2019 International Conference on Document Analysis and Recognition (ICDAR)

Oracle Character Recognition by Nearest Neighbor Classification with Deep Metric Learning

Yi-Kang Zhang[1,2], Heng Zhang[1] Yong-Ge Liu[4,5], Qing Yang[1], Cheng-Lin Liu[1,2,3]
[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences,
95 Zhongguancun East Road, Beijing 100190, P.R. China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, P.R. China
[3]CAS Center for Excellence of Brain Science and Intelligence Technology, Beijing, P.R. China
[4] School of Computer & Information Engineering, Anyang Normal University, Henan, P.R. China
[5] Key Laboratory of Oracle Bone Inscriptions Information Processing, Ministry of Education, Henan, P.R. China
Email: {yikang.zhang, qyang, liucl}@nlpr.ia.ac.cn, heng.zhang@ia.ac.cn, liuyongge@aynu.edu.cn

***Abstract*—Oracle character is one kind of the earliest hieroglyphics, which can be dated back to Shang Dynasty in China. Oracle character recognition is important for modern archaeology, ancient text understanding, and historical chronology, etc. To overcome the limitation and class imbalance of training data in oracle character recognition, we propose a classification method based on deep metric learning. We use a convolutional neural network (CNN) to map character images to an Euclidean space where the distance between different samples can measure their similarities such that classification can be performed by the Nearest Neighbor (NN) rule. Because new categories are still being discovered in reality, our model enables the rejection of unseen categories and the configuration of new categories. To accelerate NN classification, we also propose a prototype pruning method with little loss of accuracy. The proposed method exceeds the state of the art on the public dataset Oracle-20K and outperforms CNN with softmax layer on a new dataset Oracle-AYNU.**

***Keywords*-oracle character recognition; dense convolutional network; deep metric learning; nearest-neighbor classifier**

I. Introduction

At the earliest stages of creating hieroglyphics, oracle characters were inscribed on cattle bones or turtle shells for making divinations about 3000 years ago. Oracle characters recorded various human activities during that period, thereby constituting a foundation for the study of Chinese etymologies as well as providing a glimpse into the history of China.

Different from general character recognition such as handwritten Chinese character recognition [1], [2], [3], oracle character recognition has the characteristics that much fewer training samples are available and the number of training data varies a lot among categories. Only a handful of papers aim to recognize oracle characters automatically: Sheng et al. [4] present a recognition method that represents each oracle character as a non-directed graph, which treats end points and intersection as nodes; Li et al. [5] propose a coding based method for recognizing oracle characters; Guo et al. [6] design a novel hierarchical representation for oracle character that combines Gabor-related low-level features and sparse encoder [7] related mid-level ones. It is complementary to convolutional neural network (CNN) based models[8]. Compared with rule-based recognition methods [4], [5], CNN is of great advantage for its capacity and generalization ability. However oracle character data has two main obstacles in contrast to other famous datasets such as ImageNet [9]: (1) the number of samples varies a lot among categories; (2) the number of samples is quite insufficient with respect to the number of categories. Guo et al. [6] do not encounter these issues because their oracle character dataset contains sufficient data for small number of categories.

Considering the data insufficiency and class imbalance, we propose to recognize oracle characters based on nearest neighbor (NN) rule with metric learning [10]. A CNN model is trained with triplet loss [11] to map oracle character images to an Euclidean space where the distance can measure the similarity. With the learned distance metric, the NN classifier [12] is then used to recognize oracle characters. Experiments show that our method can improve the accuracy significantly compared to CNN with softmax layer on a new dataset Oracle-AYNU. Moreover, similar to [13] which extends upon the NN to an open-set classifier, our model can also reject instances of unseen categories and figure out new categories. This is more practical and valuable because new categories are still being discovered in reality. We also design a prototype pruning method to accelerate the NN search with little loss of accuracy.

The main contributions of this paper are as follows:

- As far as we know, this work leads the first to utilize metric learning and NN classifier to solve the data imbalance and data insufficiency problems in oracle character recognition;
- Our model can reject instances from unseen categories and figure out new categories;
- We propose a prototype pruning method to accelerate the procedure of NN search.

II. Related Works

Deep Learning Based Sketch Recognition. In some sense, oracle character can be regarded as sketch drawn in ancient fashion. Early work on sketch recognition extracts handcrafted features followed by feeding them to a classifier. Yu

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00057

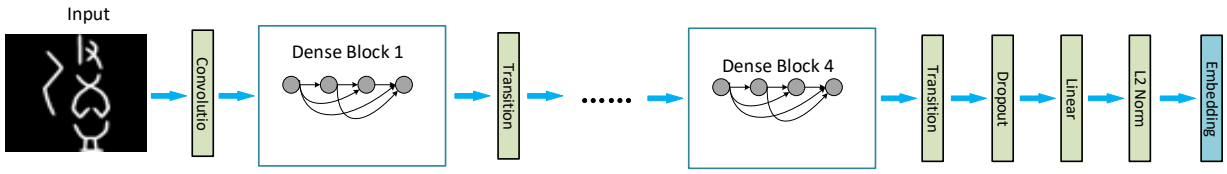

Fig. 1: The network architecture of our task. The input of our network is a $64 \times 64 \times 1$ oracle character image, it is firstly fed into four consecutive densely connected dense blocks, each of which is followed by a transition layer. After the last transition layer is a dropout layer [14], followed by a fully-connected layer whose output size is 128 and a L2 normalization layer.

et al. [15] lead the first to propose a multi-scale and multi-channel deep learning framework, which releases models from human-designed features. It occupies two characteristics: (1) it incorporates a multi-channel generalisation that encodes sequential ordering in the sketching process; (2) it is a multi-scale network and assembles with joint Bayesian fusion. As far as we know, there is only one paper which recognizes oracle characters by CNN [6], but it only conducts experiments on more balanced data and covers small-scale categories.

Deep Metric Learning. Briefly speaking, deep metric learning [16], [17] aims to learn a metric function modeled by a deep nerual network for measuring the similarity between instances. For example, FaceNet [11] proposes a system that directly learns a mapping from face images to a compact Euclidean space. Once this space has been built, recognition, verification and clustering can be easily implemented via various techniques. During training, FaceNet uses triplets of roughly aligned matching/non-matching face patches to get a triplet loss. The triplets are generated via a novel online triplet mining method.

Dense Convolutional Network. Dense Convolutional Network (DenseNet) [18] connects each layer to all its subsequent layers. Plain convolutional networks with L layers have L connections, while DenseNet has $\frac{L(L+1)}{2}$ direct connections. DenseNet has several compelling advantages: it alleviates the vanishing-gradient problem, encourages feature reuse, strengthens feature propagation and substantially reduces the number of parameters. As mentioned above, the oracle character data is insufficient, these advantages can make network easier to train. Therefore, our network is constructed based on DenseNet blocks.

III. Proposed Method

The proposed method first uses a CNN with DenseNet blocks for distance metric learning, and then recognition is performed using the NN rule. In the following subsections, we first introduce the network architecture, the loss and the training procedure. Then, the classification rule is introduced. Last but not least, because the NN search is time-consuming, a prototype pruning method is illustrated to accelerate this procedure.

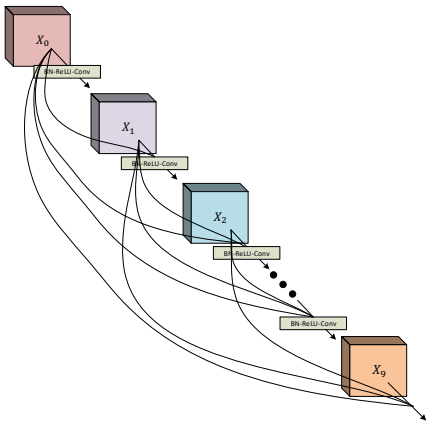

Fig. 2: Our dense block structure. Each of our dense block has 9 layers and a growth rate of 6, which means that all the internal features $x_1, x_2, ..., x_9$ have 6 channels. The output of this dense block is the concatenation of $x_0, x_2, ..., x_9$, which is then fed into the transition layer.

A. Network Architecture

Our net mainly consists four densely connected blocks each of which is followed by a transition layer. The last transition layer is followed by a fully-connected layer. Fig.1 illustrates the layout of the resulting DenseNet schematically.

In each dense block, direct connections from each layer to its subsequent layers are built. Fig.2 illustrates the data stream in one block. Consequently, the ℓ^{th} layer receives the feature-maps of all preceding layers, x_0,...,$x_{\ell-1}$, as input:

$$x_\ell = H_\ell([x_0, x_1, ..., x_{\ell-1}]). \quad (1)$$

$[x_0, x_1, ..., x_{\ell-1}]$ refers to the concatenation of the feature maps produced in layers 0,...,ℓ-1. $H_\ell(\cdot)$ is defined as a composite function of three consecutive operations: batch normalization (BN) [19], followed by a ReLU [20] and a 3×3 convolution (Conv).

The size of feature map in each block is unchanged, it is subsampled by the transition layers which do convolution and pooling. The transition layers used in our experiments consist of a batch normalization layer and a 1×1 convolutional layer followed by a 2×2 average pooling layer.

Finally, we utilize a dropout layer, a fully-connected layer and a L2 normalization layer to map the final feature maps to

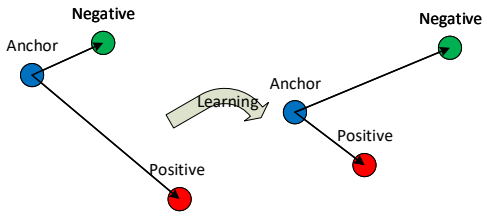

Fig. 3: The **Triplet Loss** minimizes the distance between an anchor and a positive, maximizes the distance between the anchor and a negative. Positive and anchor are of the same category, negative otherwise.

a unit vector. The Euclidean distance of such two unit vectors can measure the similarity of their corresponding input images.

B. Training Algorithm

In our task, an image x can be embedded into a d-dimensional Euclidean space via $f(\cdot)$, which is modeled by CNN. We constrain the embeddings to live on the d-dimensional hypersphere ($||f(x)||_2 = 1$). Then the triplet loss is calculated in this space. Here we want to ensure that an oracle character image x_i^a (anchor) is closer to all other images x_i^p (positive) of the same category than it is to any image x_i^n (negative) of any other category. This is visualized in Fig.3. Thus we want,

$$||f(x_i^a) - f(x_i^p)||_2^2 + \alpha < ||f(x_i^a) - f(x_i^n)||_2^2, \quad \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \tau. \tag{2}$$

Here α is a margin that is enforced between positive and negative pairs. τ is the set of all possible triplets in the training set and has cardinality N, thus $i = 1, 2, ..., N$.

The loss that is being minimized is

$$L = \sum_{i=1}^{N} [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]_+. \tag{3}$$

Here $[\cdot]_+$ means $max(\cdot, 0)$, so if a triplet satisfies equation 2, it will not backward propagate any gradient, otherwise its gradient will make x_i^a closer to x_i^p and futher from x_i^n.

Because a large proportion of the triplets has no contribution to updating parameters, it is crucial to select hard triplets that are active enough to improve the model. The training procedure consists of two stages. In the first stage, both x_i^n and x_i^p are randomly selected. After the convergency, the model obtains the ability to distinguish hard or simple triplets. Thus the training of the second stage focuses on hard triplets, specifically each mini-batch consists of N anchor-positive (x_i^a, x_i^p) pairs and each of them belongs to a unique category. In order to construct the hard triplets (x_i^a, x_i^p, x_i^n), we calculate the distance of $f(x_i^a)$ and $f(x_j^p)$ for all $j \neq i$ and select the nearest one as the x_i^n.

C. Classification Rule

After mapping the oracle character images into the d-dimensional Euclidean space, the classification can be solved by NN rule. The training set $\{x_i | i = 1, 2, ..., n\}$ is embedded offline to get n prototypes. For a test instance $\hat{x}$, we embed it into the Euclidean space $f(\hat{x})$ and find its nearest prototype, i.e., $argmin_{x_i} ||f(\hat{x}) - f(x_i)||_2^2$. Then we classify $\hat{x}$ into the category of its nearest neighbour. Because the NN classifier is an inherent open-set classifier, we also use it to do rejection and open-set classification:

- For a character image of an unseen category, its distance with the nearest prototype tends to be larger. Therefore a threshold can be set to fulfill rejection.
- Once prototypes of new categories are provided, the NN classifier can recognize instances of these categories. Thus our method can take more and more categories into account.

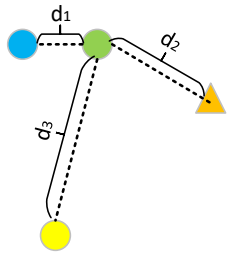

Fig. 4: This figure shows the embeddings in Euclidean space for four instances. The shape(circle or triangle) refers to the category. d_1, d_2, d_3 refers to the distance and $d_3 > d_2 > d_1$. If we remove the blue circle, the green circle will be inaccurately classified into the category of triangle according to NN classifier, so the blue circle is a support prototype and can not be removed.

D. Prototype Pruning

Despite the superior performance, NN classification suffers from the shortage of high consumption of storage and computation due to the large number of prototypes. This consumption can be reduced by means of prototype pruning [21]. The proposed prototype pruning method is similar with [22] but has a better performance for the introducing of outlier prototypes.

Specifically, the prototype pruning method retains two types of prototypes:

- Support prototypes. As the definition of support vector in support vector machine [23], support prototype refers to the prototype that holds the decision boundary of the NN classifier. If a prototype holds the decision boundary, it can be interpreted as removing this prototype will cause some mistakes. Support prototypes are selected via this property. It is obvious that removing an prototype can only result in misclassification for instances belonging to the same category. Therefore when judging support prototypes, we just need to check whether removing it results in misclassification for training instance of the same category. The condition that removing an prototype x will

result in a misclassification for x' can be formulated as:

$$\begin{aligned} &d_3 > d_2 > d_1, \\ &where\ d_1 = ||f(x') - f(x)||_2, \\ &\qquad d_2 = min_{x'''\in\hat{C}}||f(x''') - f(x')||_2. \\ &\qquad d_3 = min_{x''\in C}||f(x'') - f(x')||_2. \end{aligned} \tag{4}$$

Here, C means the set of prototypes which have the same category with x except for x and x', $\hat{C}$ means the set of prototypes in other categories. Once the above condition is satisfied, $f(x)$ is proved a support prototype. We show an example in Fig.4.

- Outlier prototypes. As seen in Fig.5, the appearances for oracle characters of the same category vary a lot. Therefore, an outlier often represents a distinct instance which is of more significance because of its irreplaceability. According to the loss function(3), the inliers should satisfy (2). However the outlier prototypes don't often satisfy (2) because it is distinct with other instances of the same category. We use the same margin α as (3) to detect the outlier prototypes. If x is an outlier prototype, then for any x' of the same category, the following formulation is satisfied:

$$||f(x') - f(x)||_2^2 > d^2 + \alpha,\ d = min_{x''\in\hat{C}}||f(x'') - f(x')||_2. \tag{5}$$

Here $\hat{C}$ means the set of instances of other categories.

IV. Experiments

The proposed method is evaluated on two datasets with different class distributions. We first evaluate the generalized classification performance on two datasets. Then the performance of rejecting instances from unseen categories and open-set classification is evaluated. Furthermore, we show the advantage of the proposed prototype pruning method by some contrast experiments.

A. Datasets

- **Oracle-AYNU.** The oracle character dataset Oracle-AYNU consists of 2583 categories including 39062 instances (2 to 287 instances per category). All of them are binary images and are resized to 64*64. We show some instances in Fig.5. It is utilized to confirm the effectiveness of our method when classifying known categories, discovering and classifying unseen categories and pruning prototypes.
- **Oracle-20K.** Oracle-20K [6] is composed of 20,039 oracle character instances belonging to 261 categories. The largest category consists of 291 instances, whereas the smallest category contain 25 instances. It is utilized to confirm the generalization of our method. Guo et al. [6] only perform experiment of classifying known categories, we perform the same experiment with our method and compare the results.

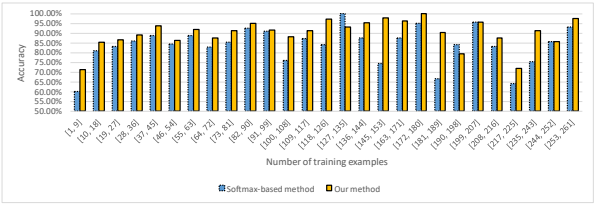

Fig. 6: The relationship between accuracy and number of training examples. The vertical axis means the average accuracy, the horizontal axis means the number of training instances per category.

TABLE I: Experiment results on Oracle-AYNU

Model	Accuracy
DenseNet with softmax	76.23%
Ours	83.37%

B. Classification

We evaluate the classification performance of the proposed method on two datasets and compare with the CNN classifier with softmax output layer. For each category of Oracle-AYNU, 10% of them are testing instances and the others are training instances. We guarantee that there is at least one testing instance for each category. The α in triplet loss is 0.2 in accordance with [11]. In order to effectively contrast our method and softmax layer based method, we adopt the same hyper-parameters, optimization algorithm [24] and network architecture (except for the last layer, one is softmax and the other is L2 normalization).

Experiment results on Oracle-AYNU are shown in TABLE I. Fig.6 reflects the relationship between accuracy and number of training examples. The proposed method outperforms the softmax layer based method in most cases. Taking classes with fewer than ten training instances as an example, our method achieves an accuracy of 71.23% on average, while the softmax layer based method only 60.07%.

The proposed method is also evaluated on the Oracle-20K dataset [6] to verify the generalization. Similar to the setup of [6], two third instances of each category are used for training, the others are used for testing. Experiment results on Oracle-20K are shown in TABLE II. Without external data, [6] achieves an accuracy of 89.2% which exceeds the state of the art at that time. Our method produces a large improvement by an accuracy of 92.43%. In addition, from the table we can see that the accuracy of DenseNet with softmax layer is 92.81%, a little higher than ours in Oracle-20K. It is reasonable because Oracle-20K sidesteps the obstacles of Oracle-AYNU.

C. Rejection

As described above, our model can be used to reject instances from unseen categories based on their distances with the nearest neighbors. We take 258 categories from the 2583 categories in Oracle-AYNU as unseen categories, containing 3502 instances. 10% instances of each known category are used for testing, the others are used for training.

Fig.7 shows the distribution of distance with the nearest neighbour for known and unseen categories. The distribution

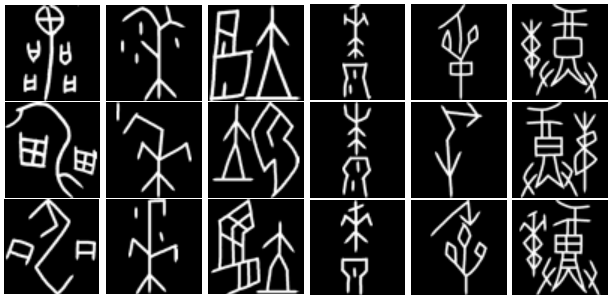

Fig. 5: Some oracle character images from Oracle-AYNU, and every 3 images in the same column come from the same category.

TABLE III: Comparison between our prototype pruning method and other methods

Pruning Method	Candidate Prototypes	Percentage Pruned	Accuracy
Support prototypes (RNN [22])	6575	19.52%	73.67%
Random selected	6575	19.52%	77.83%
Random selected	7450	22.12%	79.03%
Random selected + Outlier prototypes	7450	22.12%	78.80%
Support prototypes + Outlier prototypes(Ours)	7450	22.12%	81.17%
No pruning	33675	100%	83.37%

TABLE II: Experiment results on Oracle-20K

Model	Accuracy
Hierarchical representations based recognition[6]	89.2%
DenseNet with softmax	92.81%
Ours	92.43%

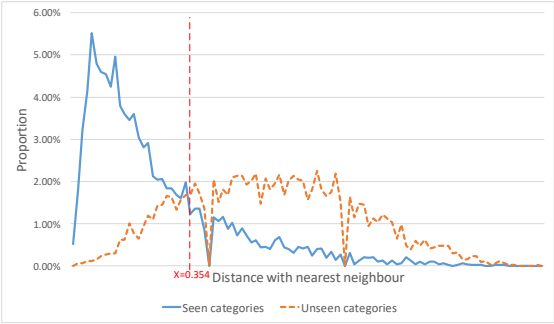

Fig. 7: The distribution of distance with the nearest neighbour for known and unseen categories. The vertical axis means the proportion, the horizontal axis means the distance. The red dashed line means the threshold that we use to reject instances from unseen categories.

for known categories is counted on testing set. A large distinction can be seen from the figure, compared with the known categories, the unseen categoies mainly distributed on larger distance. A threshold should be set for rejection. We denote the accuracy for correctly rejecting unseen categories as $accuracy_{nn}$, correctly accepting known categories as $accuracy_{pp}$. The threshold is selected by maximizing the average of $accuracy_{nn}$ and $accuracy_{pp}$. In this way, we get a threshold of 0.354. Based on this threshold, 80.76% examples of unseen categories can be accurately rejected and 76.68% examples of known categories are accurately accepted.

D. Open-Set Classification

The traditional softmax layer based classifier can not be easily adapted to new categories, however the NN classifier is an inherent open-set classifier. Our method can adapt to new categories as long as the corresponding instances are provided.

The partition of Oracle-AYNU is same as section C. We have 2325 known categories and 258 unseen categories. The known categories include 4886 tesing instances and 34176 training instances. The unseen categories include 3502 instances. We train our model on the training set of known categories and gradually add new categories into our testing set. When we add a new category into the testing set, every instance will be configured as an prototype. When classifying an instance with NN, we need to leave out its corresponding prototype if it comes from unseen categories. The accuracy when gradually adding new categories are shown in Fig.8. The accuracy doesn‘t drop as the new categories becoming more.

E. Prototype Pruning

Under the NN rule, the consumption of storage and computation is proportional to the number of prototypes. Therefore we prune the prototypes except for support prototypes and outlier prototypes. The partition of Oracle-AYNU is same as subsection A. The α used for detecting the outlier prototypes in (5) is also set as 0.2 which is consistent with (3).

Among the 33675 prototypes, there are 6575 support prototypes and 957 outlier prototypes, 82 prototypes belong to both of them. Some results are shown in TABLE III.

From the table we can see that, although the union set of support prototypes and outlier prototypes only consists of 22.12% prototypes, it can hold a relatively high accuracy.

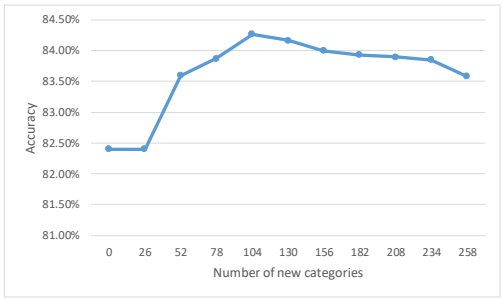

Fig. 8: The accuracy as more and more new categories are taken into account. The vertical axis means the accuracy, the horizontal axis means the number of new categories that are taken into account.

To further claim the effectiveness, we figure out some other experiments. Actually, merely using support prototypes is exactly the strategy adopted in [22]. However, its accuracy is even lower than using the same amount of randomly selected prototypes. Thus, there is a doubt whether support prototypes are valid. However, when combined with outlier prototypes, randomly selected samples will fall into a worse result compared with support prototypes, 78.80% versus 81.17%. Because the union set of support prototypes and outlier prototypes contains 7450 prototypes, the performance of 7450 randomly selected prototypes is also evaluated for comparison. Results show that the accuracy of ours is still higher.

V. Conclusion

In this paper, we propose a method for oracle character recognition based on NN classification with deep metric learning. In our method, a CNN maps oracle character images to a compact Euclidean space for measuring the similarity. We also propose a prototype pruning method for accelerating the NN search procedure in classification. Experimental results on two datasets show that the proposed method yields higher recognition accuracy than conventional CNN classifier, and in addition, it can reject instances of unseen categories and recognize new categories if prototypes are provided. In the future work, we plan to apply the proposed method to real ancient oracle characters collected from oracle bones, and the metric learning can be improved by selecting better triplets.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) Grants 61721004, 61573355, 61836014, the Beijing Science and Technology Program Grant Z181100008918010, and the Open Program of MOE Key Laboratory of Oracle Information Processing of Anyang Normal University.

References

[1] Z. Zhong, X. Y. Zhang, F. Yin, and C. L. Liu, "Handwritten chinese character recognition with spatial transformer and deep residual networks," in *International Conference on Pattern Recognition*, 2016, pp. 3440–3445.

[2] X. Y. Zhang, Y. Bengio, and C. L. Liu, "Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark," *Pattern Recognition*, vol. 61, no. 61, pp. 348–360, 2017.

[3] Y. P. Zhang, L. Shan, N. Shuai, W. J. Liu, and S. Y. Peng, "Robust offline handwritten character recognition through exploring writer-independent features under the guidance of printed data," *Pattern Recognition Letters*, vol. 106, pp. 20–26, 2018.

[4] Q. S. Li, Y. X. Yang, and A. M. Wang, "Recognition of inscriptions on bones or tortoise shells based on graph isomorphism," *Computer Applications in Engineering Education*, vol. 47, no. 8, pp. 112–114, 2011.

[5] F. Li and P.-Y. Woo, "Coding principle and method for automatic recognition of jia gu wen characters," *International Journal of Human-Computer Studies*, vol. 53, no. 2, pp. 289–299, 2000.

[6] J. Guo, C. H. Wang, E. Roman-Rangel, H. Chao, and Y. Rui, "Building hierarchical representations for oracle character and sketch recognition," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 104–118, 2016.

[7] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and U. Montreal, "Greedy layer-wise training of deep networks," *Advances in Neural Information Processing Systems*, vol. 19, pp. 153–160, 2007.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[9] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and F. F. Li, "Imagenet: a large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[10] P. Moutafis, M. Leng, and I. A. Kakadiaris, "An overview and empirical comparison of distance metric learning methods," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 612–625, 2017.

[11] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *IEEE Conference on Computer Vision and Pattern Recognition*, 6 2015, pp. 815–823.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[13] P. R. M. Júnior, R. M. de Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. Penatti, R. d. S. Torres, and A. Rocha, "Nearest neighbors distance ratio open-set classifier," *Machine Learning*, vol. 106, no. 3, pp. 359–386, 2017.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[15] Q. Yu, Y. X. Yang, F. Liu, Y. Z. Song, T. Xiang, and T. M. Hospedales, "Sketch-a-net: A deep neural network that beats humans," *International Journal of Computer Vision*, vol. 122, no. 3, pp. 411–425, 2017.

[16] L. Yang, R. Jin, R. Sukthankar, and Y. Liu, "An efficient algorithm for local distance metric learning," in *National Conference on Artificial Intelligence*, 2006, pp. 543–548.

[17] L. Yang, R. Jin, and R. Sukthankar, "Bayesian active distance metric learning," in *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, 2007, pp. 442–449.

[18] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 7 2017, pp. 2261–2269.

[19] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. 37, pp. 448–456, 2015.

[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

[21] J. A. Olvera-Lpez, J. A. Carrasco-Ochoa, J. F. Martnez-Trinidad, and J. Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, no. 2, pp. 133–143, 2010.

[22] G. W. Gates, "The reduced nearest neighbor rule," *IEEE Trans.inf.theory*, vol. 18, no. 3, pp. 431–433, 1972.

[23] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, 2014.