model that can solve the gradient problem. For the field of font style recognition, I hope that scientists It is possible to continue to develop font style recognition technology and continue to study font recognition models. In this way, people can recognize ancient texts and have an impact on future text development.

References

[1] A. Bozkurt, (2013, Nov. 19). Font Datasets used in "Font and Calligraphy Style Recognition Using Complex Wavelet Transform" [Online]. Available: https:// github. com/ alicanb/ fonts.

[2] ACRON (2018) Australian Cybercrime Online Reporting Network (ACORN) [Online]. Available: http://www. acorn. gov. au/learn-about-cybercrime.

[3] A. Biswal, (2021, Sep. 14). Bagging in Machine Learning: Step to Perform And Its Advantages [Online]. Available: https:// www.simplilearn.com/tutorials/machine-learning-tutorial/ bagging-in-machine-learning.

[4] A. Ze, (2021, Jan. 20). [Machine Learning] Logistic Regression (very detailed) [Online]. Available: https://zhuanlan. zhihu. com/ p/74874291.

[5] S.H. Tsang, (2018, Sep. 16). Review: ResNet --Winner of ILSVRC 2015 (Image Classification, Localization, Detection) [Online]. Available: https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization -detection-e39402bfa5d8.

[6] X. Li, et.al, “SwordNet: Chinese Character Font Style Recognition Network” Tianjin, 2016, pp.1-11.

[7] Wangcy, (2016, Dec. 14). Maxpool Layer [Online]. Available: https: // www.jianshu.com/p/6928203bf75b.

[8] L. Han, et.al, “A New Method of Mixed Gas Identification Based on a Convolutional Neural Network for Time Series Classification” Beijing, 2019, pp.9-9.

[9] yalesaleng, (2018, Jul. 13). Global average pooling (GAP) [Online]. Available: https://www.jianshu.com/p/04f7771f4da2.