# Deep Dive into Recognition of Sino-Nom characters

Chu Ngoc Vuong
VNU University of Engineering and Technology
21020674@vnu.edu.vn
Group 16

Le Viet Viet Linh
VNU University of Engineering and Technology
21020644@vnu.edu.vn
Group 16

Vu Minh Tuan
VNU University of Engineering and Technology
21020664@vnu.edu.vn
Group 16

Pham Duc Trung
VNU University of Engineering and Technology
21021548@vnu.edu.vn
Group 16

***Abstract*—This report details a project focused on regcognition and classifying Sino-Nom characters using image processing and deep learning techniques. The project addresses the complexity of Chinese script, which includes thousands of characters with intricate strokes and varied handwriting styles. We utilized convolutional neural networks (CNNs) to develop a model capable of recognizing these characters. Our approach involved preprocessing a given dataset of Sino-Nom characters, then training multiple convolutional network models to accurately classify them. The model's architecture was optimized through experimentation with different configurations and data augmentation techniques to improve performance.And finally, different techniques and multiple individual models are also combined, known as ensemble learning to create a more robust and accurate prediction model. The results show our approach have achieved high recognition accuracy of 93,6 % with the given problem, demonstrating the potential of deep learning for Sino-Nom characters recognition.**

## I. Introduction

Sino-Nom characters recognition presents a unique and challenging problem in the field of pattern recognition and machine learning due to the complexity and diversity of Sino-Nom script. Unlike alphabetic languages, Sino-Nom based on the Chinese characters are logograms, where each character represents a word or a meaningful part of a word. With thousands of distinct characters, each composed of intricate and often subtle variations in strokes, developing an effective classification system is a formidable task.

This project aims to explore the application of deep learning techniques, specifically convolutional neural networks (CNNs), to classify handwritten Chinese characters. CNNs are well-suited for image recognition tasks due to their ability to automatically learn and extract hierarchical features from raw pixel data. By leveraging CNNs, we aim to both create a model and leverage existing one that can accurately distinguish between different Chinese characters based on their representations.

This project aims to develop a robust system for recognizing handwritten Chinese characters, specifically focusing on the Sino-Nom character set. The primary objectives include preprocessing a diverse dataset containing both high-quality and low-quality images of these characters, designing and training a Convolutional Neural Network (CNN) model capable of classifying them accurately, and evaluating the performance of the trained model

In the following sections, we will discuss the methodology employed in this project, including data preprocessing, data augmentation, model design, and training. We will then present the results obtained from our experiments for the Sino-Nom character character recognition.

## II. Data

### A. Analysis

The given dataset has 2,130 labels, each comprising several images containing the same character but with different strokes. Moreover, many labels (though not all) include low-quality images, which require preprocessing before applying them to the model. The following paragraphs will analyze the distribution of the number of images of each label.

From the chart in Figure 1, it can be concluded that the majority of the labels have fewer than 50 images, and only a small number of them have more than 100 images. This uneven distribution of images across labels may lead to a decrease in the model's accuracy..

Next, through analysis, we discovered that 1,667 labels include low-quality images, while the rest are composed of high-quality images. If we train the model with these low-quality images, the results will undoubtedly be poor. For that reason, we decided to preprocess all the images to enhance their sharpness and turn them into scan-like images.

### B. Preprocessing

After several approach, we choose to preprocess the low-quality images using a technique called thresholding. The input image is converted to grayscale and then processed with the Adaptive Threshold function in the OpenCV libraryThe adaptive method used is ADAPTIVE_THRESH_GAUSSIAN_C. We also tested ADAPTIVE_THRESH_MEAN_C, but the results were not as good as with the former method. Finally, the image is resized to the desired dimensions. The code for this function is illustrated below, where img_path is the local disk path of the image, and width and height are the desired dimensions of the image. The Figure 2 illustrate the

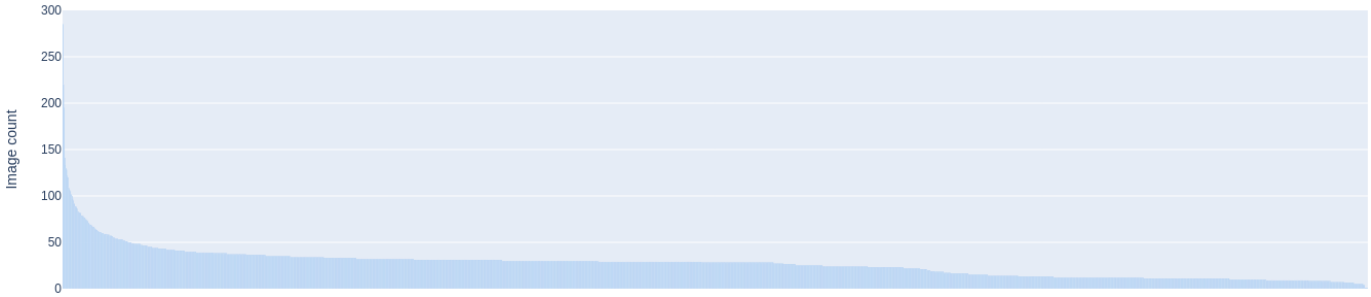
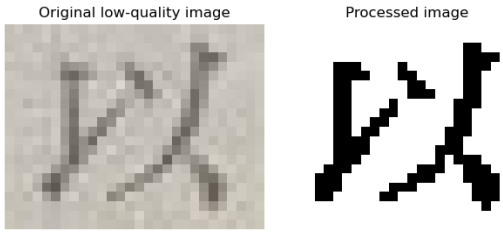
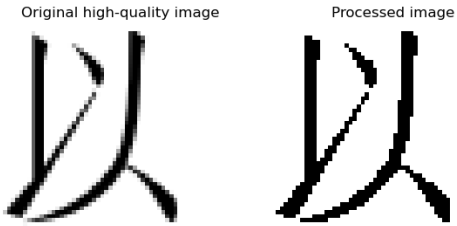



Fig. 1: Distribution of Labels Across Images

raw images and the processed images for both high-quality and low-quality images.




(a) Low-quality image transition




(b) High-quality image transition

Fig. 2: Pre-processing result

To briefly look at the two results, it can be said that after processing, the high-quality image stays quite the same; however, the low-quality image is sharpened and converted into a much better quality.

### C. Augmentation

Due to the limited size of our dataset, we implemented data augmentation techniques to expand the training data. We selected four augmentation methods, generating two versions of the dataset: one with 280k images and another with 700k images

- **Affine Transformation**: a combination of scaling, rotation, shearing, translation.
- **Rotation Transformation**: rotates the image with an angle.
- **Elastic Transformation**: transforms the morphology of objects in images.
- **Perspective Transformation**: models the effect of perspective in 3D scenes projected onto a 2D image plane.

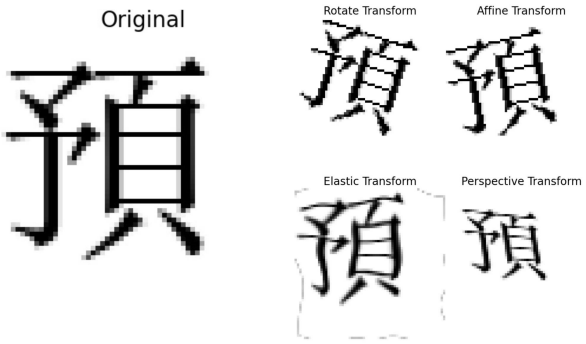



Fig. 3: Original image compare to 4 corresponding transformed images

By following the mentioned methods, we implemented random transformations using functions given by the **torchvision** module. The figure above illustrates the transformations applied to a random image from the dataset.

## III. Method

### A. Custom-Built Model

This was our initial approach into the project.after some research We identified two prominent research papers related to our topic [8] and [9] . Through two focus specifically on the recognition of handwritten Chinese characters, we recognize the direct relevance of these papers to our problem. Both papers have various methods from preprocessing to model training, and reported promising results, leading us to replicate and evaluate their methodologies on our dataset.

*1) Simple Convolution:* The first paper by Yuhao Zhang presented a straight-forward approach as they primarily experimented with basic convolutional neural networks of convolutional layer and fully connected layer section. They optimized and upgrade their models by adjusting the number of layers to improve performance and accuracy. Inspired by their basic architecture, we developed our own model,started with most basic one,the M5 which mean there are 5 layer in total, and progressively increased its complexity by adding layers. We choose to utilize varying kernel sizes for each layer, rather

than exclusively employing 3x3 kernels as proposed in the paper, while maintaining the same padding to preserve the image dimensions.

In addition, we implemented modifications to the model configuration by introducing two dropout layers with a rate of 0.25 between each fully connected layer in the M7X model inspired by [10]. This strategic inclusion aimed to enhance robustness and mitigate overfitting, while the engineering process yielded promising results, our pursuit of further optimization led us to explore additional research papers and techniques in search of even better performance. All the configuration is shown in Table I, after many experimentation, we settled on a 7-layer architecture. Further increasing the model depth did not create improvements in accuracy, suggesting that our current configuration provides a good balance between model complexity and performance.

TABLE I: Custom-built Model's Configuration

| M5 | M6 | M7 | M7X |
|---|---|---|---|
| 5 weight layers | 6 weight layers | 7 weight layers | 7 weight layers |
| Input image(64 x 64 gray-scale) | | | |
| conv11-64 | conv11-64 | conv11-64 | conv11-64 |
| Maxpool | | | |
| conv7-128 | conv7-128 | conv7-128 | conv7-128 |
| Maxpool | | | |
| conv5-256 | conv5-256 | conv5-256 | conv5-256 |
| Maxpool | | | |
| | | conv3-512 | conv3-512 |
| | | Maxpool | |
| | FC - 1024 | | |
| | | | Dropout 0.25 |
| FC - 1024 | | | |
| | | | Dropout 0.25 |
| FC - 2139 | | | |

*2) Alexnet and GoogleNet:* The next paper we investigated utilized a well-known architectures named AlexNet and GoogleNet. Despite being a fundamental convolutional neural network similar to our previous approach, we still choose to further exploration by re-built the AlexNet base on our dataset, hoping for improved results. However, AlexNet failed to deliver a good results, and its performance even fell short of the custom-built model we previously discussed.

GoogleNet, as known as Inception V1, introduced in [6], on the other hand, this network popular excels at image classification by utilizing multiple convolutional filters of varying sizes in parallel. While building GoogleNet from scratch would have been a time-consuming and complex undertaking, we opted to utilize a pre-trained version. The specifics of this implementation and its results will be discussed in detail in a later section.

### B. Pre-Trained Model

*1) Inception Network:* As mentioned previously regarding our use of GoogleNet, this "Inception module" is a groundbreaking convolutional neural network (CNN) architecture that revolutionized image classification. When building a CNN, selecting the appropriate kernel size is a challenging task. Small kernels, such as 3x3, can capture fine details and allow the network to go deeper, learning more complex features. However, using only small kernels can significantly increase the number of layers required to cover a larger receptive field, leading to higher computational costs and potential overfitting due to increased model complexity. On the other hand, large kernels, such as 7x7, can capture broader spatial features in a single layer, reducing the depth of the network. However, it can result in a loss of finer details and spatial resolution.

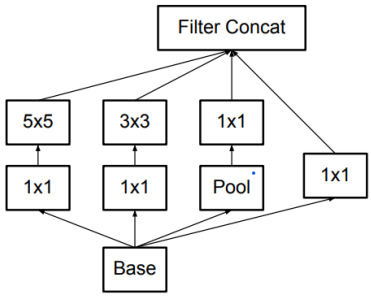



Fig. 4: GoogleNet (Inception V1) module

The Inception module addresses these issues by incorporating multiple kernel sizes within the same layer. Its core innovation lies in the "Inception module," which consists of parallel convolutional layers with different kernel sizes (1x1, 3x3, 5x5) showed in Figure 4, effectively capture features at different scales, leading to more comprehensive and accurate understanding of the image content. The outputs of these parallel convolutions and pooling operations are concatenated along the channel dimension, providing a rich feature representation that benefits from both small and large receptive fields.

For this matter we leverage the Inception V3 [7] model from timm library [1], an upgrade to the GoogleNet by replacing 5x5 convolutions with two successive 3x3 convolutions showed in Figure 5. Additionally, it uses auxiliary classifiers to help with gradient propagation in deeper networks, acting as additional outputs during training.

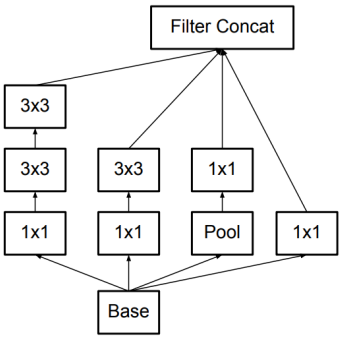



Fig. 5: Inception V3 module

*2) Xception Network:* The Xception model, which stands for "Extreme Inception" [2], is an extension of the Inception architecture, designed to further improve the efficiency and performance of convolutional neural.

Xception improves this by using depthwise separable convolutions, which replaces the standard Inception module with a series of depthwise separable convolutions, followed by a pointwise convolution. Depthwise convolution applies a single filter per input channel, capturing spatial relationships independently within each channel. Pointwise convolution, a 1x1 convolution, then combines these spatial features across channels. This separation allows the network to capture spatial and cross-channel information more efficiently and with fewer parameters. The Xception model that we use are the xception41.tf_in1k, trained on ImageNet-1k dataset.

*3) Residual Network:* Residual Network, or Resnet [4], solves a problem in deep learning architectures called Vanishing/Exploding gradient. This occurs when numbers of layers are increased, causing high error rate in both training and test dataset. ResNets address the problem using a technique called "skip connections", which connects activations of a layer to further layers, by skipping some layers in between. This forms a residual block show in , which can be stacked to create Resnets. The architecture enable the network to learn complex patterns and achieve impressive accuracy on various tasks, particularly in image recognition.

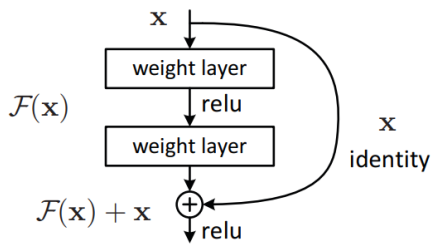



Fig. 6: Residual learning block

Base on the nature of the given dataset, we have decided to experiment with the smallest architecture with the least number of layers among the Resnets variants: ResNet18. With a dataset consists of approximately 50,000 images of 2139 classes, which is relatively small, fewer layers helps Resnet18 less prone to overfitting and more efficient in training. Specifically, we use the *resnet18.a1_in1k* variant from the *timm* library. This variant was pre-trained on the ImageNet-1k dataset using the ResNet Strikes Back A1 recipe.

*4) EfficientNet:* EfficientNet is a family of convolutional neural network architectures which introduces a scaling method to uniformly adjust all dimensions of depth, width, and resolution using a compound coefficient. Unlike traditional approaches that scale these factors arbitrarily, EfficientNet's method applies a consistent scaling strategy to network width, depth, and resolution based on a fixed set of scaling coefficients. For example, if we want to use $2^N$ times more computational resources, then we can simply increase the network depth by $\alpha^N$, width by $\beta^N$, and image size by $\gamma^N$, where $\alpha$, $\beta$, $\gamma$ are constant coefficients determined by a small grid search on the original small model. EfficientNet uses a compound coefficient $\theta$ to uniformly scales network width, depth, and resolution in a principled way.

The base EfficientNet-B0 network combines inverted bottleneck residual blocks from MobileNetV2 [5] with squeeze-and-excitation blocks to enhance its performance. Inverted residual blocks have a structure where the input and output are thin bottleneck layers, contrasting with traditional residual models that use expanded representations for the input. The intermediate expansion layer in these blocks employs lightweight depthwise convolutions to filter features, introducing non-linearity.

The squeeze-and-excitation block further improves the network's representational power by performing dynamic channel-wise feature recalibration, enabling more effective feature extraction and utilization. EfficientNet models, from version B1 to B7, are scaled from the initial B0 version and typically use fewer parameters and FLOPS than other ConvNets, while delivering equal or even higher accuracy. Due to their impressive performance, we selected two pre-trained models, *efficientnet_b0* and *efficientnet_b1_pruned*, from the *timm* library as our primary models. These models are more lightweight than other advanced versions of EfficientNet, making them appropriate for the scale of our project.

*5) Vision Transformer:* Vision Transformer, or ViT for short [3], is a novel deep learning model that applies the Transformer architecture, originally designed for natural language processing, to image recognition. By treating images as sequences of patches and leveraging self-attention mechanisms, ViT achieves performance on various image classification tasks. Despite its state-of-the-art capabilities, applying ViT to our dataset has not yielded significant improvements in results. We thought that it could be due to the dataset quality as ViT models typically require large and diverse datasets, while each label in the traning set only have around 50 images. And also traning with transformer is a very difficult process because the hyperparameters and architecture of ViT need to be carefully tuned. Factors such as the size of the patches, the depth of the model, and the number of attention heads can significantly impact performance. With our limited resources, we haven't been able to experiment sufficiently to achieve optimal results.

### C. Ensemble learning

Ensemble learning combines multiple machine learning models to improve predictive performance. Instead of relying on a single model, ensembles leverage the strengths of diverse models, effectively "voting" on the best prediction, we choose to combine the top performing models from different methods to produce the final prediction. This process can reduce variance, increase accuracy, and enhance robustness against overfitting, making ensembles powerful tools for complex problems.

We experimented with a variety of models and then combined them. The models we chose for ensembling were EfficientNet B0, EfficientNet B1, Xception, and ResNet18. We also tried incorporating Inception v3 but it ended up decreasing performance.

## IV. Result

### A. Data Augmentation

Using the listed methods, we have augmented the original dataset into 2 separate datasets, namely Data200 and Data700 by the following ratio:

TABLE II: Ratio used to augment each original images

| Dataset | Original | Affine | Rotation | Elastic | Perspective |
|---|---|---|---|---|---|
| Data200 | 1 | 1 | 1 | 1 | 1 |
| Data700 | 1 | 2 | 2 | 4 | 4 |

For the first 2-3 epochs trained on every models, Data200 and Data700 resulted in a significantly better accuracy (∼5-7%) comparing to the original dataset. However, the augmented datasets quickly converged after 5-6 epochs, while the original data peaked approximately the same accuracy after 15-18 epochs. In the end, the augmentation methods might be able to accelerate the training processes but couldn't improve the overall accuracy.

### B. Training Results

The Table III shows the results of all the models during training. We experimented with a variety of models and then combined them. It is evident that state-of-the-art image classification models, utilizing advanced techniques, significantly outperform models relying solely on conventional convolutional layers. These cutting-edge models leverage innovative architectures, such as residual connections, and efficient network designs, to achieve superior accuracy. For instance, models like EfficientNet, ResNet, and Xception have demonstrated remarkable capabilities in image recognition tasks, surpassing the performance of traditional convolutional neural networks. This underscores the importance of exploring and incorporating novel architectural advancements to enhance image classification capabilities.

TABLE III: Experiment results on the Validation dataset

| Method | Validation Accuracy (%) |
|---|---|
| M6 | 79.8 |
| M7 | 80.6 |
| M7X | 81.6 |
| AlexNet | 70.2 |
| ResNet18 | 88.5 |
| EfficientNet B0 | 91.0 |
| EfficientNet B1 | 92.4 |
| Inception V3 | 87.5 |
| Xception | 88.6 |
| Vision Transform Tiny | 69.1 |
| **4 Top Model Ensemble** | **93.6** |

## V. Conclusion

In this report, we present our implementation of Sino-nom character recognition utilizing multiple deep learning models to achieve optimal performance. Our approach involves exploring diverse model architectures and subsequently combining their strengths through ensemble techniques. Nevertheless, we believe that there is still room for improvement with more computational resources and a more diverse dataset.

## References


[1] timm (PyTorch Image Models) — huggingface.co. https://huggingface.co/timm. [Accessed 28-05-2024].

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014.

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[8] Yuhao Zhang. Deep convolutional network for handwritten chinese character recognition. *Computer Science Department, Stanford University*, 2015.

[9] Zhuoyao Zhong, Lianwen Jin, and Zecheng Xie. High performance offline handwritten chinese character recognition using googlenet and directional feature maps, 2015.

[10] Junyi Zou, Jinliang Zhang, and Ludi Wang. Handwritten chinese character recognition by convolutional neural network and similarity ranking, 2019.