Image Classification in Practice: High Efficiency and Performance From Singular Value Decomposition

Weiyue Li
Halıcıoğlu Data Science Institute
University of California San Diego
San Diego, CA 92093
wel019@ucsd.edu

Abstract

In this project, we used NumPy implemented Logistic Regression with Stochastic Gradient Descent to classify Japanese Hiragana hand writings from the KMNIST dataset. We then used **Singular Value Decomposition** to reduce the size of images for the goals of decreasing memory allocations and hopefully increasing the performance of the model. After applying Singular Value Decomposition, we were able to achieve 99% of testing accuracy on classifying お and ま with 40% less memory allocation on the original images as well as around 87% of testing accuracy on classifying す and ま with 40% less memory allocation on the original images.

1 Introduction

We trained a logistic regression classifier to identify Japanese Hiragana hand writing. We utilized Python library `NumPy` for model implementation and `Matplotlib` for training visualization. By training our models on a training set of size 60,000, we obtained a simple neural network that classifies Hiragana hand writing into the corresponding labels with high accuracy (above 98% for お and ま and above 85% for す and ま).
However, since we trained our network on a 2020 M1 MacBook Pro with only 16 GB of memory, it is essential to allocate the memory efficiently. This would be an issue when we train neural networks on larger image datasets (such as COCO). To solve this issue, we chose to use a linear algebra technique named Singular Value Decomposition. By applying singular Value Decomposition to the images, we were able to compress the size of the matrices that represent the images while keeping the most important features so that we were able to save memory spaces on local devices.
After normalizing (z-score or min-max) and one-hot encoding, we fed the data into the neural network, where weight vectors are simulated via stochastic gradient descent (SGD). We were able to achieve 99% of testing accuracy on お and ま with 40% less memory allocation on the original images as well as 87% of testing accuracy on す and ま with 40% less memory allocation on the original images.

2 Dataset

2.1 Dataset introduction

The Hiragana handwriting dataset we used in this project is KMNIST(Kuzushiji-MNIST). KMNIST have 10 classes, which represent 10 different Hiragana characters (see Figure 1). The images are sizes of 28×28, and are reshaped into 1 dimension array of length 784 for input purposes.
There are 6000 samples per class in the training set, 60000 samples in total, and 1000 samples per class in the test set, 10000 samples in total.

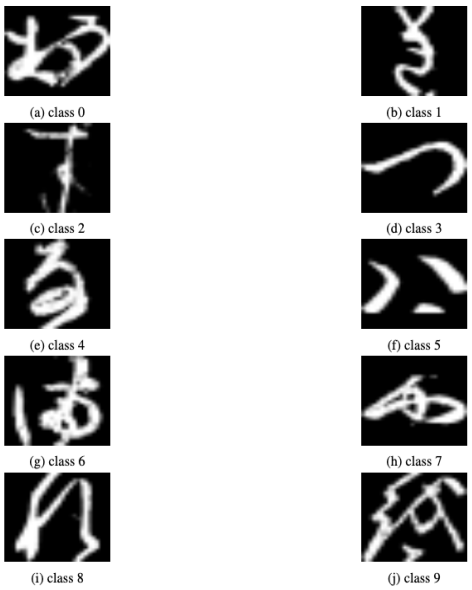

Figure 1: Sample figure caption. Source: KMNIST Dataset website

2.2 Data preprocessing

The labels for each sample will be one-hot encoded as vectors of length 10, in which each index indicates whether this sample belongs to the corresponding class or not, with 1 represents true and 0 represents false. The patterns are normalized so that tasks can be performed on a common scale without distorting differences in the range of values or lose information. Two normalization methods, z-score normalization and min-max normalization are implemented, and can be altered through change in hyperparameters. The z-score normalization is calculated with the formula $f(x) = \frac{(x-\mu)}{\sigma}$, where μ is mean of x and σ is standard deviation of x. The min-max normalization is computed by the formula $f(x) = \frac{x-min(x)}{max(x)-min(x)}$. x stands for the data to normalize. Afterwards, before starting the training process, to augment the data, a bias w_0 was added to the samples in the form of value 1 at the beginning of the array, thus changing the dimension of each individual sample input from 784 to 785.
During the training process, in order to to estimate the model's performance on unseen data before testing as well as prevent over-fitting, a k-fold ($k = 10$) cross validation method was used to split a portion of the training data into validation data at each fold.

3 Singular Value Decomposition

Singular Value Decomposition is a linear algebra technique that breaks down a real $m \times n$ matrix A into the form of Equation 1, where U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ diagonal matrix, and V is an $n \times n$ orthogonal matrix.

$$A = U\Sigma V^T \tag{1}$$

The main benefit of Singular Value Decomposition is that we can apply this technique to matrix of any size, which means that the matrix representations of the images do not have to be squared matrices.
There are two ways of obtaining singular values of such matrix A. It could be obtained by getting the diagonal entries of the decomposed Σ, or it could be obtained by taking the square-root of the eigenvalues of $A^T A$.

3.1 Decreasing Image Size

A large image will have a large matrix representation. What SVD could help is that we can extract the largest singular values with their corresponding eigenvectors and rewrite the matrix into a smaller size. This is because the larger the singular values, the most important information they have. By abandoning the smaller singular value, we could also potentially eliminate noise in the pictures.
For example, suppose we have an $m \times n$ matrix $A = U\Sigma V^T$, we would have m singular values in the Σ. If we only take the largest k singular values (locate in the top left), k left most columns in U and V, the approximated matrix of $A \approx U_{m\times k}\Sigma_{k\times k}V^T_{k\times n}$ would be resized from $m \times n$ elements into $k \times (1 + m + n)$ elements. See visualization in Figure 2.This could potentially save us $m \times n - k \times (1 + m + n)$ elements to approximate the same image!
Note: in the below sections, the k is defined as Singular Value Limit.

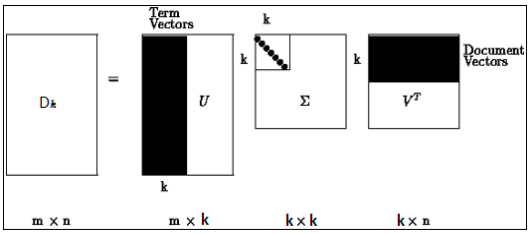

Figure 2: Sample figure caption. Source: ResearchGate

4 Logistic regression

Logistic Regression is used for binary classification. The activation function selected was a sigmoid function (Equation 2), where the range of the sigmoid function is $(0, 1)$.

$$g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T\mathbf{x})} \tag{2}$$

For the convenience of binary classification, we call one class as class 0 and another class as class 1. It is useful to know that the probability of an image x being classified to class 1 is the prediction of the activation function in Equation 3. This also imply the probability of being classified into class 0 in Equation 4.

$$P(C_1|\mathbf{x}) = \frac{1}{1 + exp(-\mathbf{w}^T\mathbf{x})} = y \tag{3}$$

$$P(C_0|\mathbf{x}) = 1 - y \tag{4}$$

For logistic regression, we will be using binary cross-entropy cost function (Equation 5) to calculate our loss, where $t^n \in \{0, 1\}$ is the label for image n, and we will minimize it via Stochastic Gradient Descent. The gradient for this loss function is shown in Equation 6.

$$E(w) = -\sum_{n=1}^{N} \{t^n \ln(y^n) + (1 - t^n) \ln(1 - y^n)\}. \tag{5}$$

$$\frac{\partial E(w)}{\partial w_j} = -\sum_{n=1}^{N}(t^n - y^n)x^n_j \tag{6}$$

5 Results

In the below section, we showcased the performance of the simple logistic regression neural network for classifying Japanese Hiragana characters into お and ま and into す and ま , as well as the memory and runtime efficiency by applying **singular value decomposition**(SVD).
To simplify the Japanese Hiragana character expressions, we defined class 0 as お , class 2 as す , and class 6 as ま . The labels will be used in the below sections.

5.1 Results for Class 0 vs Class 6

The Table 1 below shows the results of classifying into class 0 and class 6 with different hyperparameters when setting number of epochs to 100 and k-fold to 10. In particular, the hyperparameters that yields the highest SVD applied validation accuracy (99.1%) are bolded. Under this setting, we are able to achieve 100.1% of the performance of the same model without applying SVD on the testing dataset while saving approximately 40% of memory allocation for the images!
Figure 3 compares the training and validation loss under 0.1 learning rate, batch size of 32, z-score normalization, and singular value limit of 8 over 100 epochs under the difference of if SVD was applied.
For fine-tuning, we applied three different choices of learning rate 0.1, 0.01, 0.001 combined with three choices of batch size 32, 64, 128, two choices of normalization z-score, min-max, and 4 choices of singular value limits 3, 5, 8, 13. During the training and validation process, we noticed that the smaller the batch size, the better the performance with SVD. The singular value limit that compresses images to 60% of their original sizes has the best performance. On the other hand, the larger the learning rate with z-score normalization, the better the result on these two classes when applying SVD. When combining these findings, the model with SVD could perform better than the model without SVD on the testing dataset when saving a lot of memory spaces!

Table 1: Model Results on Class 0 vs Class 6

Learning Rate	**Batch Size**	**Normalization**	**Singular Value Limit (Percent Size)**	**SVD Validation Accuracy**	**SVD Testing Accuracy**	**Percent Testing Accuracy**
0.001	32	min-max	3 (0.218)	0.981	0.982	0.994
0.001	64	min-max	3 (0.218)	0.979	0.982	0.995
0.001	128	min-max	3 (0.218)	0.978	0.980	0.995
0.1	32	min-max	3 (0.218)	0.986	0.988	1.000
0.01	32	min-max	3 (0.218)	0.976	0.982	0.994
0.1	32	z-score	3 (0.218)	0.989	0.985	0.997
0.1	32	z-score	5 (0.364)	0.988	0.990	1.005
0.1	32	z-score	8 (0.582)	**0.991**	0.989	1.001
0.1	32	z-score	13 (0.945)	0.988	0.986	0.999

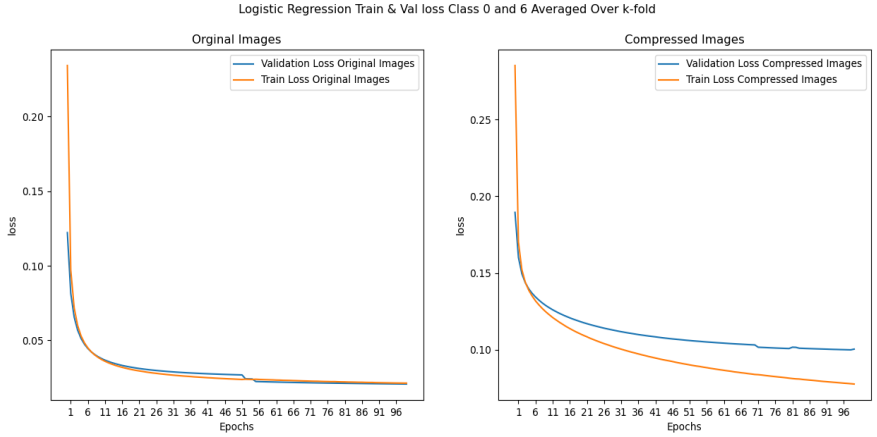

Figure 3: Training and Validation Loss of the Final Result for Logistic Regression Between Class 0 and Class 6 comparing with or without SVD (lr=0.1, batch-size=32)

5.2 Results for Class 2 vs Class 6

The Table 2 below shows the results of classifying into class 2 and class 6 with different hyperparameters when setting number of epochs to 100 and k-fold to 10. In particular, the hyperparameters that yields the highest SVD applied validation accuracy (88.8%) are bolded. Under this setting, we are able to achieve 98.5% of the performance of the same model without applying SVD on the testing dataset while saving approximately 40% of memory allocation.
Figure 4 compares the training and validation loss under 0.1 learning rate, batch size of 32, z-score normalization, and singular value limit of 8 over 100 epochs under the difference of if SVD was applied.
For fine-tuning, we applied three different choices of learning rate 0.1, 0.01, 0.001 combined with three choices of batch size 32, 64, 128, two choices of normalization z-score, min-max, and 4 choices of singular value limits 3, 5, 8, 13. During the training and validation process, we noticed that the smaller the batch size, the better the performance with SVD. The singular value limit that compresses images to 60% of their original sizes has the best performance. On the other hand, the larger the learning rate with z-score normalization, the better the result on these two classes when applying SVD.

Table 2: Model Results on Class 2 vs Class 6

Learning Rate	Batch Size	Normalization	Singular Value Limit (Percent Size)	SVD Validation Accuracy	SVD Testing Accuracy	Percent Testing Accuracy
0.001	32	min-max	3 (0.218)	0.829	0.862	0.995
0.001	64	min-max	3 (0.218)	0.830	0.851	0.987
0.001	128	min-max	3 (0.218)	0.810	0.837	0.984
0.1	32	min-max	3 (0.218)	0.851	0.869	0.982
0.01	32	min-max	3 (0.218)	0.851	0.868	0.987
0.1	32	z-score	3 (0.218)	0.870	0.864	1.003
0.1	32	z-score	5 (0.364)	0.872	0.861	0.981
0.1	32	z-score	8 (0.582)	**0.888**	0.864	0.985
0.1	32	z-score	13 (0.945)	0.875	0.860	0.978

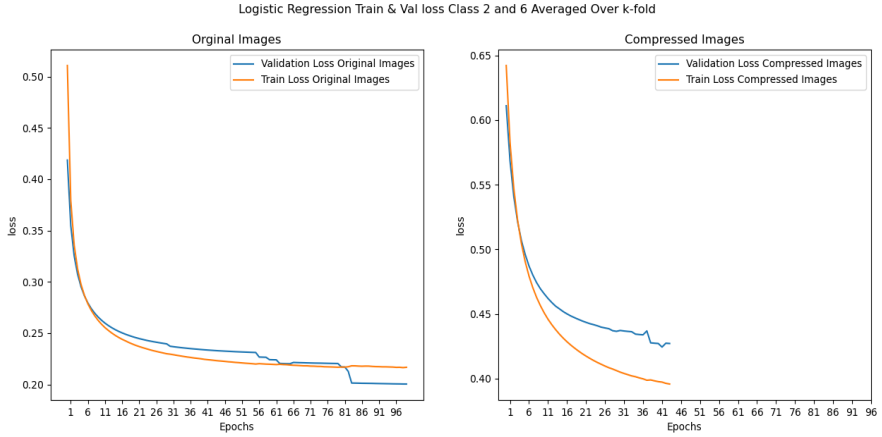

Figure 4: Training and Validation Loss of the Final Result for Logistic Regression Between Class 2 and Class 6 comparing with or without SVD (lr=0.1, batch-size=32)

5.3 Discussion

Although we showed a sample from each class in Section 2.1, one sample may not be representative. We extracted samples from class 0, 2, and 6 from KMNIST Github repository, and they are shown in Figure 5, 6, and 7. According to the images, from the perspective of a human being, class 0 characters are very different from class 2 and 6 in terms of shape and size. However, class 2 and class 6, even if different in original label, can be written in a similar way. Both of them have vertical strokes as the main character structure, and most of the horizontal strokes are distributed in a similar manner. We believe that due to this similarity in samples, the weight generated based on them tend to show homogeneous features. As a result, the model cannot be as accurate as it was on classification between class 0 and class 6 when decide on the features.

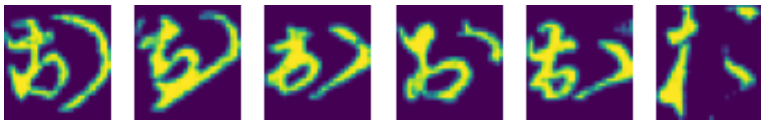

Figure 5: More Samples of Class 0

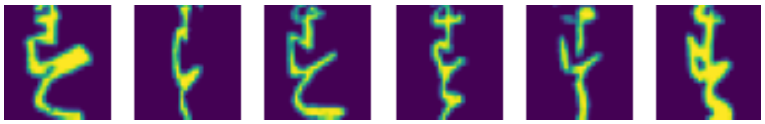

Figure 6: More Samples of Class 2

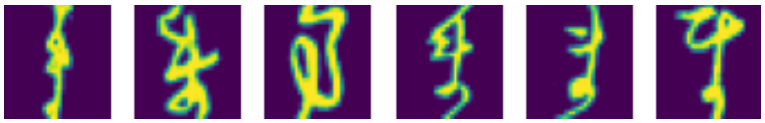

Figure 7: More Samples of Class 6

We will then show the same sample images compressed by SVD using different number of singular value limit 3, 8, 13. Figure 8, 9, and 10 are compressed images of Figure 5; Figure 11, 12, and 13 are compressed images of Figure 6; Figure 14, 15, and 16 are compressed images of Figure 7.
We can clearly see that when the singular value limit is 8, the compressed images has already displayed human detectable features in their graph for all three classes. These features contains relatively the "most important" information in those original images while maintaining at a smaller size to save computer memory.

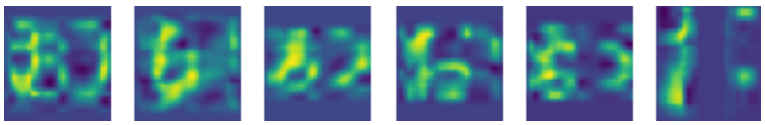

Figure 8: Samples of Class 0 with Singular Value Limit of 3

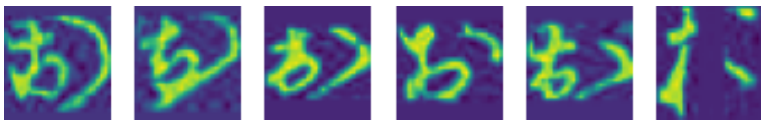

Figure 9: Samples of Class 0 with Singular Value Limit of 8

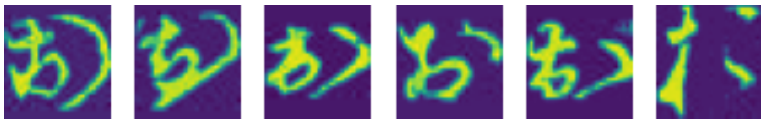

Figure 10: Samples of Class 0 with Singular Value Limit of 13

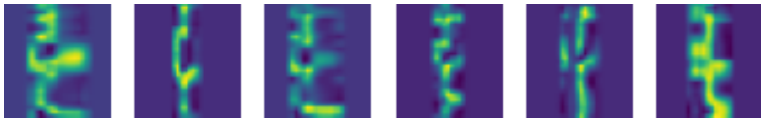

Figure 11: Samples of Class 2 with Singular Value Limit of 3

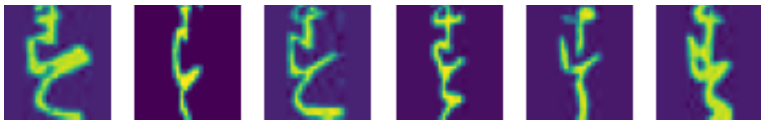

Figure 12: Samples of Class 2 with Singular Value Limit of 8

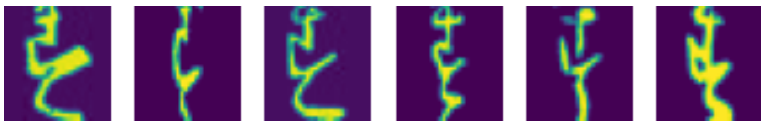

Figure 13: Samples of Class 2 with Singular Value Limit of 13

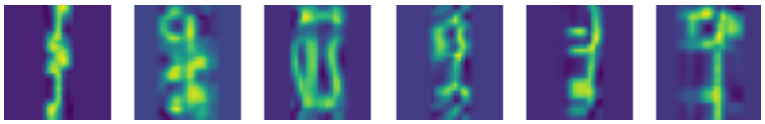

Figure 14: Samples of Class 6 with Singular Value Limit of 3

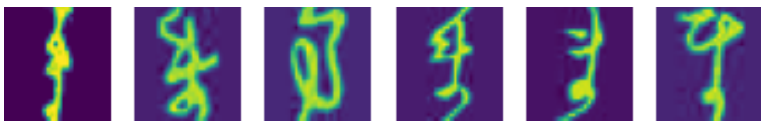

Figure 15: Samples of Class 6 with Singular Value Limit of 8

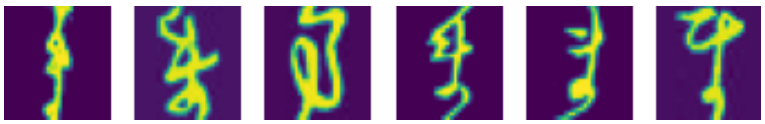

Figure 16: Samples of Class 6 with Singular Value Limit of 13

6 Conclusion

In this project, we trained two NumPy based logistic regression classifier on KMNIST Hiragana handwriting data. The process of training is the same while one classifier's dataset were decomposed by Singular Value Decomposition. After the data is augmented and one-hot encoded, we put the data through a cross validation training process, and split the data into 90% training set and 10% validation set on each fold. Weights are updated throughout the fixed 100 epochs using Stochastic gradient descent. During gradient descent, cross entropy loss between prediction value and observed value is calculated and used to update the weight vectors. After fine-tuning our model with different learning rates, batch sizes, normalization methods, and singular value limit, we finally achieved accuracy around 99% in the logistic regression classification between SVD processed class 0 and class 6, and around 87% in the logistic regression classification between SVD processed class 2 and class 6.
Surprisingly, we found out that SVD processed images who have smaller sizes could achieve better performance than the original bigger and more detailed images! The best performance on the SVD processed KMNIST dataset roughly when we take 8 components from the decomposed matrices.
Some possible improvement on this project could be more hyperparameter fine-tuning. We can try more combinations of different hyperparameters so that we might be able to achieve higher accuracy from the SVD processed images while requiring less memory spaces on the local device for training data.

7 Reference

[1] Clanuwat, Tarin, et al. "Deep learning for classical japanese literature." arXiv preprint arXiv:1812.01718 (2018).

[2] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[3] L. Kong, W. Li and Y. Li, "Image Classification in Practice: Implementing Logistic and Softmax Regression Using Numpy." (2022)

[4] Elizabeth A. Compton and Stacey L. Ernstberger, PhD, "Singular Value Decomposition: Applications to Image Processing." (2020)

[5] "KMNIST Dataset" (created by CODH), adapted from "Kuzushiji Dataset" (created by NIJL and others), doi:10.20676/00000341