Authorship Identification for Handwritten Images

Albert (Xiang) Li, Katie (Ziqi) Chen, Emily (Hao) Ding, Kenneth Long*
University of Washington

Abstract

Distinguishing authorship through handwritten images is challenging. In this paper, both statistical models and machine learning models were applied to test whether authorship of given a handwritten image can be identified. Through the comparison of predictions, by splitting training and testing dataset, a significant difference between different people's writing style was found. The lowest accuracy score for prediction was Multiple Linear Regression model at about 35%; Logistic Regression model for comparing one writer to other writers achieved 80%; simple MLP Neural Networks for multiple author classification achieved around 66-68%; the highest accuracy for multiple author classification of 76% was obtained using a Convolutional Neural Network (CNN) model. While evaluated MLP predictions were found significantly different ($p<0.05$) from a uniform random distribution for all authors, $p \cong 0.049$ was observed even on a small data subset ($n=10$). In the future, more advanced models aimed at facilitating authentication and identification of users should be pursued.

Keywords: Handwritten image, writer identification, machine learning, deep learning, statistical model, linear regression, logistic regression, Multiple Linear Perceptron, Tensorflow model.

1 Introduction

The problem at hand originated in the field of forensic handwriting analysis. Forensic handwriting analysts have previously individually performed handwriting analysis for writer identification in a legal context, and legal professionals in the justice system have relied on these skills and abilities to aid in investigative and prosecutorial capacities [1]. The overarching purpose of this research is to determine whether a model can differentiate authorship based on handwritten images. If it can be shown that writer identification is reliable, an algorithm can be used to identify different persons, allowing handwriting algorithms to adapt to that particular writing style, thereby increasing recognition accuracy. This process is especially important in the area of online handwriting recognition. Other applications that use this technique include signature validation in the financial industry and government.

Using these techniques, the confidence with which statements can be made and the number of documents that can be processed can increase. Making these systems language independent can offer a wide variety of advances in ML itself, as well as specific handwriting applications. These types of technologies can also be utilized in the financial industry and government for signature validation, allowing early detection to suggest investigating possible fraud or additional verification measures [2].

One of the target users that could benefit from this project is law enforcement. When performing forensic handwriting analysis, the exact same task needs to be accomplished - identifying the author of a certain handwriting image. Models like this one can be used on existing suspects' handwriting images to train and classify whether the handwriting belongs to the suspect. Law enforcement can also use the accuracy score as a reference for whether to adopt forensic handwriting as strong evidence. There are no perfect forensic handwriting analysts and there is some dispute regarding whether handwriting analysis is sufficiently reliable to be admissible. Statistical models can provide judges with a complementary reference to decide whether to adopt the results of forensic handwriting analysts.

2 Related Work

Previous work has been done on classifying handwriting by different writers using various methods. Most of them are tend to be Neural Networks of varying complexities. Traditionally, this task was performed by a human by examining slant, stroke order, width, pressure, and other factors such as when the user picks up the writing utensil from the paper (creating extra loops, tails, et c.) [1]. Edge detecting convolutions have been used in attempts to pick up on these types of features in the past. By taking advantage of the statistical processing power of computers, Neural Networks have been used to replace the traditional hand-built rule and filter systems in areas like Computational Linguistics and Computer Vision. An attempt to use a simple Artificial Neural Network with 3 layers of 8 input nodes and 5 hidden layer nodes for this task, proved to be sufficient when there were a small number of target classifications [1]. Previous research has shown that Multivariate Multiple Linear Regressions are implemented by Neural Networks using Linear Perceptrons, where the weights of the perceptron edges correlate to the coefficients found in traditional linear regression [3]. It has also be shown that Multiple Linear Perceptron Neural Networks can perform Nonlinear Regression Analysis [3]. Modern work specifically targeting the highest accuracy uses Recurrent Neural Networks, but Convolutional Neural Networks have been known to perform well, often still requiring hand built 2D convolutions [4].

3 Methods

The methods used included matching provided target labels and formatting of the data for the analysis tools used in this research, traditional Statistical Linear Regression, Multiple Linear Perceptron Neural Network models, Convolutional Neural Networks, construction of uniformly random synthetically labeled subsets of the data for each writer to provide distributions that could match the null hypothesis and the Kolmogorov–Smirnov test for 2 samples. The python module Pillow was used to load the NIST Special Database 19 Second Edition PNG files and resize these images to 64 x 64 x 1 byte. The Statistical and ML methods used were those available in Statsmodels, SciKit Learn, and TensorFlow python modules. The python hashlib module implementation of SHA1 was used to create image hashes. The hash dictionary was stored in python pickle format on disk for reuse. Pandas and NumPy were used to load, parse and write csv files.

3.1 Data Processing

The dataset came from NIST Special Database 19. It consists of 5 zipped files with a total of 3,992,357 PNG images, each with 128 x 128 x 3 byte resolution. Individual characters were isolated for individual images for many organizations of this dataset. The dataset was collected from Bethesda High school students and Census Bureau employees in Suitland, Maryland. There were a total of 4044 writers represented in this dataset. Each writer had between 315 and 418 individual character writing samples. Every participant was asked to write all alphanumeric characters in a case-sensitive manner based on provided text on a provided form. The dataset was then classified by author (the by_write organization of the dataset per the description in SD19 User Guide Edition 2), meaning all images are labeled by filename with a unique Writer ID in folders also bearing the Writer ID. It was also organized by character type, character class, and full unprocessed sample forms. The by_field dataset was also used to provide matching character types and classes as possible image label targets. Personally identifiable information was removed from the dataset prior to its release to the public by the NIST.

In order to match Writer ID, Character Type and Character class, a hash dictionary was constructed using SHA1. Hashes were used as keys and values were by_field paths and labels. Regular Expressions (using the python regex module) were used to parse the by_field filename for the character type and class from the file path. There were 11 collisions observed between images provided in the dataset. Upon inspection of these images, they were pixel perfect matches with differing labels as provided by filepath. These images were not used in modelling or analysis. See Appendix for a list of these collisions.

Processing a huge dataset was one of the most challenging parts of this research. Several different approaches were tried to efficiently convert raw images into the desired format. One approach produced two Comma Separated Value formats from the data loaded in memory. In one format individual pixels in a 1D representation of the image were stored in individual columns of the csv. In the other format, a python list was converted to string format and stored as a cell value in an image column. It was more convenient to use the individual pixel csv with SciKit Learn and statsmodels, while TensorFlow preferred the array like image representation.

One of the approaches adopted for machine learning models was to first write a Python script to move 40,000 images into a single folder from different writers' folders. All the sample images were looped through to simplify the filename to only include Writer ID, making it more convenient to access labels. All these images were then read into a single Pandas dataframe. Each observation in the dataframe represents a single image. For each observation, an "image" column using a 2D NumPy array represents the grayscale pixel in the image, and a "writer" column indicates the writers' label for the image. This dataframe facilitated training the CNN model using Tensorflow.

3.2 Statistical Model

The statistical models used were Multiple Linear Regression and Logistic Regression. Multiple Linear Regression models the relationship between the continuous outcome variable and multiple independent variables. Logistic Regression models the relationship between the binary outcome (1 or 0) and multiple independent variables. In this dataset, the outcome variable is the authorship of each handwritten image, which is labeled as Writer ID. The independent variables are the grayscale intensity values of each pixel of each image.

As each 128 x 128 image was converted from 3 byte RGB to 1 byte grayscale pixels, and each pixel was stored as a column, there are 16386 pixel columns in each row of the dataset. Feature selection was used to select the best variables, which those that contributed the most to the outcome variable were selected. In particular, LASSO was used here to select 17 independent variables with alpha level 0.1. The correlation between these columns and the Writer ID is shown in Figure 1.

For Multiple Linear Regression, these 17 pixel columns were used as input to predict the Writer ID that corresponded to each image.

Before using Logistic Regression, the continuous outcome variable was converted to binary outcome. Here two cases were tested. The first case compared images of writing by the first writer (Writer ID 0000) to images of writing by the rest of writers. The outcome variable of 1 represented classification as the first writer, and 0 represented classification as any other writers. The second case was to compare images of writing by the second writer (Writer ID 0001) to images of writing by other writers. Similarly, the outcome variable of 1 represented classification as the second writer, and 0 represented classification as any other writers. The 17 selected pixel columns were then applied to the Logistic Regression model to predict whether the images were belonged to the first writer in the first case and the second writer in the second case.

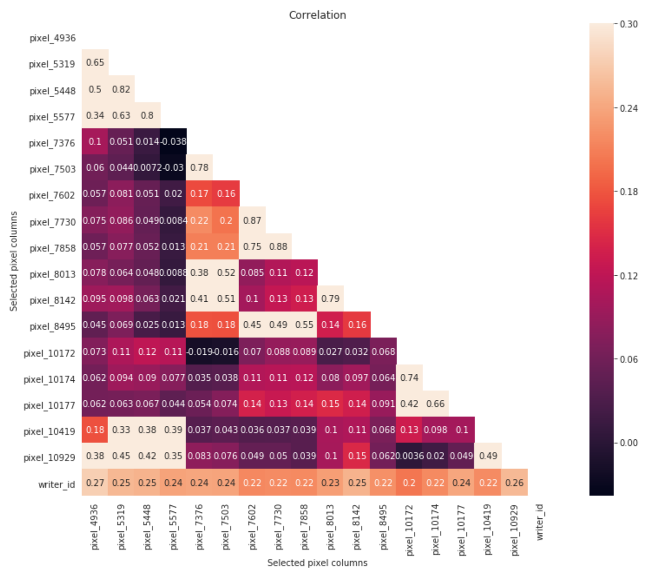


Figure 1: The correlation between best-selected pixel columns and the writer id

3.3 Machine Learning Model with MLP Classifier

SciKit Learn's Multiple Linear Perceptron Classifier was used with 3 hidden layers, the first two being 256 nodes, and the last 128. Additional layer sizes and numbers of layers were tested, including powers of 2 from 5 to 9. Higher and lower powers tended to provide very similar or worse results, and took much more time to train. The csv format including individual pixel columns was used for convenience due to Pandas and SciKit Learn data type inference commonly encountering issues when parsing the image csv format, and SciKit Learn's design relying on column layout for independent variables. Various subsets of the full dataset consisting of 5 and 10 writers were used to train and test the classifiers accuracy using a 0.75/0.25 train/test split. The default "adam" solver was used, although Stochastic Gradient Descent ("sgd") was attempted, it proved to perform less well and take about the same amount of time to train. The maximum number of iterations to attempt convergence were tested at 200, 20 and 10. The improvement between 200 and 20 was small enough given the time required for 200 iterations that 20 iterations were used. The same range of accuracy was achieved using both 200 and 20 iterations. To compare the predictions from the MLP Classifier against the Null Hypothesis, that "all writers write the same", subsets of the data were used to synthetically generate training and test data. Labels 0-9 were randomly uniformly assigned to writing samples from the same author per author and an MLP Classifier was trained with the same parameters on the synthetic data. Distributions of the predicted classifications given the Null Hypothesis were generated.

A 2D histogram of the prediction results for the model trained on 10 writers shows the distributions of classifications. Sample plot opacity is set to 0.01, meaning 100 correct predictions would give an opaque point. A reference is plotted for visual comparison.

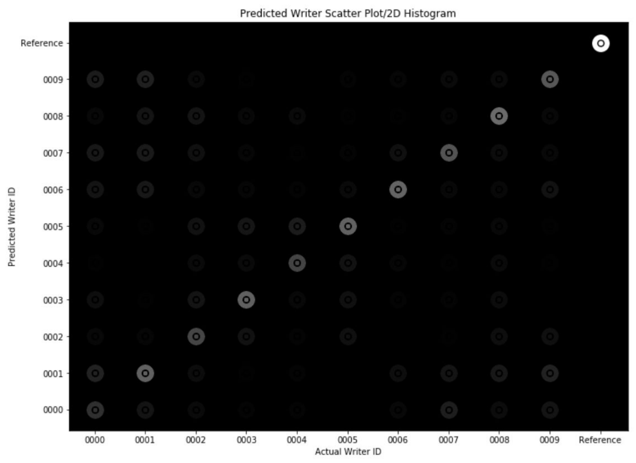


Figure 2: Actual Writer vs Predicted Classification 2D Histogram

3.4 Tensorflow Model with CNN Classifier

The second ML model is Tensorflow with **Convolutional Neural Network** (CNN). CNN is the current state-of-the-art model for image classification tasks. It applies a series of filters to extract the images' features, and the model will then use the features for classification. Our input is the preprocess data from previous part, which contains the grayscale of each pixel on the image. The label is the corresponding writer ID for each image.

Our model contains four main components, they are convolutional layers, pooling layers, flatten layer and dense layers. Convolutional layers apply a number of convolution filters to the pixel and create subregions that have similar features. For each subregion, the model produces an output representing the feature of the small subregion. For the convolutional layer, the model applied 32 3 * 3 filters, which means extracting 3 * 3 subregion to extract features. Using relu as the activation function is typically used for image classification.

Pooling layers are used to downsample the image data in order to decrease the running time for the model. Using the max pooling algorithm is the most commonly used pooling algorithm, to extract subregions of the feature map. The max pooling algorithm only keeps the maximum value and discards all other values, which really helps the model to running faster. Using a 2 * 2 filter could add a flatten layer in the model. Since the input passed into the model is a 2D numpy array, the model need to be flatten the 2D array into a 1D array to execute the classification.

The last component are dense layers. Every node in the dense layers is fully connected to every node in the previous layer. Dense layers performs image classification using the features extracted by convolutional layers and pooling layers. For our model with 5 writers, I used 5 neurons for each writer ID.

In order to utilize the model as much as possible, the model was tested different image sizes to find a good balance between accuracy and running speed. When the image size increased from 28 * 28 to 64 * 64, there is a **3%** increase in the accuracy score, but the running time almost doubled. Even for our small training data with 40,000 image, the increase in time is still consideratly high.

Additionally, after tested different hidden layer sizes, convolutional layers sizes, and dense layer sizes for better prediction, the model tested the accuracy for different image size and number of writers.

After some comparison, the model use a 64 * 64 image size. In this size, the image can still be clearly recognized and the running speed improved a lot compared with 128 * 128 size. A sparse_categorical_crossentropy as the loss function was applied since writers' IDs are categorical labels and there are more than two kinds of labels in the dataset.

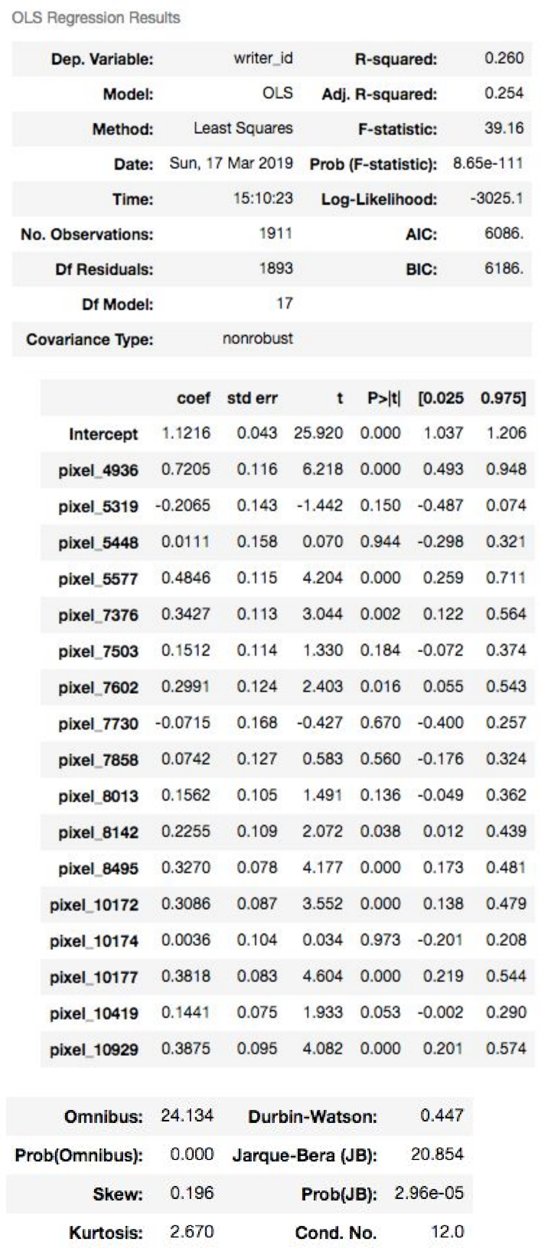
The model only used **10%** of the data as validation test and a batch size of 12 to run the test. In this case, using 12 images as a set of input can train the model instead of using a single image. The model used a 15 epochs for the model since the validation accuracy shows a decrease in the 14th epoch.

4 Results

Through multiple models, there are different accuracy as well as different implication of the results as following.

4.1 Multiple Linear Regression

With 17 best-chosen pixel columns as independent variables, the Writer ID of each image was predicted using Multiple Linear Regression. The accuracy score is 35%. The model summary is shown in Figure 3.



OLS Regression Results

Dep. Variable:	writer_id	R-squared:	0.260
Model:	OLS	Adj. R-squared:	0.254
Method:	Least Squares	F-statistic:	39.16
Date:	Sun, 17 Mar 2019	Prob (F-statistic):	8.65e-111
Time:	15:10:23	Log-Likelihood:	-3025.1
No. Observations:	1911	AIC:	6086.
Df Residuals:	1893	BIC:	6186.
Df Model:	17		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	1.1216	0.043	25.920	0.000	1.037	1.206
pixel_4936	0.7205	0.116	6.218	0.000	0.493	0.948
pixel_5319	-0.2065	0.143	-1.442	0.150	-0.487	0.074
pixel_5448	0.0111	0.158	0.070	0.944	-0.298	0.321
pixel_5577	0.4846	0.115	4.204	0.000	0.259	0.711
pixel_7376	0.3427	0.113	3.044	0.002	0.122	0.564
pixel_7503	0.1512	0.114	1.330	0.184	-0.072	0.374
pixel_7602	0.2991	0.124	2.403	0.016	0.055	0.543
pixel_7730	-0.0715	0.168	-0.427	0.670	-0.400	0.257
pixel_7858	0.0742	0.127	0.583	0.560	-0.176	0.324
pixel_8013	0.1562	0.105	1.491	0.136	-0.049	0.362
pixel_8142	0.2255	0.109	2.072	0.038	0.012	0.439
pixel_8495	0.3270	0.078	4.177	0.000	0.173	0.481
pixel_10172	0.3086	0.087	3.552	0.000	0.138	0.479
pixel_10174	0.0036	0.104	0.034	0.973	-0.201	0.208
pixel_10177	0.3818	0.083	4.604	0.000	0.219	0.544
pixel_10419	0.1441	0.075	1.933	0.053	-0.002	0.290
pixel_10929	0.3875	0.095	4.082	0.000	0.201	0.574

Omnibus:	24.134	Durbin-Watson:	0.447
Prob(Omnibus):	0.000	Jarque-Bera (JB):	20.854
Skew:	0.196	Prob(JB):	2.96e-05
Kurtosis:	2.670	Cond. No.	12.0

Figure 3: Multiple Linear Regression model summary

The summary showed that the R-squared is 26%, which means the model only explained 26% variability of the response variable. Low R-squared value indicates that the model did not fit the data well. Several of independent variables are not statistically significant since their p-value are greater than alpha level 0.05, which means the model did not gain enough information from having these variables.

Figure 4 compares the actual Writer ID and predicted Writer ID. It shows that for Writer ID 0000, the model predicted it wrong. For other writers, the model predicted partially right. With the low accuracy score, it was reasonable to conclude that Multiple Linear Regression model is not a good choice to estimate the classification outcome.

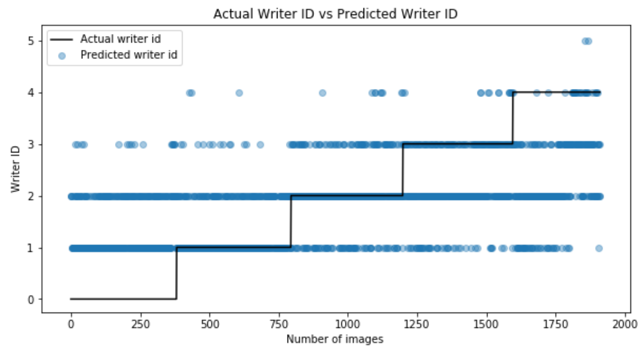
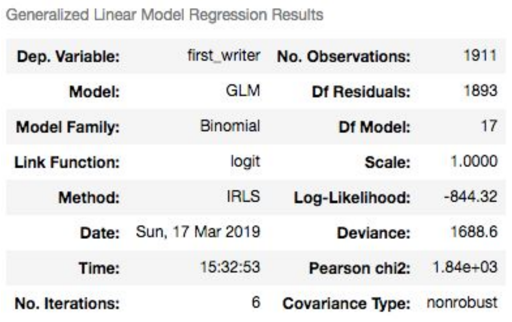


Figure 4: Actual Writer ID vs Predicted Writer ID

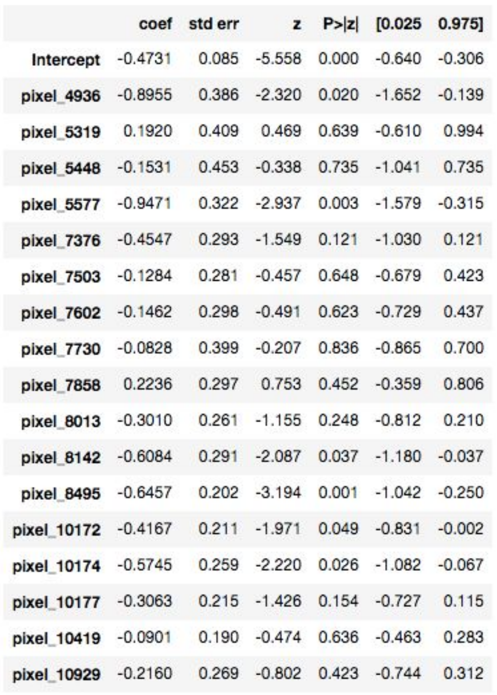
4.2 Logistic Regression

The model tested two cases with Logistic Regression. In the first case, the model compared images of the first writer (Writer ID 0000) to images written by other writers. With the same 17 pixel columns as described above as independent variables, the accuracy score for this case is 80%. The model summary for this case is shown in Figure 5.



Generalized Linear Model Regression Results

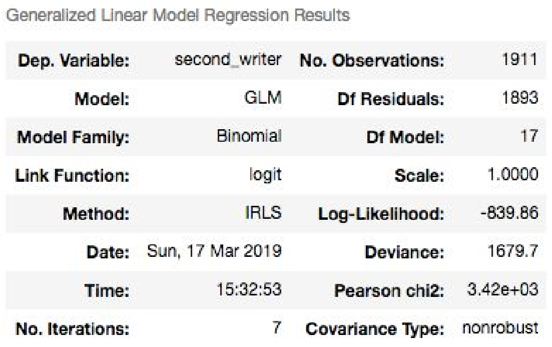
Dep. Variable:	first_writer	**No. Observations:**	1911
Model:	GLM	**Df Residuals:**	1893
Model Family:	Binomial	**Df Model:**	17
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-844.32
Date:	Sun, 17 Mar 2019	**Deviance:**	1688.6
Time:	15:32:53	**Pearson chi2:**	1.84e+03
No. Iterations:	6	**Covariance Type:**	nonrobust



	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.4731	0.085	-5.558	0.000	-0.640	-0.306
pixel_4936	-0.8955	0.386	-2.320	0.020	-1.652	-0.139
pixel_5319	0.1920	0.409	0.469	0.639	-0.610	0.994
pixel_5448	-0.1531	0.453	-0.338	0.735	-1.041	0.735
pixel_5577	-0.9471	0.322	-2.937	0.003	-1.579	-0.315
pixel_7376	-0.4547	0.293	-1.549	0.121	-1.030	0.121
pixel_7503	-0.1284	0.281	-0.457	0.648	-0.679	0.423
pixel_7602	-0.1462	0.298	-0.491	0.623	-0.729	0.437
pixel_7730	-0.0828	0.399	-0.207	0.836	-0.865	0.700
pixel_7858	0.2236	0.297	0.753	0.452	-0.359	0.806
pixel_8013	-0.3010	0.261	-1.155	0.248	-0.812	0.210
pixel_8142	-0.6084	0.291	-2.087	0.037	-1.180	-0.037
pixel_8495	-0.6457	0.202	-3.194	0.001	-1.042	-0.250
pixel_10172	-0.4167	0.211	-1.971	0.049	-0.831	-0.002
pixel_10174	-0.5745	0.259	-2.220	0.026	-1.082	-0.067
pixel_10177	-0.3063	0.215	-1.426	0.154	-0.727	0.115
pixel_10419	-0.0901	0.190	-0.474	0.636	-0.463	0.283
pixel_10929	-0.2160	0.269	-0.802	0.423	-0.744	0.312

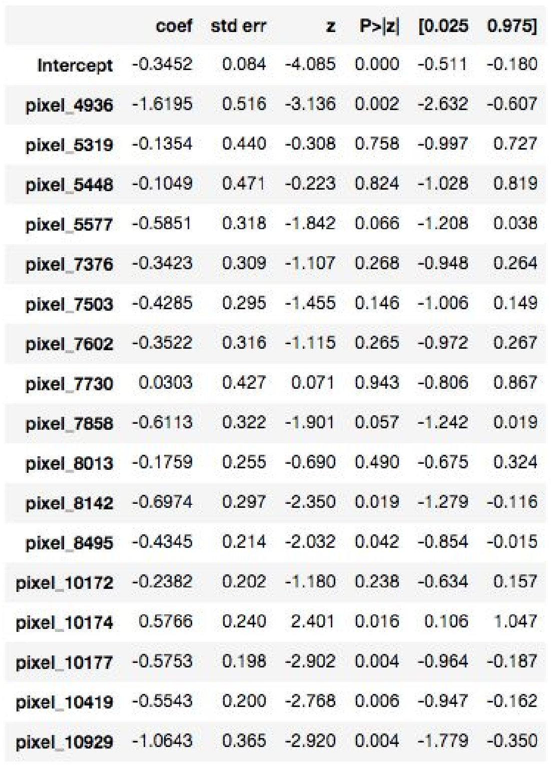
Figure 5: Logistic Regression summary with the first case

In the second case, the model compared images written by the second writer (Writer ID 0001) to images by other writers. With the same 17 pixel columns as independent variables, the accuracy score is 78%. The model summary is shown in the Figure 6.



Generalized Linear Model Regression Results

Dep. Variable:	second_writer	**No. Observations:**	1911
Model:	GLM	**Df Residuals:**	1893
Model Family:	Binomial	**Df Model:**	17
Link Function:	logit	**Scale:**	1.0000
Method:	IRLS	**Log-Likelihood:**	-839.86
Date:	Sun, 17 Mar 2019	**Deviance:**	1679.7
Time:	15:32:53	**Pearson chi2:**	3.42e+03
No. Iterations:	7	**Covariance Type:**	nonrobust



	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-0.3452	0.084	-4.085	0.000	-0.511	-0.180
pixel_4936	-1.6195	0.516	-3.136	0.002	-2.632	-0.607
pixel_5319	-0.1354	0.440	-0.308	0.758	-0.997	0.727
pixel_5448	-0.1049	0.471	-0.223	0.824	-1.028	0.819
pixel_5577	-0.5851	0.318	-1.842	0.066	-1.208	0.038
pixel_7376	-0.3423	0.309	-1.107	0.268	-0.948	0.264
pixel_7503	-0.4285	0.295	-1.455	0.146	-1.006	0.149
pixel_7602	-0.3522	0.316	-1.115	0.265	-0.972	0.267
pixel_7730	0.0303	0.427	0.071	0.943	-0.806	0.867
pixel_7858	-0.6113	0.322	-1.901	0.057	-1.242	0.019
pixel_8013	-0.1759	0.255	-0.690	0.490	-0.675	0.324
pixel_8142	-0.6974	0.297	-2.350	0.019	-1.279	-0.116
pixel_8495	-0.4345	0.214	-2.032	0.042	-0.854	-0.015
pixel_10172	-0.2382	0.202	-1.180	0.238	-0.634	0.157
pixel_10174	0.5766	0.240	2.401	0.016	0.106	1.047
pixel_10177	-0.5753	0.198	-2.902	0.004	-0.964	-0.187
pixel_10419	-0.5543	0.200	-2.768	0.006	-0.947	-0.162
pixel_10929	-1.0643	0.365	-2.920	0.004	-1.779	-0.350

Figure 6: Logistic Regression summary with the second case

Compared to Multiple Linear Regression, using Logistic Regression greatly improved the accuracy score. Therefore, it is reasonable to conclude that Logistic Regression is more suitable for classification outcome. However, there is one issue with this model. Figure 7 shows that when the model tried to predict whether images belong to the first writer (Writer ID 0000) or other writers, the model predicted 0 for all images. That means that when the model set the outcome variable to be 0 for all writers except the first writer, the model would mostly based on other writers' images to predict the Writer ID. This would affect the performance of predicting the images written by the first writer. Therefore, the model predicted all images written by the first writer to 0 instead of 1. This is the disadvantage of using Logistic Regression since the dataset has multiple writers.

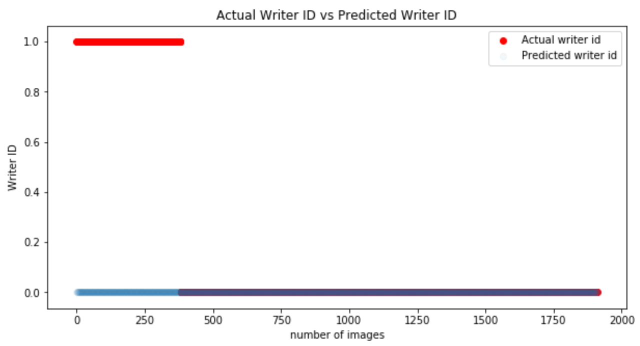


Figure 7: Actual Writer ID vs Predicted Writer ID

4.3 SciKit Learn with MLP Classifier

The 10 writer model achieved observed accuracies in the range 66-68% depending on parameters and train/test split. The distributions of the MLP Classifier classification prediction per writer by the model trained on all 10 writers writing samples were compared to the prediction distributions generated by the model trained on the synthetically generated data subsets per writer by writer. That is, the Writer ID 0000 classification distribution by the first model was compared to the distribution generated by the model trained on Writer ID 0000's writing samples with uniformly random labels and so on for each writer, using the Kolmogorov–Smirnov test for 2 samples. The two-sided Kolmogorov-Smirnov statistic for 2 samples can be adapted for discrete distributions, and in this special case, assuming the Null Hypothesis is true, this would be a uniform distribution. The derivative of the curve would be 0, meaning that the discrete distribution approximates a continuous distribution. The accuracy for the uniformly random label trained models ranged from 4% - 15%. The K-S test statistics are displayed in Table 1. The highest observed p-value was 0.049873 for Writer ID 0009, and the lowest D value was 0.1913 for the same author. The lowest p-value was 1.357 x 10^{-10} with a D value of 0.4834.

Writer ID	K - S D	p-value
0000	0.29649914089347074	0.00029600221897536656
0001	0.39871794871794874	2.125603393134146e-08
0002	0.4732060834949474	2.0072078639186591e-10
0003	0.4834408602150537	1.3565777717688764e-10
0004	0.41043344840813195	6.223625890586029e-06
0005	0.42121212121212126	1.4479523882384366e-07
0006	0.32244404113732605	0.00010774158302194342
0007	0.48035043804755945	9.68145064064899e-10
0008	0.28179271708683473	0.00020588241843005027
0009	0.19138755980861244	0.04987295474230525

Table 1. K - S statistics comparing Null Hypothesis and observed distributions

Considering the sample sizes for all of these subsets are over 300, a critical value of D for 300 was calculated using 1.36/($300^{½}$) ≅ 0.0785. All K-S D are over this threshold and all p-values are below 0.05.

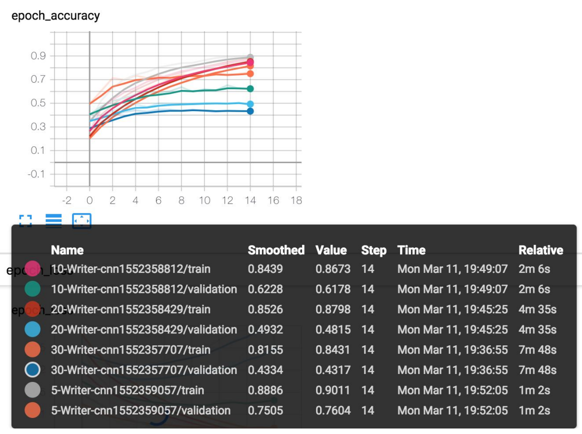
The probabilities for a given sample image were also observed, an example is provided in Table 2.

Author	Probability
0000	0.00000001
0001	0.00000000
0002	0.00070024
0003	0.00000000
0004	0.99918116
0005	0.00011208
0006	0.00000000
0007	0.00000000
0008	0.00000651
0009	0.00000000

Table 2. Probabilities for classification of a given example image

4.4 CNN with Tensorflow

With all the methods and adjustments described above, the CNN model was successfully built with different sizes of input to predict the authorship of a handwritten image. The highest accuracy score is 76%.



	Name	Smoothed	Value	Step	Time	Relative
●	10-Writer-cnn1552358812/train	0.8439	0.8673	14	Mon Mar 11, 19:49:07	2m 6s
●	10-Writer-cnn1552358812/validation	0.6228	0.6178	14	Mon Mar 11, 19:49:07	2m 6s
●	20-Writer-cnn1552358429/train	0.8526	0.8798	14	Mon Mar 11, 19:45:25	4m 35s
●	20-Writer-cnn1552358429/validation	0.4932	0.4815	14	Mon Mar 11, 19:45:25	4m 35s
●	30-Writer-cnn1552357707/train	0.8155	0.8431	14	Mon Mar 11, 19:36:55	7m 48s
●	30-Writer-cnn1552357707/validation	0.4334	0.4317	14	Mon Mar 11, 19:36:55	7m 48s
●	5-Writer-cnn1552359057/train	0.8886	0.9011	14	Mon Mar 11, 19:52:05	1m 2s
●	5-Writer-cnn1552359057/validation	0.7505	0.7604	14	Mon Mar 11, 19:52:05	1m 2s

Figure 8: Epoch accuracy scores with different writers in Tensorflow dashboard

The image above shows the training accuracy and validation accuracy for different input sizes. There is a clear trend that as the number of writers in the input increase, the accuracy decreased. When the model only have 5 writers in the input, it can achieve 76% validation accuracy, but only 43% validation accuracy for 30 writers. It shows how complicated for the model to find all the differences in terms of writing style.

5 Discussion

As observed with the MLP model, the likelihood of observing the prediction distribution seen when training the classifier on 10 writers, if the null hypothesis were true, that all writers write "the same", in the way a single writer does, is significantly different (α=0.05) from what was observed. Therefore, the null hypothesis that there is no difference in writer writing style should be rejected. In many cases, this p-value is so low that the conclusion that must be drawn is that it is beyond chance, but notably, there was one writer whose p-value approached 0.05. This finding is too close to recommend use for applications like law-enforcement using only the models evaluated.

The training accuracy is higher than the validation accuracy for any number of inputs, which is normal for machine learning. The training accuracy was able to increase as long as the number of epochs are increased. This shows that computer model is really good at memorizing everything and try to "cheat" in the prediction. However, for validation accuracy, it basically maintains at the same level from the 8th epoch. Moreover, the accuracy number even decreased when at around 12th to 14th epoch. This is why it is important to run multiple models with different parameters to test validation accuracy.

Through the project, valuable insights about machine learning were gained. First, machine learning is an iterative process and it is important to build a simple model first, and then improve it iteratively. Finding methods to improve the process was challenging due to the limitations of time and existing knowledge. Taking this into consideration, the first statistical model could provide clues that could improve other models.

It is also important to keep the evaluation standard consistent. During modelling, different approaches were used to calculate accuracy. Since the standards are different, it is difficult to directly compare performance for different models. Using a simple, expressive, and consistent method to evaluate the results would be the best approach.

6 Future Work

When working with datasets containing millions of images or more, the balance between accuracy and running speed becomes a major concern. Finding even one close call of 1 in 10 that was very close to a uniformly random distribution indicates the high probability of misclassification for larger sets. Moreover, image recognition models also have countless parameters. In order to improve efficiency, in the future, applying transfer learning could greatly benefit modeling.

Transfer learning is a technique that improves the overall efficiency of the model by taking a piece of a model that has already been trained on a related task and reusing it in a new model. With the power of transfer learning, a model is able to borrow from another model and still be trained with new data on top of it. This strategy can help save a lot of time and computing power.

A standard measurement across different models is needed. Each of these three models was built using different tools, so it is hard to compare accuracy results directly.

In addition, separating the training set and validation set in accordance with the target users could of value. The validation set allows quick evaluation of the performance of the model and guides tuning of hyperparameters.

Currently, the model separate dataset using built-in function to randomly separate the training set and validation set. In the future, the real use case scenarios should be accounted for when choosing training set and validation set.

References

[1] Sargur N. Srihari,1 Ph.D.; Sung-Hyuk Cha,2 Ph.D.; Hina Arora,3 M.E.; and Sangjik Lee,4 M.S. Individuality of Handwriting. In *Journal of Forensic Science*, Volume 47, No. 4, July 2002.

[2] Srihari, S. N., Xu, A., & Kalera, M. K. (n.d.). Learning Strategies and Classification Methods for Off-line Signature Verification. Retrieved from https://cedar.buffalo.edu/~srihari/papers/IWFHR2004.pdf

[3] Larrie V. Hutton. USING STATISTICS TO ASSESS THE PERFORMANCE OF NEURAL NETWORK CLASSIFIERS. In *Johns Hopkins APL Technical Diges*t, Volume 13. Number 2, pages 291-299. The Johns Hopkins University Applied Physics Laboratory LLC, 1992.

[4] Xu-Yao Zhang, Guo-Sen Xie, Cheng-Lin Liu, and Yoshua Bengio. End-to-End Online Writer Identification With Recurrent Neural Network. In *IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS*, volume 47, no. 2, pages 285-292, APRIL 2017.

Appendix

Supplementary Materials

Code for this project is available at https://github.com/albertli354/Final-Project-Info370.

Observed Image Hash Collisions in NIST Special Database 19 by_field archive

by_field/hsf_0/lower/63/63_00132.png conflicted with by_field/hsf_0/upper/43/43_00119.png

by_field/hsf_0/lower/66/66_00125.png conflicted with by_field/hsf_0/upper/46/46_00118.png

by_field/hsf_0/lower/7a/7A_00128.png conflicted with by_field/hsf_0/upper/5a/5A_00123.png

by_field/hsf_1/const/65/65_01702.png conflicted with by_field/hsf_1/const/6e/6E_00678.png

by_field/hsf_1/const/6e/6E_00925.png conflicted with by_field/hsf_1/const/71/71_00076.png

by_field/hsf_1/const/56/56_00450.png conflicted with by_field/hsf_1/const/72/72_02464.png

by_field/hsf_2/digit/31/31_05710.png conflicted with by_field/hsf_2/digit/31/31_05713.png

by_field/hsf_2/lower/6a/6A_00292.png conflicted with by_field/hsf_2/upper/4a/4A_00320.png

by_field/hsf_4/digit/30/30_00000.png conflicted with by_field/hsf_4/digit/30/30_00001.png

by_field/hsf_4/lower/72/72_00000.png conflicted with by_field/hsf_4/lower/72/72_00001.png

by_field/hsf_4/upper/44/44_00000.png conflicted with by_field/hsf_4/upper/44/44_00001.png