COMP2261 - Machine Learning Report

Michal Pluta

***Abstract*— This study compares k-Nearest Neighbors (kNN) and Logistic Regression for classifying Chinese characters by evaluating the models' accuracy and efficiency in handling high-dimensional data, thereby providing insights into complex character recognition and other digital applications.**

***Index terms*—Machine Learning, K-nearest-Neighbours, Logistic Regression, Hyperparameter Tuning, CNN**

I. Introduction

Mandarin Chinese is one of the most spoken languages with 1.3 billion native speakers globally. In countries with significant Chinese-speaking populations, the ability to accurately and automatically identify characters has applications across a variety of domains. To name a few:

- Recognising mail addresses in the postal service, reducing the dependency on manual sorting.
- Digitisation of cultural heritage, where manual digitisation is too difficult or infeasible time-wise.
- Improving accessibility for the visually impaired or those who have reading difficulties.

This report explores the application of machine learning methods on a dataset comprising Chinese (Simplified) characters, akin to the widely recognised MNIST dataset for handwritten digits. The significance of this study lies in the inherent complexity and variety of Chinese script, and the motivation lies in a personal interest in learning the language.

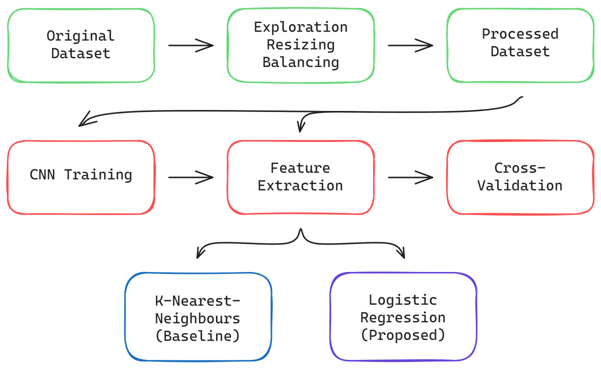
Original Dataset
Exploration Resizing Balancing
Processed Dataset
CNN Training
Feature Extraction
Cross-Validation
K-Nearest-Neighbours (Baseline)
Logistic Regression (Proposed)

Figure 1: Workflow of the proposed ML System

The primary objective is to demonstrate the performance, tuning process, and limitations of two separate machine learning models.

Project Summary

The study conclusively demonstrated that Logistic Regression significantly outperforms k-Nearest Neighbors (kNN) in terms of accuracy and inference time when classifying Chinese characters. However, Logistic Regression's longer training time becomes a substantial limitation when dealing with large datasets. The key lesson from this project is the importance of selecting an appropriate algorithm that is not only effective but also scalable, taking into account the size & complexity of the dataset.

II. Dataset Overview

While Chinese has more than 50,000 characters with roughly 6,500 in daily use, we will only deal with a subset part of the HSK 1 curriculum. HSK is a Chinese Proficiency Exam with levels ranging from 1 (Beginner) to 9 (Near-native).

Chinese characters, also known as Hanzi, are composed of strokes, the basic units of writing, arranged in a specific order and direction. The way these strokes combine give rise to a vast array of characters, each with its own unique meaning and structure.

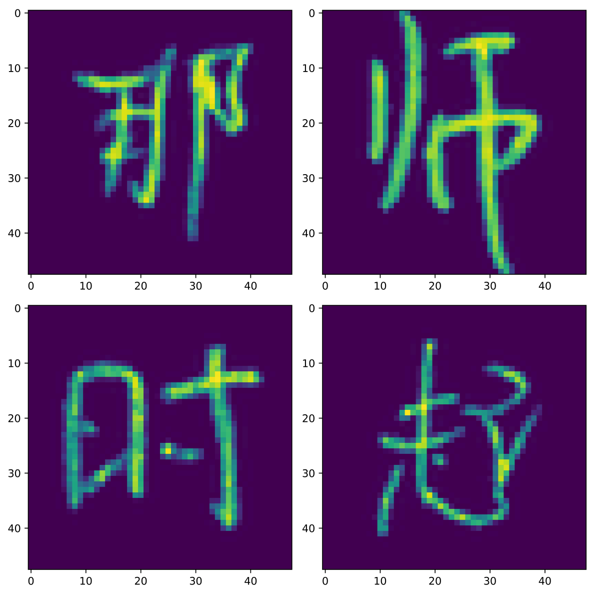

Figure 2: Examples of images in the dataset

The uniqueness of handwritten Chinese characters becomes even more pronounced when considering China's population. This alone leads to a huge variation in style, consistency, size, alignment, and thickness of strokes (Figure 2).

III. Dataset transformations

The original dataset [1] contains 178 classes of images, split using an 80/20 train-test split. The images are greyscale with a 1:1 aspect ratio, and in total there are 131,946 samples. To minimise bias and to ensure the train and test sets were representative of the whole dataset, the sets were merged and then shuffle-split through stratification. This also allowed the dataset to be standardised using one directory only.

Image Resizing

One clear issue with the original dataset were the varying image dimensions.

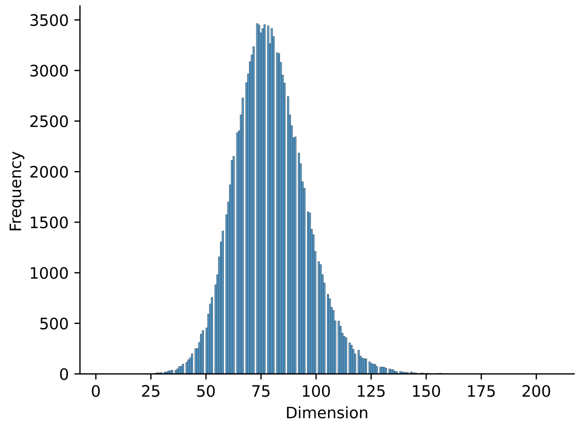

Figure 3: Distribution of image dimensions in the original dataset

This was a problem because the CNN used for feature extraction has specific input tensor dimension requirements, hence, all images had to be standardised. 48×48 was chosen as a suitable image size as it meant most images could be downsampled. Downsampling is often a better technique than upsampling as it doesn't limit the model's ability to learn key features like object boundaries. Additionally, due to the complexity of Chinese characters, the images could not be downsampled further due to the likely loss of information.

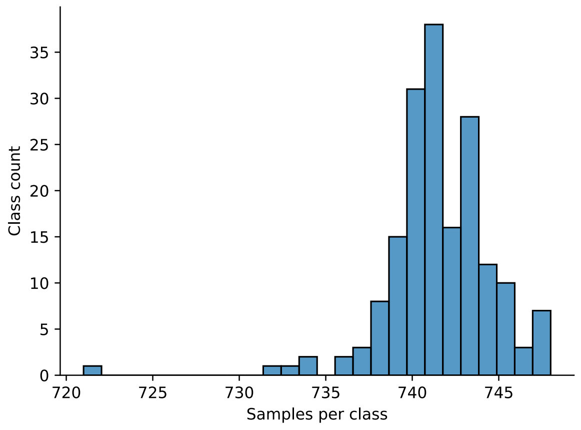

Figure 4: Distribution of the number of samples in each image class.

From further exploration, a class imbalance was evident.

Imbalance metric	Value
Imbalance Ratio	1.03745
Interquartile Range	3.0
Coefficient of Variation	0.00409

Table 1: Dataset imbalance metric values

Evaluating the imbalance ratio, IQR, and coefficient of variation, it was clear the imbalance was minimal and insignificant.

Formal Definition

More formally, the images $x^{(\mathrm{i})}$, and one-hot encoded labels $y^{(\mathrm{i})}$ in this system are defined as follows:

$$x^{(\mathrm{i})} \in \mathbb{R}^{48\times 48} \tag{1}$$

$$y^{(\mathrm{i})} \in \mathbb{R}^{178} \tag{2}$$

By implication, the input tensors are defined as

$$x \in \mathbb{R}^{131946\times 48\times 48} \tag{3}$$

$$y \in \mathbb{R}^{131946\times 178} \tag{4}$$

This allows us to define the CNN's feature extraction operation, extract() as

$$\mathrm{extract}\left(x^{(\mathrm{i})}\right) = f^{(\mathrm{i})} \in \mathbb{R}^{512} \tag{5}$$

Feature Extraction

Since the dataset was comprised of images, relevant features had to be extracted from each image. One of the most effective ways of doing this is by using a CNN (Convolutional Neural Network).

While pre-trained CNNs such as Resnet50 were initially considered due to their simplicity and outstanding performance, they yielded feature vectors which were simply too large (2048), resulting in unmanageable training times. One way to handle this would be to perform dimensionality reduction using a technique such as PCA, however, this is just as, if not more computationally intensive.

Instead, a custom CNN was developed using Keras' deep-learning library. The model features 3 2D-Convolution layers, each followed by a MaxPooling layer. While developing the model, severe overfitting was noticed. This was addressed by adding Dropout layers to the model, which would randomly set a fraction of the input units to 0. This improved the model's ability to generalise.

The CNN was tuned using a Bandit-Based tuner, Hyperband [2], provided by the Keras library. Once optimised, the model was cross-validated to ensure major overfitting was not present.

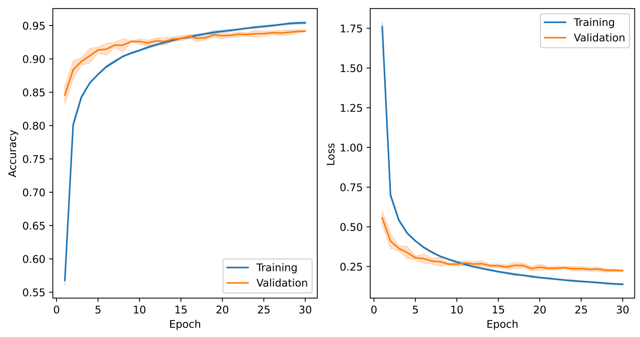

Figure 5: Graph of training and validation accuracy throughout training the CNN

From Figure 5 it is clear that some overfitting is still present, however, this amount is tolerable. The reason for the model performing better on validation data in the first few epochs is due to the Dropout layer. A dropout layer is a regularisation mechanism that is turned on when training and is turned off when testing. This results in the model performing worse on the training data. [3], [4]

IV. Evaluation Metrics

- **Weighted Accuracy** - Since my dataset has a small amount of imbalance, to be on the safe side I will use the weighted accuracy across all predicted classes instead of average accuracy.
- **F1 Score** - In my classification problem there is no greater negative impact caused by a low sensitivity (Recall). This is not the case with something like medical image classification where low sensitivity could have serious consequences. Therefore, I will use the F1 score which is the harmonic mean of Recall and Precision as they correlate inversely with each other.
- **Training time** - While the total number of Chinese characters remains fairly constant, every so often new characters for complex ideas or newly discovered chemical elements are proposed, and this would warrant retraining/adjusting the model. Lower training times would allow the model to be updated faster.
- **Inference Time** - In applications such as real-time translation, language learning, or even autonomous driving where signs need to be read within fractions of a second, it is crucial to identify text with minimal latency. Additionally, for services with large volumes of data, an efficient, scalable, and high-throughput system is ideal.

V. Model Evaluation

Overview & Justification

For our baseline model, we've selected kNN (k-Nearest Neighbors), a standard, naive algorithm used in classification tasks. Our proposed model is Logistic Regression, chosen for its specific benefits suited to this dataset.

Starting with kNN, it is a non-parametric, lazy-learner. Unlike typical models that learn patterns from training data, kNN instead defers all computations until it's time to make predictions on new data. This characteristic may be advantageous on smaller datasets, but given the large size of our dataset, it could significantly impact the speed of making predictions.

Additionally, kNN suffers a key drawback, the Curse of Dimensionality. The kNN algorithm assumes that similar points in space share the same label [5]. While this may be the case for smaller dimensions, the feature vectors we obtained in the data exploration section have dimensionality 512, this means that any two vectors belonging to the same class may not be close to each other at all.

On the other hand, logistic Regression is a parametric model that learns the weights of the features during the training process, which makes it more suitable for handling high-dimensional data [6].

Another key benefit of logistic regression is its efficiency. Unlike kNN which requires storing the entire dataset and searching for the k nearest neighbors on each prediction, logistic regression only needs to apply the learned weights to the test data, likely making predictions much faster especially for large datasets.

Parameter exploration - kNN

Although kNN is classified as a lazy learner and lacks traditional parameters, it utilizes several hyperparameters including `n_neighbors`, `weights`, `metric`, `p`, `leaf_size`, and `algorithm`. Through experimenting, I found that the hyperparameters `n_neighbors`, `weights`, and `metric` significantly impacted validation accuracy, leading me to focus on their optimization.

I began by first exploring the relationship between `n_neighbors` and validation accuracy.

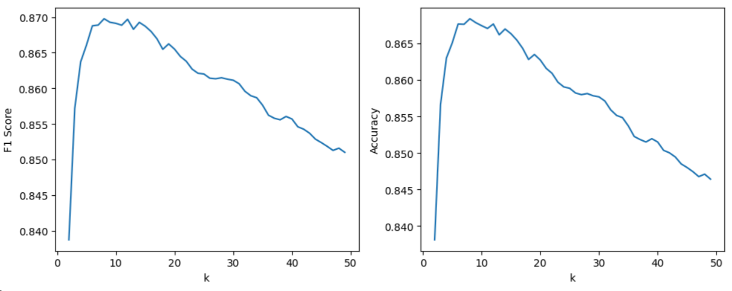

Figure 6: Accuracy & F1 Score of K-Nearest Neighbours with varying k

I found that the optimal values for `n_neighbors` lie somewhere between 5 and 20, which allowed me to narrow down the field of possible candidate values. Additionally, I noticed that the f1-score was roughly correlated with the validation accuracy, which led me to believe that the model was effective at not only being accurate but also at maintaining a good balance in avoiding misclassifications.

	Default	**Candidate Values**
`n_neighbours`	`5`	`range(5, 20)`
`weights`	`'uniform'`	`'uniform', 'distance'`
`metric`	`'minkowski'`	`'euclidean', 'manhattan', 'minkowski'`

Table 2: Summary of kNN hyperparameters and candidate values

By performing an exhaustive Grid Search of all the possible configurations, I found that `n_neighbors=10`, `weights='distance'`, and `metric='euclidean'` were ideal.

Parameter exploration - Logistic Regression

On the other hand, logistic regression does have parameters which are intrinsic to its model. More specifically, logistic regression assigns weights to each feature and adjusts these weights during the training process.

Similarly to kNN, logistic regression also has its hyperparameters, namely `solver`, `C`, `penalty`, `dual`, `tol`, `l1_ratio`. Once again, from experimentation, I found that `solver`, `C`, `penalty`, and `l1_ratio` (if `penalty` was set to `elasticnet`) were the most influential in terms of training time and validation accuracy.

Since my task was concerned with multi-class classification, only the `lbfgs`, `newton-cg`, `sag`, and `saga` solvers supported multinomial loss and so this reduced my field of candidate values.

	Default	GridSearch
`solver`	`'lbfgs'`	`'lbfgs', 'newton-cg', 'sag', 'saga'`
`penalty`	`'l2'`	`'l1', 'l2', 'elasticnet', 'None'`
`C`	`1`	`10, 5, 1, 0.5, 0.1`
`l1_ratio` (saga only)	`None`	`0.25, 0.5, 0.75`

Table 3: Summary of hyperparameters for Logistic Regression and candidate values

Similarly, by performing an exhaustive Grid Search of all the possible configurations, I found that `solver=lbfgs`, `C=5`, and `penalty=l2` were ideal.

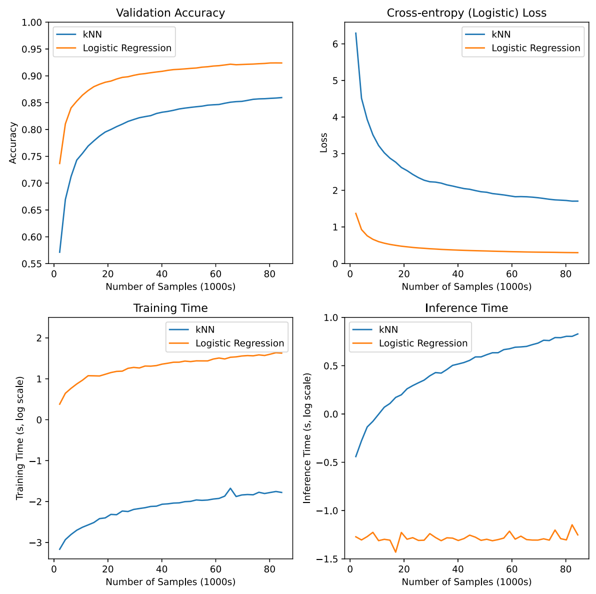

Figure 7: Accuracy vs Training Accuracy of Logistic Regression with varying amounts of training samples

Upon comparing the performance of both models against each other with varying amounts of training data, it was clear that logistic regression was much more accurate at classifying and was on average 8% more accurate (92% vs 84%). Unsurprisingly, it was also clear that the kNN model was much faster at training. This was essentially because kNN's 'training' is just loading the feature vectors into memory. Most importantly, the logistic regression model had a much lower inference time, which is solely attributed to the way it learns the training data and updates its weights. In theory, this makes the logistic regression model much more scalable.

VI. Self Evaluation

Lectures

Throughout the lectures, I was most intrigued by the mathematical concepts that underpin each machine-learning model. It made me glad to know that the mathematical foundations I had practiced tirelessly last year were crucial to understanding both the inner workings of each model as well as the intuition behind how they all were developed. The lectures also showed me that not all models behave like a 'black box', and many have visual representations such as kNN and Lloyd's Algorithm in KMeans.

Coursework

One particular thing this coursework has taught me is the machine learning workflow. The lectures didn't go into much detail about the practical aspect, however, this coursework was a very good introduction on how to approach a machine learning problem. From data preprocessing and feature selection to model training and evaluation, I learned to consider each step critically. Most importantly this coursework highlighted the iterative nature of machine learning, where models are continuously refined based on performance metrics.

Module difficulties

My main difficulty in the module is being able to effectively translate my theoretical understanding into a practical solution. Ultimately, most of this content is completely new to me, hence I've never had a chance to implement or tune any models independently. Thankfully, the practicals were able to address some of these concerns as I got hands-on experience implementing and adjusting various models.

Reflection

I believe the biggest challenge of this assignment was the size of the dataset and the disproportionate computational power I possessed. I approached this project wanting to do image classification related to one of my interests, however, I didn't fully consider how long it would take to train even a single model with such a vast quantity of data. Moreover, I believe it would have been more effective to use a pre-trained CNN such as Resnet to ensure the feature extraction was as accurate as possible since all the other models rely on this.

Unique contributions

While my models do not constitute anything novel, I find it nevertheless an area of research that should and likely will be explored.

References

[1] V. Kyzym, "Chinese handwriting recognition : HSK 1". [Online]. Available: https://www.kaggle.com/datasets/vitaliikyzym/chinese-handwriting-recognition-hsk-1

[2] K. Team, "Keras Documentation: Hyperband Tuner". [Online]. Available: https://keras.io/api/keras_tuner/tuners/hyperband/

[3] A. Soleymani, "Your validation loss is lower than your training loss? this is why!". [Online]. Available: https://towardsdatascience.com/what-your-validation-loss-is-lower-than-your-training-loss-this-is-why-5e92e0b1747e

[4] K. Team, "Keras Documentation: Keras FAQ". [Online]. Available: https://keras.io/getting_started/faq/

[5] "K-Nearest-Neighbors / Curse of Dimensionality 2022". [Online]. Available: https://www.cs.cornell.edu/courses/cs4780/2022fa/lectures/lecturenote02_kNN.html

[6] A. KumarI, "KNN vs logistic regression: Differences, examples". [Online]. Available: https://vitalflux.com/knn-vs-logistic-regression-differences-examples/