William Peracchio

Professor Lopresti

CSE 498-010

14 December 2020

Final Project: Comparing Algorithm Robustness Across Languages in MNIST

Abstract

MNIST (Modified National Institute of Standards and Technology) is one of the most important sample datasets used in the fields of computer vision and machine learning. The input data is simple and widely used for training algorithms to identify handwritten digits. Despite this popularity, MNIST does not represent the entirety of numeral systems in use by humans. There are many numeral systems in place beyond the standard Arabic numerals of 0-9, such as the Chinese numeral system. Internationalization is one of the most topical themes in computer science currently, and designing robust systems which are resilient for many different users is critical. This paper performs a comparative analysis of common classification algorithms across two different numeral systems - Arabic numerals through the standard MNIST dataset, and Chinese numerals through a Newcastle University-created dataset that mimics the style of MNIST. Following a comparative analysis of the different algorithms on the dataset, a discussion follows which includes commonly confused numbers for the algorithms and discussions for potential further work in the field of sample dataset internationalization considerations.

Introduction

Internationalization is one of the most pressing modern issues in the computer software industry, and for good reason, with software more likely than ever to reach a truly global audience. According to the W3C, internationalization is defined as “the design and development

of a product, application or document content that enables easy localization for target audiences that vary in culture, region, or language" [1]. One of the most common and obvious manifestations of internationalization is designing for differences in languages (human languages, not programming languages), as a number of different language features require modification to traditionally English-based or Latin-based language systems. This may include features such as Arabic's right-to-left writing, alphabets beyond the Latin alphabet, or even the inclusion of logographic languages, which may lack a specific "alphabet" in the first place.

While internationalization does pose additional challenges to modern computer science development across the field, it also allows for the development of more resilient and robust systems that can benefit all users. For example, the ASCII-encoded system was originally developed in the 1960s and was sufficient for representing English and a small handful of other, generally Latin-based languages [2]. Now, most modern applications utilize the unicode system, which has support for over 140,000+ different characters and gives support to many more languages [3]. Unicode was developed as an evolution of character encoding, with internatlizationization in mind. But it was not just new users that benefited from the standard of internationalization, as even English-speakers are now able to benefit from the use of unicode's features (such as this emoji 😊, which would not have been possible under the ASCII system). While internationalization poses additional challenges to the field of computer science, it is also clear that it represents an opportunity for the development of more robust, more resilient, and ultimately, more equitable systems.

As computer vision continues to become a larger area of computer science research, it is critical that this focus on internationalization and the engineering of robust systems is kept in mind. Cultural differences will continue to manifest themselves in the data present to a given

computer system, and the output of these systems will be a reflection of their inputs - *garbage in, garbage out* as the old data science adage goes. One of the most important benchmarks in the area of computer vision is that of the Modified National Institute of Standards and Technology (MNIST) database [4]. The MNIST database features a collection of normalized handwritten digits, which tasks a given algorithm to predict which number was being drawn. Though the MNIST dataset represents an ideal scenario in some respects due to the normalization of the data, its widespread usage across the field of computer vision makes it a prime candidate for comparison to numeral systems that are present in other languages. While Arabic numerals (0123456789) are the default in much of the Western world, it is not the only numeral system widely used. Perhaps most notably, Mandarin Chinese, with over 1 billion speakers, often utilizes a different numeral system (零一二三四五六七八九十百千万亿). It is for this reason that this paper explores the relationship between the performance of Arabic and non-Arabic numeral based MNIST-style data.

Overview of Project

This project seeks to examine the relationship between algorithm performance and robustness across different language numeral systems. For the numeral systems examined, algorithms will be compared in terms of relative performance. The purpose of this project is not to shoot for the highest accuracy in classification, as that has already been studied in-depth for MNIST as a whole [4]. Rather, this paper will instead focus on how algorithms compare to one another when considering different numeral systems. This project will cover Arabic and Chinese numerals, and measure the performance of several classification algorithms against these datasets, which are described below.

In addition, a special point of analysis examines the confusion matrices for the given models of classification, which helps to determine how models perform at the individual digit level in terms of classification. Particularly poor classification models will likely not produce meaningful confusion matrices, and so analysis on confusion matrices will focus on the better performing algorithms.

Dataset Description

The following discusses the specifics of each of the datasets utilized for analysis. A comparison of the datasets can be found in **Table 2** at the end of the section.

Arabic Numerals

The standard MNIST dataset, which utilizes Arabic numerals, features a large collection of normalized handwritten digits and is one of the most important benchmarks in the fields of computer vision and machine learning. The MNIST dataset, as the name suggests, is a modified subset of the National Institute of Standards and Technology's handwriting dataset that features only digits. The dataset contains 60,000 instances, each of size 28px x 28px, which leaves a feature vector with 784 inputs [4]. As would be expected with the Arabic numeral system, there are 10 potential output classes for a potential algorithm to output (0-9).

Chinese Numerals

The Chinese numeral dataset was obtained from an MNIST-inspired project from Newcastle University [5]. The dataset contains data from one hundred Chinese national students, with each student drawing each of the 15 simplified Chinese characters 10 times (Simplified Mandarin is the standard writing system for Mainland China) [6]. Much like the original MNIST dataset, the images were normalized and grayscale to fit within the square. The dataset contained both a high-resolution, 64px x 64px per instance version, and a smaller, 28px x 28px per instance

version of half of the participants that were downsampled. This led to feature vectors of size 4096 and 784 per image, respectively. Both datasets were used for analysis, and are noted separately as low resolution and high resolution, respectively.

It should be noted that Chinese does not have a direct, one to one mapping of Arabic numerals, and instead utilizes a different way of creating numbers. For example, the number 498 in Chinese becomes 四百九十八 - literally 4 hundreds, 9 tens, 8 (ones). This leads to a greater number of "digits" as potential output classes, with 15 in Mandarin compared to 10 in the Arabic numeral system. This is shown in **Table 1**, which shows a comparison between the numeral systems. With a greater number of potential output classes, one would generally expect lower performance metrics assuming the Chinese digits are not significantly "easier" than their Arabic counterparts.

Table 1: Summary of the Numeral Systems

Character Value	**Arabic**	**Mandarin (Simplified)**
0	0	零 (〇 is also used, but is not considered for this dataset)
1	1	一
2	2	二
3	3	三
4	4	四
5	5	五
6	6	六
7	7	七
8	8	八
9	9	九
10		十
100		百
1000		千
10000		万
100000000		亿
# Output Classes	**10**	**15**

Table 2: Summary of the MNIST Datasets

Dataset	Arabic	Chinese (HighRes)	Chinese (LowRes)
# Output Classes	10	15	15
Image Dimensions	28x28	64x64	28x28
Total Participants	100	100	50
Total Instances	60000	15000	7500

Data Transformation

Due to the preprocessed nature of MNIST datasets, there was relatively little processing required to get the data into a format that the algorithms could interpret. All of the data instances came with their accompanying data labels, which were used as ground truths in model evaluation. Some Spark helper methods were used for ensuring that data was in the correct format for the algorithms, which are further explained in the attached code.

The standard MNIST dataset was read from a custom LIBSVM format, direct from the original data source, which Apache Spark was able to process. The images were already isolated into a grayscale channel before processing into the LIBSVM format, and did not need to be processed beyond that [4], [7].

The Chinese MNIST dataset had to be read from individual JPG files and into a dataframe in Scala's implementation of Apache Spark. As with the standard MNIST dataset, the images were isolated to a single, grayscale channel [6], [7].

Each of the datasets were broken into train and test instances in an 80%/20% ratio, and the resultant analysis uses the following classification algorithms to train the model on the training subset and evaluate on the test set. Accuracy, in this case the ability of the models to correctly predict the digit being drawn, was the primary measure analyzed for the purpose of this project.

Classification Algorithms

Each of the following algorithms came from Apache Spark's Scala implementation of the Machine Learning (ML, version 2.4.2) library. Five algorithms investigated across the datasets: naive Bayes, decision trees, random forests, multinomial logistic regression, and multilayer

perceptrons. A brief discussion of each of the algorithms tested and the hyperparameters tuned is discussed below [7].

Naive Bayes

The naive Bayes algorithm was used as a baseline for classification, as it only probabilistically weighs the value of each potential output class based upon input classes. Naive Bayes normally requires inputs to be independent for sufficient performance, but considering that drawings (with continuous lines from a writing utensil) are likely to produce highly-dependent resultant vectors, low performance is expected for this classifier [8].

Decision Tree

Decision trees are one of the most interpretable classification algorithms, and creates a series of decisions that are used to determine the final output class. Decision trees can have trouble dealing with non-categorical data, and are not expected to be particularly resilient for this dataset due to the numerical structure of the underlying data. Individual decision trees are also susceptible to being particularly brittle, and do not always generalize well to previously unseen data [7].

Random Forest

Random forests are an ensemble method which utilizes a collection of decision trees to help output a final class for classification problems. Random forests are generally expected to be more resilient than their singular tree counterparts, as random forests are an ensemble method and are trained using a subset of the data each time, which helps them generalize to unseen data before. The number of trees used in the forest was tuned for the best performance, but was not found to change the results significantly past 50 trees in the forest [7].

Multinomial Logistic Regression

Multinomial logistic regression generalizes logistic regression, normally used for binary classification, for multiclass problems and outputs a final predicted class. Multinomial logistic regression considers the inputs of all variables, and can handle dependent variables well. This leads to the expectation that it will perform relatively well for the hand-drawn nature of the input data, which as discussed before, with each pixel likely to be dependent upon neighboring pixels [7].

Multilayer Perceptrons (MLP)

Multilayer perceptrons are an implementation of an artificial neural network, which simulates the neurons of the brain in transforming a series of inputs into a series of outputs via a user-specified number of "hidden" layers in the middle. While the number of hidden layers can vary for a given problem, MLPs can generally still see high performance with just a single hidden level. MLPs generally feature high performance, even for previously unseen data, at the expense of interpretability. The purpose of this paper will not be to customize a given neural network to maximum performance, which has already been covered in depth in other areas of research, but rather see how performance compares across the numeral systems for just a single hidden layer model [7].

Results

Table 3: Summary Results by Algorithm

Algorithm	Arabic Accuracy	Chinese (LowRes) Accuracy	Chinese (HighRes) Accuracy
Naive Bayes	0.8207	0.0404	0.0392
Decision Tree	0.6694	0.2673	0.2166
Random Forest	0.8536	0.5003	0.3595
Multinomial Logistic Regression	0.9147	0.4770	0.4128
Multilayer Perceptrons	0.9705	0.7923	0.7615

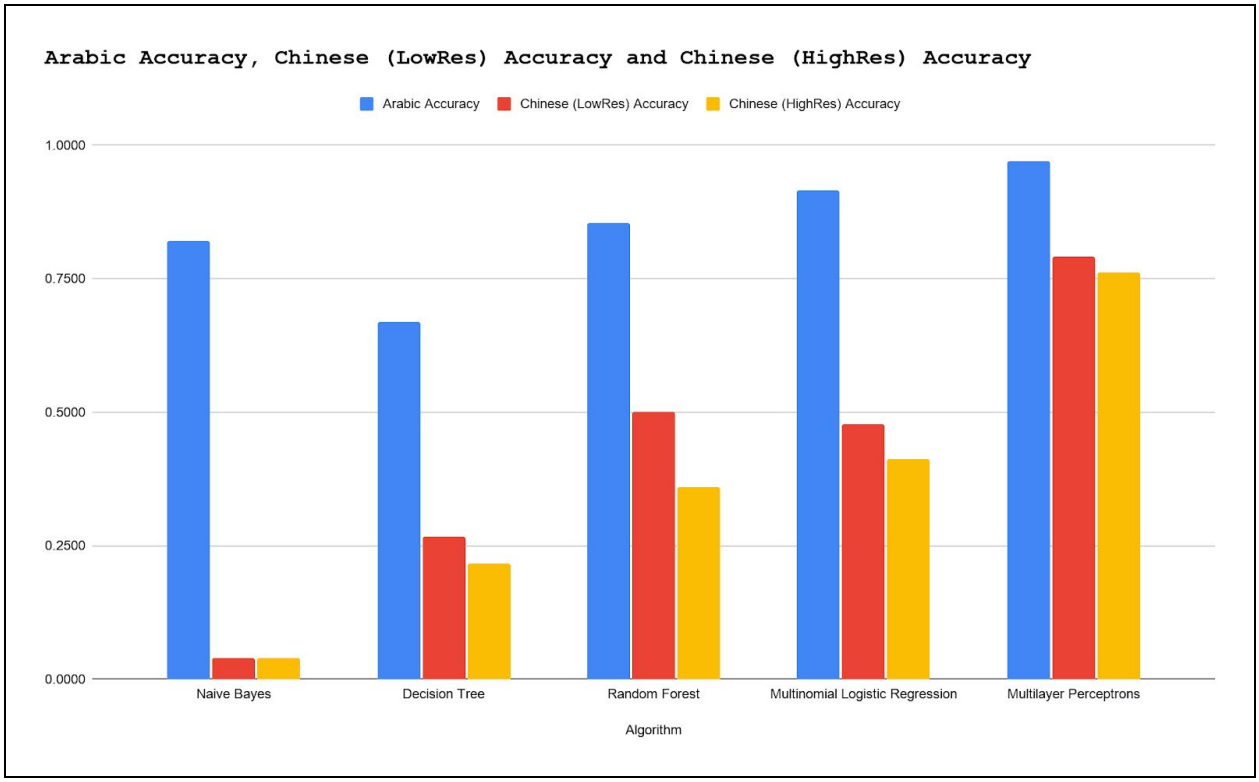

Figure 1: Summary Performance by Algorithm

Analysis

As was expected, Arabic numerals exhibited significantly higher accuracy measures for every different classification algorithm tested, shown in **Table 3**. This is unsurprising considering Arabic numerals featured a fewer number of potential output classes and a much larger training sample to pool from compared to the Chinese datasets. The difference in performance between the high resolution and low resolution Chinese datasets is very surprising, considering the low resolution is a downsampled version of the high resolution dataset. Potential explanations could include that the high resolution version introduced additional noise into the models, which may have led to overfitting.

Naive bayes and decision trees performed the worst out of the tested classification metrics, which was inline with expectations, considering these methods either do not handle dependent data well or generalize to unseen data well. Naive baye in particular was no better

than a random guess for the Chinese datasets, and the decision tree was only a slight improvement over a random guess. Pure probabilistic models that do not consider the context of the characters will likely have a difficult time adapting for larger output classes, as the dropoff between the Arabic and Chinese datasets is dramatic.

Random forests and multinomial logistic regression had comparable performance between themselves, but still showcased a dramatic difference in performance between the Arabic and Chinese datasets. Random forests experienced a greater discrepancy between the high resolution and low resolution Chinese datasets than the multinomial logistic regression, which may again be caused by the introduction of noise in the high resolution data that led to overfitting.

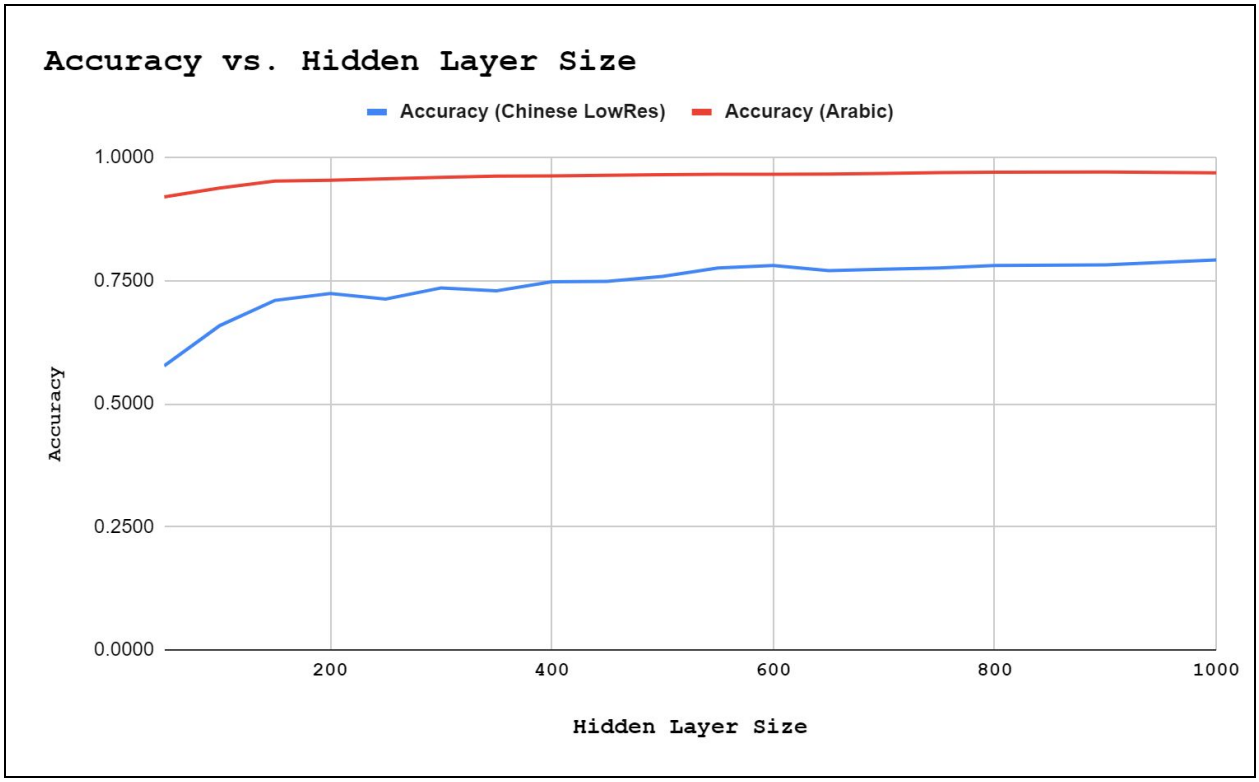

Figure 2: Accuracy vs. Hidden Layer Size for Arabic and Chinese Low Resolution Datasets

Multilayer perceptrons (MLPs) performed very well for the Arabic dataset, and while the gap between the Arabic and Chinese datasets was not closed, it was minimized in terms of raw

accuracy differences compared to all other algorithms. It should be noted that the higher resolution Chinese dataset began running into memory issues for the upper bound of the hidden layer tested, and could potentially converge with the low resolution dataset with more nodes in the hidden layer. This speaks to the robustness of the MLP algorithm, as it performed by far the best across all of the algorithms for all datasets tested. As shown in **Figure 2**, performance stabilized or slightly increased as the size of the hidden layer approached the size of the output layer. The high resolution version of the Chinese dataset ran into memory errors, and was not tested to the full size of its output layers. Given the high performance of the MLP, the confusion matrices of the models will be analyzed to see where the models are making errors.

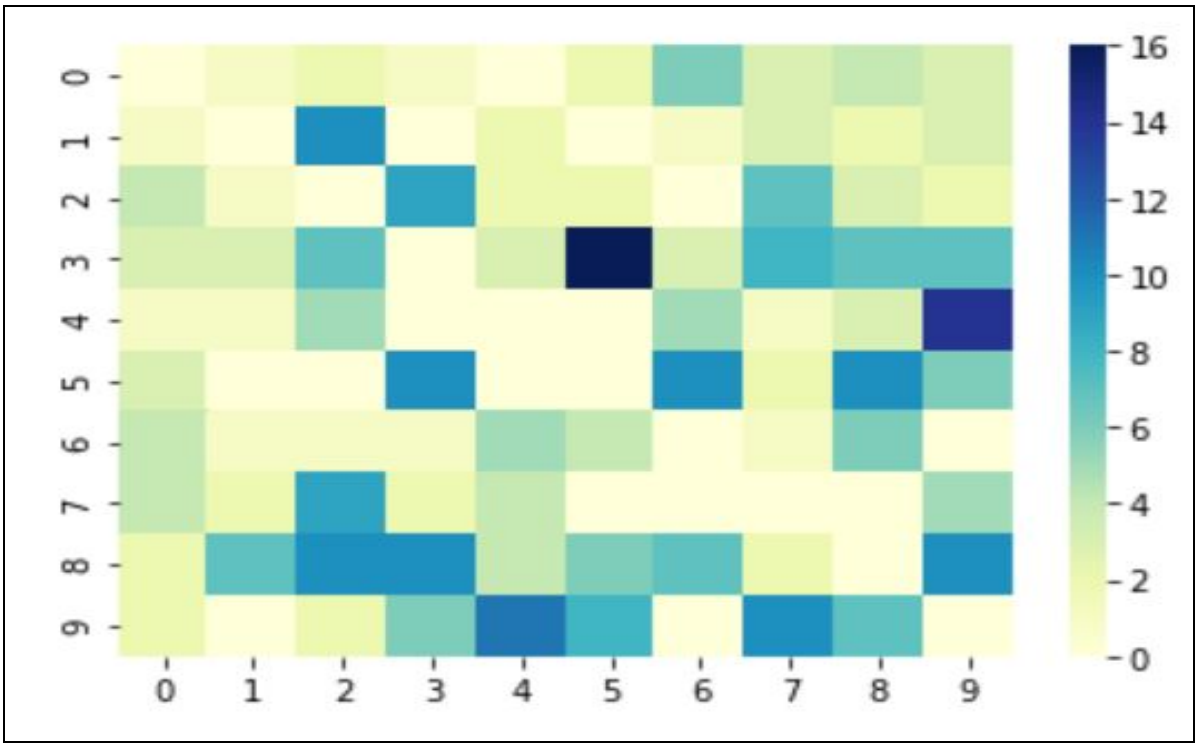

Figure 3: MLP for Arabic Numerals Heatmap (Diagonals Normalized)

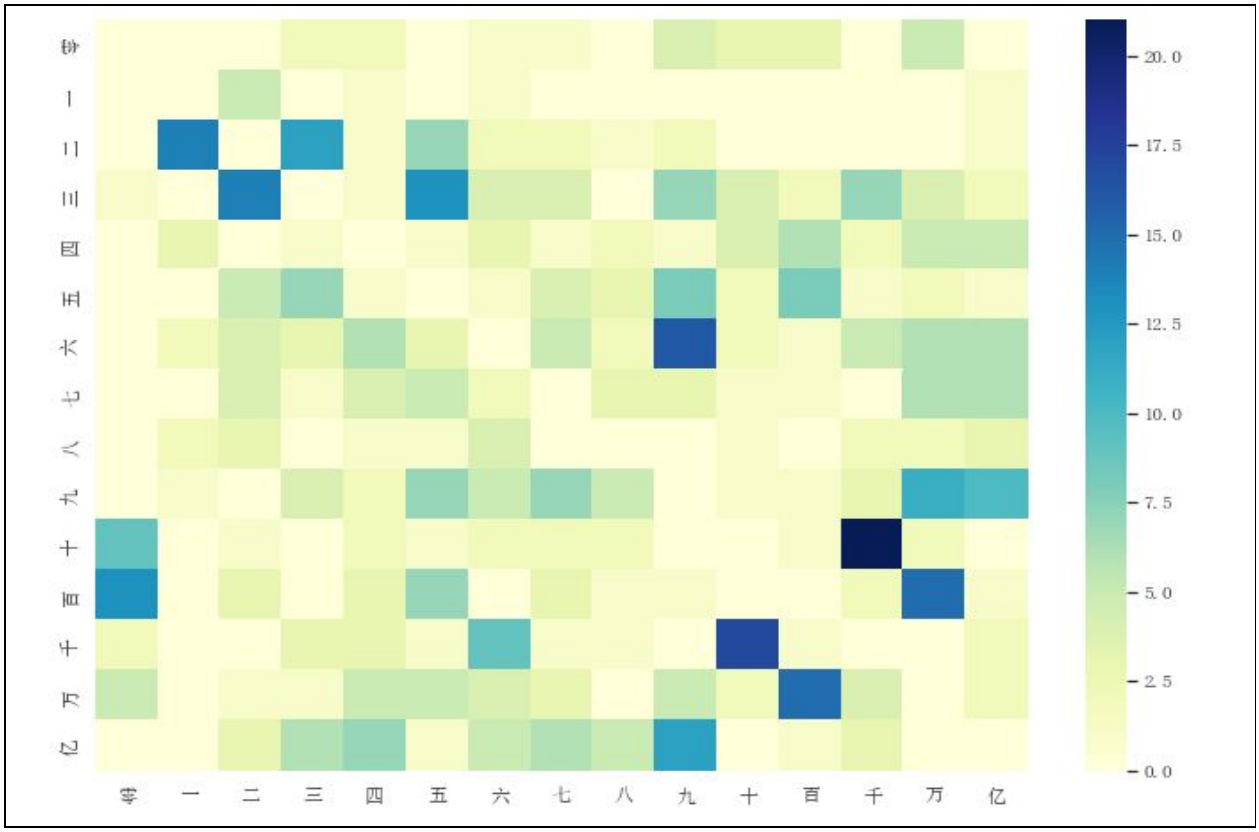

Figure 4: MLP for Chinese Numerals Low Resolution Heatmap (Diagonals Normalized)

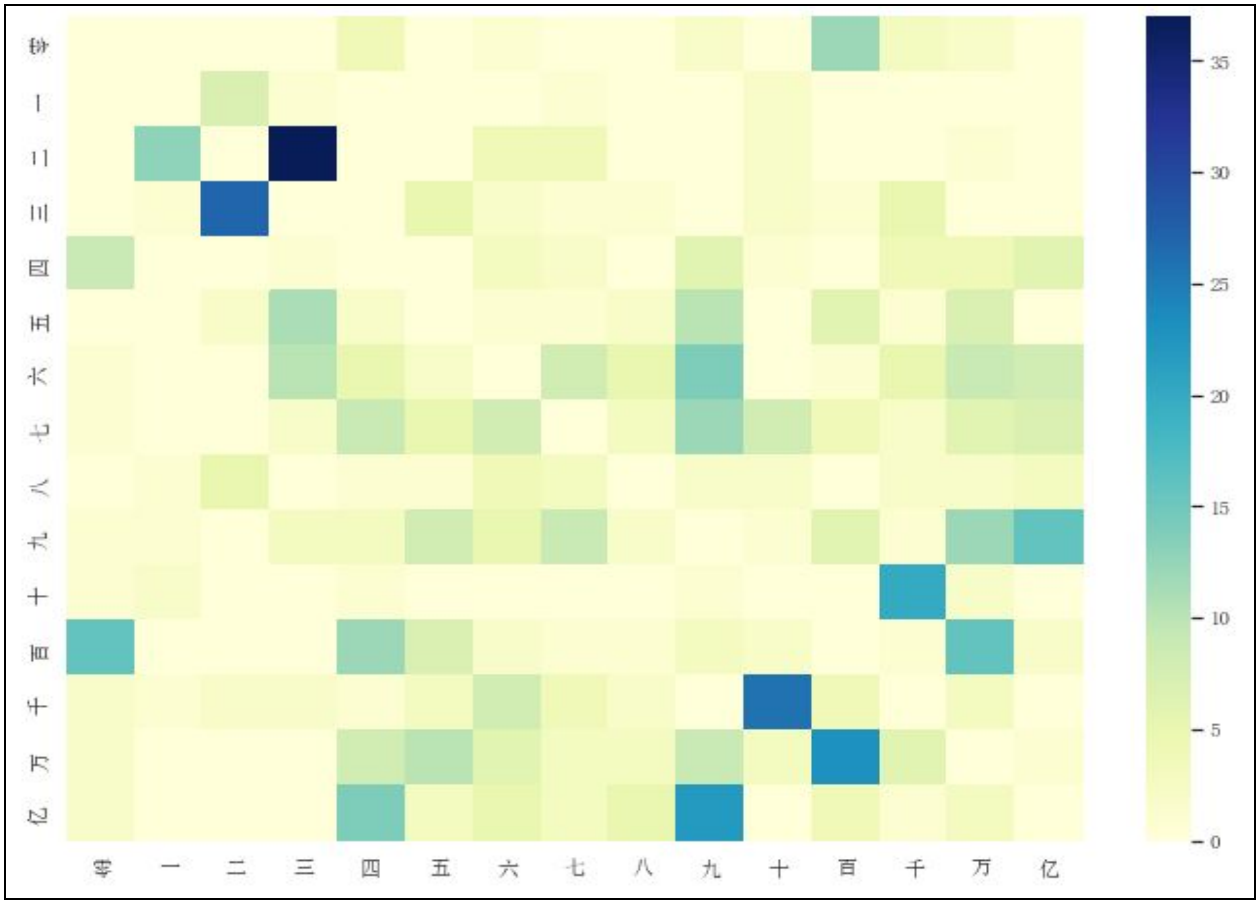

Figure 5: MLP for Chinese Numerals High Resolution Heatmap (Diagonals Normalized)

Python was used to visualize the confusion matrices of the MLP models for each of the datasets through the help of the Pandas, Numpy, Matplotlib, and Seaborn libraries [9-12]. Each of the figures above represent the confusion matrix in the form of a heatmap. The diagonals (correct labels) were set to 0 for easier visualization, as even for the worst performing MLP, were orders of magnitude larger than the incorrect labels (as would be expected for an accurate model). The rows correspond to the actual value, and the columns represent the predicted value of the model. Therefore, a darker blue in the heatmap indicates a greater number of false positives/false negatives, and allows to see visually which two outputs the model is likely confusing.

Interesting results emerge upon looking at the heatmaps, as the Arabic numerals had the output class pairs of 3-5 and 4-9 appear to get confused most often, in both directions. This makes sense, as there appears to be visually more in common with these pairs than say, the pair 1-8. For the low resolution Chinese dataset, 十-千 and 百-万 are the most common pairing mistakes, although there are a handful of other pairwise errors that are much more prominent compared to the Arabic heatmap as a whole (which makes sense given the higher error rate). In terms of analysis, the characters appear to be very visually similar in terms of structure, and would make sense to be "difficult" for an algorithm to learn and accurately predict. With regard to the high resolution Chinese dataset, 二-三 was the most prominent pair of errors, though the pairs 十-千 and 百-万 are also visually apparent. The inclusion of this additional pair 二-三 is interesting, and may relate back to the idea of the high resolution dataset introducing additional noise as compared to the lower resolution version.

Further Study

Exploration was done into looking at other MNIST-themed datasets, particularly other numeral datasets. However, there was relative scarcity for MNIST-themed datasets across different numeral systems, which was somewhat expected given the dominance of Western usage of Arabic numerals and the time expensive nature of assembling 100+ human participants and normalizing their handwriting. Other datasets found that were still MNIST-themed had other focuses for general image classification and were not specifically numeral, or even alphanumeric in nature, such as the Fashion MNIST dataset [13].

Further study could also be done into utilizing the context with which numbers appear. As mentioned previously, Chinese numbers follow a different pattern of construction than their Arabic counterparts, such as how numbers from 100 to 999 broadly follow the pattern of

X(hundred)X(ten)X. The characters for 100 and 10 are much more likely to be in between the characters representing 0 through 9. This consideration into context would likely help improve the performance of a classifier, although does increase in scope beyond the standard MNIST context. Domain specific knowledge of the topic is critical for effectively designing for internalization, as it can help improve algorithmic performance without necessarily increasing computational complexity.

Other considerations for further study would also include the combination of Arabic and non-Arabic numerals in a given application. In many places in China, Arabic and Chinese numerals coexist in modern writing and signage, depending on the specific use case of the numbers (Chinese numerals are used in more traditional settings, whereas Arabic numerals for many financial applications). This coexistence of the two systems further increases the size of the output classes for a potential classifier, and would further describe the need for systems to be resilient to a larger number of classes.

Conclusion

A comparative analysis of two different numeral systems was completed, and the results were generally inline with expectations. The standard MNIST dataset with only 10 output classes performed significantly better than both of the Chinese datasets, which had 15 output classes. Multilayer perceptrons proved to be the most robust algorithm tested, and performed at a high level of accuracy for the standard MNIST and reasonably well for both Chinese datasets. Interestingly, the higher resolution version of the Chinese dataset may have introduced additional noise into the dataset, and generally performed worse than its lower resolution counterpart. The usage of confusion matrix heatmaps for the MLPs led to interesting visual discoveries, including often confused pairs of numbers for both Arabic and Chinese numerals. This project opens the

door for further research in a number of related areas, and does show the need to consider internationalization when designing systems. Though MNIST may represent a toy situation for training models, it does show that the toys used as a testing playground should also be representative of the world they seek to emulate.

References

[1] "Localization vs. Internationalization." [Online]. Available: https://www.w3.org/International/questions/qa-i18n.en. [Accessed: 14-Dec-2020]

[2] "HTML ASCII Reference." [Online]. Available: https://www.w3schools.com/charsets/ref_html_ascii.asp. [Accessed: 15-Dec-2020]

[3] "Unicode Standard." [Online]. Available: http://www.unicode.org/standard/standard.html. [Accessed: 14-Dec-2020]

[4] "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges." [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 14-Dec-2020]

[5] K. Nazarpour and M. Chen, "Handwritten Chinese Numbers," 2017, doi: 10.17634/137930-3. [Online]. Available: https://data.ncl.ac.uk/articles/Handwritten_Chinese_Numbers/10280831. [Accessed: 14-Dec-2020]

[6] "Chinese MNIST." [Online]. Available: https://kaggle.com/gpreda/chinese-mnist. [Accessed: 14-Dec-2020]

[7] "Overview - Spark 2.4.2 Documentation." [Online]. Available: https://spark.apache.org/docs/2.4.2/. [Accessed: 14-Dec-2020]

[8] R. Gandhi, "Naive Bayes Classifier," Medium, 17-May-2018. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c. [Accessed: 14-Dec-2020]

[9] "pandas - Python Data Analysis Library." [Online]. Available: https://pandas.pydata.org/. [Accessed: 14-Dec-2020]

[10] "NumPy." [Online]. Available: https://numpy.org/. [Accessed: 14-Dec-2020]

[11] “Matplotlib: Python plotting — Matplotlib 3.3.3 documentation.” [Online]. Available: https://matplotlib.org/. [Accessed: 14-Dec-2020]

[12] “seaborn: statistical data visualization — seaborn 0.11.0 documentation.” [Online]. Available: https://seaborn.pydata.org/index.html. [Accessed: 14-Dec-2020]

[13] zalandoresearch/fashion-mnist. Zalando Research, 2020 [Online]. Available: https://github.com/zalandoresearch/fashion-mnist. [Accessed: 14-Dec-2020]