Handwritten Chinese Character Recognition with Neural Networks

Sirui Chen, Naihe Xiao, Ming Yeung Alfred Meng
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
`{sirui.chen, naihe.xiao, ming.meng}@mail.utoronto.ca`

Abstract

In this study, we explore the effectiveness of transfer learning using two pre-trained models, EfficientNet and InceptionV3+Gated Recurrent Unit (GRU), for Handwritten Chinese Character Recognition (HCCR) on the Chinese MNIST dataset. We fine-tune these models on various training datasets and evaluate their performance. Our results demonstrate that augmenting the size and diversity of the training dataset can enhance model performance. These findings suggest that pre-trained models are a promising approach for HCCR and can be further improved by utilizing larger and more diverse training datasets. The coding implementations can be found on github here and here.

1 Introduction

Handwritten character recognition (HCCR) has been a challenging task in the field of computer vision and machine learning. With the increasing use of digital devices, the need for efficient and accurate recognition of handwritten characters has become more critical than ever. In recent years, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), such as Gated Recurrent Units (GRU), have shown promising results in various tasks for image recognition. In this project, we explore the effectiveness of pre-trained EfficientNet and GRU augmented InceptionV3 models for HCCR using the Chinese MNIST dataset and evaluate their performance by fine-tuning the model on different subsets of the training data.

2 Related Work

The rapid growth of HCCR has been driven by the development of deep learning [8, 9], particularly the usage of convolutional neural networks (convNet). The first reported application of CNN for HCCR was MCDNN, a multi-column deep neural network proposed by Cire¸ssan et al [7]. After this, several adapted convNets have emerged with increasingly high accuracy, including sparse convNet [3] and trained relaxation convolutional neural network[6]. Recent benchmarks in HCCR have even surpassed human performance, with achievements of over 96%, such as HCCR-Ensemble-GoogLeNet-10 [12], CNN-Voting-5 [4], Ensemble-DCNN-Similarity ranking [5], and others.

In 2016, a novel model architecture was introduced [11], which combined the traditional directMap approach with a deep convolutional neural network. This model also included a new adaptation layer to mitigate the mismatch between the training and test data on a specific source layer. Furthermore, character-based approaches [13, 14, 15] treat each character as a distinct category and capture

This document was written with the help of ChatGPT (Mar 23).

global features such as contours to make character predictions. These emerging approaches have demonstrated improved effectiveness and the ability to capture complex characters in deep learning models.

However, the challenges in HCCR remain significant due to the large scale vocabulary, diversity in handwriting styles, and similar and confusable glyphs[11]. Since it is expensive to construct such a large-scale model for recognition, some pre-trianed model can help reduce data requirements and speed up the training process for specific tasks.

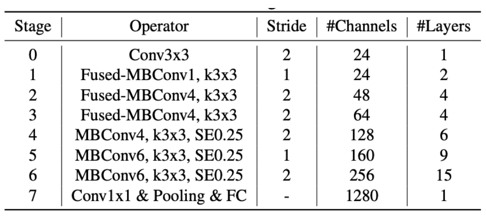
3 Model Architecture

3.1 Dataset

For training and evaluating our models, we utilized the Chinese MNIST dataset, which was created by Gabriel Preda and is available on Kaggle [1]. This dataset contains 15,000 images of handwritten Chinese characters, collected from one hundred Chinese nationals. Each participant wrote ten samples for each of the 15 characters using a standard black ink pen on white A4 paper. The resulting images were then scanned at a resolution of 300 x 300 pixels.
To explore the association between dataset size and model performance, as well as to address the computational challenges posed by the complex InceptionV3 model architecture and GRU layer, we randomly generated three subsets of the full dataset for training in each experiment. These subsets contained the same number of images in each class. Training the InceptionV3 model with the full dataset took two hours for a single epoch, whereas the EfficientNet model trained in just 15 seconds. We will use these subsets for both models in our experiments.

3.2 EfficientNet

EfficientNet is a family of deep convolutional neural networks that was introduced in 2019 by Tan and colleagues [16]. The model was inspired by Mnas-Net and specifically designed to achieve high accuracy while requiring fewer computational resources than other state-of-the-art models. We choose EfficientNetV2 as our target pre-trained model and the basic archecture is shown in Figure 1.



Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	256	15
7	Conv1x1 & Pooling & FC	-	1280	1

Figure 1: EfficientNet Model Arichitecture

From Efficientnetv2: Smaller models and faster training" by Tan, M., & Le, Q, 2021 July. In International conference on machine learning (pp. 10096-10106). PMLR.

The key idea behind EfficientNet family is to scale the model's width, depth, and resolution in a principled way, rather than simply increasing these parameters independently. Specifically, the scaling coefficients of the compound scaling method are given by:

$$depth : d = \alpha^{\phi}$$
$$width : w = \beta^{\phi}$$
$$resolution : r = \gamma^{\phi} \;\; s.t. \;\; \alpha\beta^2\gamma^2 \approx 2 \;\; and \;\; \alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

ϕ is a coefficient specified by users that controls how many more resources are available for model scaling, while α, β, γ are constants that specifies how to allocate resources for depth, width and resolution respectively. The scaling coefficients are chosen through a grid search to optimize the performance of the network while minimizing the computational cost.

3.3 InceptionV3 with GRU layer

Our second model uses the pre-trained InceptionV3 model, a deep neural network architecture developed by Google, as the backbone[10]. The main building block of the model is the Inception module, which contains a total of 159 layers; the first 94 layers are feature extraction layers, followed by a 2-layer fully connected network, and the remaining 63 layers are the classification layers. The output sequence $X = x_1, x_2, ..., x_n$ is then passed into a GRU layer, which computes the output sequence $h_1, h_2, ..., h_t$ as follows:

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\
\tilde{h_t} &= tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h_t}
\end{aligned}
$$

where σ is the sigmoid activation function, $\odot$ is the element-wise multiplication, W_z, W_r, W_h, U_z, U_r, U_h, b_z, b_r, b_h are learnable weights and biases of the GRU layer.
In the above equations, z_t is the update gat, r_t is the reset gate, $\tilde{h_t}$ is a candidate hidden state computed based on the current input x_t and the reset gate r_t, and h_t is the updated hidden state.

4 Experiments

4.1 EfficientNet

Our first experiment implements the fine-tuned EfficientNet on HCCR with three different data sizes, each contains 70, 140 and 210 images per class for a total of 15 classes. We chose the EfficientNetV2S architecture from Tensorflow that was pre-trained on ImageNet, and has 513 layers and 20,331,360 trainable parameters (without the final classifier layer).

To adapt the model to our specific task of handwritten Chinese character recognition, we replaced its original classifier with one that outputs 15 logits. The model then consists of the sequence of layers: MLP, dropout, MLP. We first trained the new classifier, then froze the first 100 layers and fine-tuned the entire model, for 10 epochs each. Please see figure 5 for the validation loss and accuracy. The final validation accuracy of the model trained on the small, medium and large datasets are 93.14%, 95.05% and 96.67%, respectively. This outcome aligns with our expectation as training on larger data size allows the model to capture more complex structure of the data.

4.2 InceptionV3 and GRU layer

As the InceptionV3 model necessitates the input data to have dimensions of 299×299, we resized the original images accordingly. As in the previous experiment, we generated three datasets: small, medium, and large, comprising 5%, 10%, and 20% of the original dataset, respectively.

We augmented the original model structure with 5 additional layers: a GaussianNoise layer and a dropout layer to add random Gaussian noise and randomness to the input to help prevent overfitting; a batch normalization layer to improve the stability and convergence of the model; and two dense layers to allow the model to capture more complex patterns in the data.

We fine-tuned the number of GPUs, learning rate, and batch size hyper-parameters to evaluate their effect on model performance and identify the best model. Due to time constraints, we conducted fine-tuning on the small dataset. The model's performance variation with the change of hyperparameters is expected: lower learning rate reduces overfitting and larger batch size enables smoother optimization, both boosting the validation accuracy. One noteworthy observation is that models trained with 4 GPUs achieve almost twice the average validation accuracy of those trained with only 1 GPU. This phenomenon may be attributed to faster training times facilitating comprehensive learning or additional regularization effects introduced by more GPUs, preventing overfitting and improving the model's generalization capability.

	train loss	train accuracy	validation loss	validation accuracy
small model	0.1137	0.9650	0.0297	0.9933
median model	0.0436	0.9842	0.1034	0.9833
large model	0.0673	0.9804	0.0350	0.9900

Table 1: Train and validation accuracy for different models

The optimal hyper-parameter configuration was `{GPU = 4, batch_size = 40, learning_rate = 0.0001}`. Using these hyper-parameters, we trained the model on the three datasets and included the resulting outputs above. The validation accuracy does not vary significantly with data size: all models achieve a remarkable validation accuracy.

5 Results and Discussion

5.1 Comparison

Based on the results of our experiments, both the EfficientNet and InceptionV3 with GRU layer models achieved excellent performance in recognizing handwritten Chinese characters, with accuracy consistently in the high 90s. However, the training time of the EfficientNet model was significantly faster compared to the InceptionV3 + GRU model, with a speedup of 10-15 times. This can be attributed to the unique design architecture of EfficientNet, which achieves a good balance between model depth, width, and resolution scaling.

As shown in Figure 1, the EfficientNet model uses fewer parameters and layers, which leads to fewer calculations during training and improved efficiency. Additionally, the efficient convolutional operations and bottleneck layers further reduce the computational cost of the model, resulting in faster training times. Though GRU is a much simpler architecture compared to other types of recurrent neural network, the overall size of InceptionV3 with GRU model is still relatively larger, which increases the computational cost and training time.

Both pre-trained models achieved promising results for HCCR, with the EfficientNet model demonstrating a more efficient approach. In real-world applications, the EfficientNet model may be preferred due to its ability to transfer to larger and more diverse datasets more efficiently.

5.2 Limitations

- Small data size: we have to use only a portion of the full data for training due to time constraint, and we used the small dataset for fine tuning the hyper-parameters(even so it took us more than 2 days to finish the full process). This leads to limited data variability and overfitting, and restricts the model's generalization performance.
- Earlystopping Removal: We discarded the original earlystopping mechanism since our validation accuracies started to be flat. However, early stopping is necessary when training on the full data with more epochs to prevent potential overfitting.
- Missing testing procedures: due to time constraints, we didn't incorporate any testing mechanisms for our models, which increases the risk of overfitting and data leakage.
- Non-representative data: our data only contains 15 out of 50000 Chinese characters hence the model may not be generalizable to some new random Chinese characters.

6 Conclusion

In conclusion, our study successfully applied transfer learning to the task of HCCR using the Chinese MNIST dataset, and obtained promising results. Our findings suggest that the performance of image classification models can be improved by training on a larger and more diverse dataset. Furthermore, our experiments demonstrated the effectiveness of pre-trained models such as EfficientNet and InceptionV3 with GRU for transfer learning in HCCR.

In comparing the two models, we found that EfficientNet outperformed InceptionV3 and GRU in terms of efficiency and optimization. These results highlight the importance of selecting well-designed pre-trained models for transfer learning in HCCR.

Moving forward, further investigation into a wider variety of pre-trained models and larger datasets would be valuable. It would also be worthwhile to test the fine-tuned models on other datasets, such as CASIA-HWDB, to better understand the impact of dataset variations on hyperparameter choices and performance. Additionally, exploring different hyperparameters and architectures, such as freezing trainable parameters and adding Dropout layers, could lead to even better performance for transfer learning in HCCR.

References

[1] Gabriel Preda. (2021, March 28). Chinese MNIST. Kaggle. https://www.kaggle.com/datasets/gpreda/chinese-mnist.

[2] Atsushi. (2023). Chinese MNIST using CNN Tensorflow Tutorial. Kaggle. https://www.kaggle.com/code/atsushi015/chinese-mnist-using-cnn-tensorflow-tutorial.

[3] Graham, B. (2014). Spatially-sparse convolutional neural networks. arXiv preprint arXiv:1409.6070.

[4] Chen, L., Wang, S., Fan, W., Sun, J., & Naoi, S. (2015, November). Beyond human recognition: A CNN-based framework for handwritten character recognition. In 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR) (pp. 695-699). IEEE.

[5] Cheng, C., Zhang, X. Y., Shao, X. H., & Zhou, X. D. (2016, October). Handwritten Chinese character recognition by joint classification and similarity ranking. In 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR) (pp. 507-511). IEEE.

[6] Wu, C., Fan, W., He, Y., Sun, J., & Naoi, S. (2014, September). Handwritten character recognition by alternately trained relaxation convolutional neural network. In 2014 14th International Conference on Frontiers in Handwriting Recognition (pp. 291-296). IEEE.

[7] Cireşan, D., & Meier, U. (2015, July). Multi-column deep neural networks for offline handwritten Chinese character classification. In 2015 international joint conference on neural networks (IJCNN) (pp. 1-6). IEEE.

[8] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. science, 313(5786), 504-507.

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521 (7553), 436-444. Google Scholar Google Scholar Cross Ref Cross Ref, 25.

[10] Phong, N. H., & Ribeiro, B. (2020). Rethinking recurrent neural networks and other improvements for image classification. arXiv preprint arXiv:2007.15161.

[11] Zhang, X. Y., Bengio, Y., & Liu, C. L. (2017). Online and offline handwritten Chinese character recognition: A comprehensive study and new benchmark. Pattern Recognition, 61, 348-360.

[12] Zhong, Z., Jin, L., & Xie, Z. (2015, August). High performance offline handwritten chinese character recognition using googlenet and directional feature maps. In 2015 13th international conference on document analysis and recognition (ICDAR) (pp. 846-850). IEEE.

[13] Yang, X., He, D., Zhou, Z., Kifer, D., & Giles, C. L. (2017, November). Improving offline handwritten Chinese character recognition by iterative refinement. In 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) (Vol. 1, pp. 5-10). IEEE.

[14] Wu, Y. C., Yin, F., & Liu, C. L. (2017). Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. Pattern Recognition, 65, 251-264.

[15] Xue, M., Du, J., Wang, B., Ren, B., & Hu, Y. (2023). Joint Optimization for Attention-based Generation and Recognition of Chinese Characters Using Tree Position Embedding. Pattern Recognition, 109538.

[16] Tan, M., & Le, Q. (2021, July). Efficientnetv2: Smaller models and faster training. In International conference on machine learning (pp. 10096-10106). PMLR.

7 Appendix

7.1 Contributions

7.1.1 Naihe Xiao

- Discussed and determined topics and work splitting.
- Coded data loading and pre-processing for inceptionv3 and GRU model.
- Coded basic model and training process for inceptionv3 and GRU model.
- Fine tuned hyper-parameters and recorded outcomes.
- Wrote sections 3.1, 3.3, 4.2, 5.2 of the report.
- Reviewed and polished the report.

7.1.2 Sirui Chen

- Discussed and determined topics.
- Added utility functions, refactored and cleaned code for inceptionv3 and GRU model.
- Training small, median, and large models.
- Wrote sections abstract, 1, 2, 3.2, 5.1, 6 of the report.
- Reviewed and polished the report.

7.1.3 Ming Yeung Alfred Meng

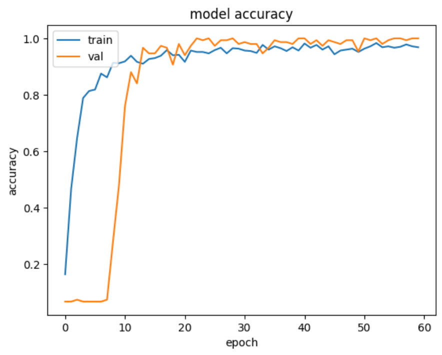
- Implemented EfficientNet to solve this problem.
- Wrote Section 4.1 EfficientNet

7.2 Fine Tuning InceptionV3 hyperparameters

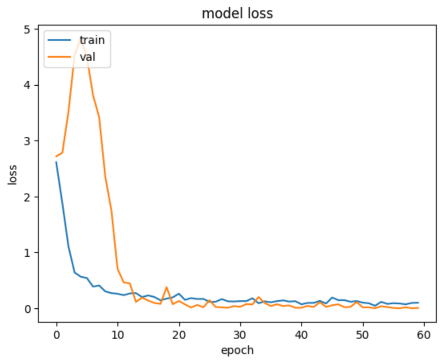
	hyper-parameters			training outcomes			
	GPUs	learning rate	batch size	train loss	train accuracy	validation loss	validation accuracy
model1	1	1e-4	10	0.0757	0.9783	0.5035	0.5035
model2	1	1e-4	20	0.0702	0.9767	0.9767	0.6533
model3	1	1e-4	40	0.1073	0.9650	0.6712	0.7533
model4	1	2e-4	10	0.1043	0.9683	1.2170	0.5667
model5	1	2e-4	20	0.1134	0.9683	1.1102	0.6600
model6	1	2e-4	40	0.1453	0.9550	0.7650	0.7533
model7	1	1e-3	10	0.1616	0.9500	3.1704	0.0667
model8	1	1e-3	20	0.2530	0.9150	4.6345	0.0733
model9	1	1e-3	40	0.2115	0.9333	5.2650	0.1933
model10	4	1e-4	10	0.1271	0.9683	0.0422	0.9800
model11	4	1e-4	20	0.1038	0.9667	0.0368	0.9867
model12	4	1e-4	40	0.1137	0.9650	0.0297	0.9933
model13	4	2e-4	10	0.1223	0.9550	0.1122	0.9600
model14	4	2e-4	20	0.0928	0.9683	0.0480	0.9800
model15	4	2e-4	40	0.1062	0.9683	0.0222	0.9867
model16	4	1e-3	10	0.2502	0.9067	0.2787	0.8867
model17	4	1e-3	20	0.2694	0.9167	0.2177	0.9267
model18	4	1e-3	40	0.3210	0.9033	0.5342	0.8267

Table 2: Accuracy for different sets of hyperparameters

7.3 InceptionV3 + GRU model training plots

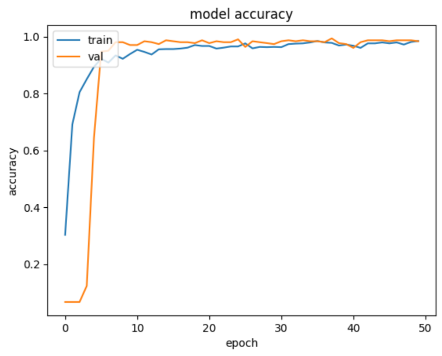


(a) small model accuracy

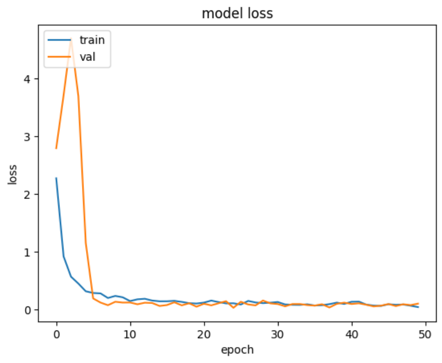


(b) small model loss

Figure 2: Small dataset Training results

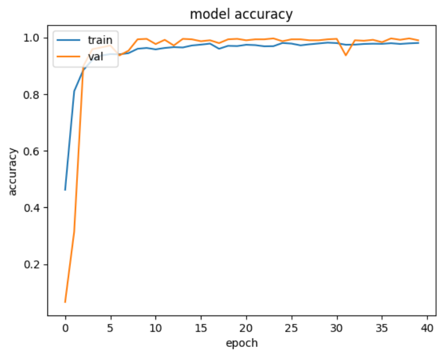


(a) median model accuracy

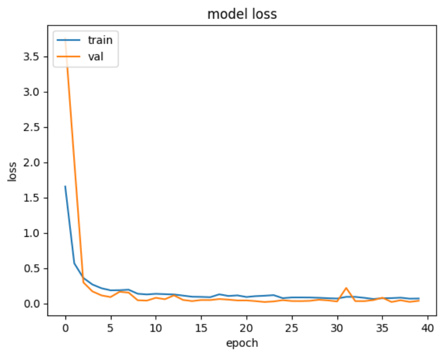


(b) median model loss

Figure 3: Median dataset Training results

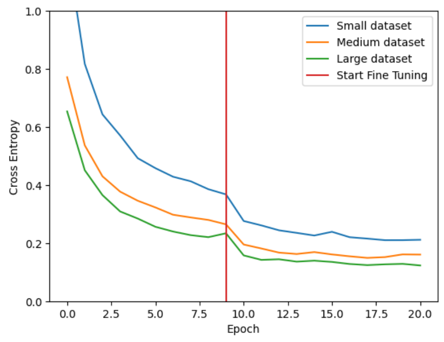


(a) large model accuracy

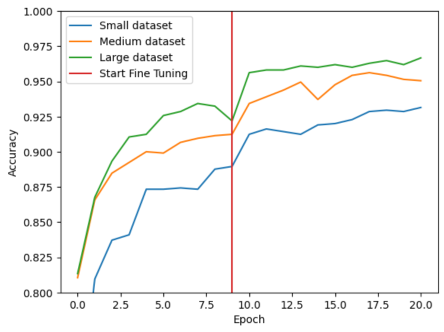


(b) large model loss

Figure 4: Large dataset Training results



(a) Validation Loss



(b) Validation Accuracy

Figure 5: Validation Loss and Accuracy of Pre-Trained EfficientNet Fine-Tuned on Dataset of Various Sizes