Implications of Prototyping in Image Machine Learning

Ryosuke Oguchi **Nicole Ou** **Rachel Wang**

Abstract

The purpose of this project is to view Image Machine Learning as a data prototyping problem. It is generally accepted in ML that more training data will lead to better predictions on an unseen dataset. (Although there are some exceptions based on algorithm) However, with more training data, there is an issue of data complexity and increased training time for learning-based algorithms. To remedy the issues of having n-based time and space complexity, we will explore the validity of using training prototypes to determine if you can reduce the training data space in a fleshed-out method to maintain accuracy on unseen datasets.

1 Introduction

To explore the possibilities of prototyping, we will look at three classic image machine learning datasets to examine the validity of prototyping. The reason why we would like to view prototyping as a valid method to incorporate into machine learning is because of the implications of data space. Of course, to maximize the probability of classification accuracy, it would be better to have more examples to train your machine learning model. However, as image data increases over time, it is hard and inefficient to utilize an ever-increasing image data bank. As a solution, we introduce prototyping, a method to create a smaller, yet powerful training database that remedies the data complexity inefficiencies without sacrificing accuracy.

Across MNIST, EMNIST, and KMNIST, they will represent the increasing complexity of image data and determine if there are possible methods to ensure accuracy can be maintained through varying sizes of prototypes. In general, we will introduce random selection and K-Means as our prototyping methods, and we will introduce more complex machine learning algorithms if greedy classifiers turn out to be ineffective tools for prediction.

2 Ideas for Prototype Selection

To explore various ideas for prototype selection, we will examine the "worst" case scenario by arbitrarily picking points under a uniform distribution. The other will be through variations of K-Means clustering, a more probabilistic stance on creating centroids.

2.1 Random Selection

The motivation behind the random selection is quite simple, as it requires a selection of M points from the full dataset of size N under a uniform distribution without replacement. But considering the prototype of size of M, the number of possible combinations that can be made will be

$$\approx \frac{\sqrt{2\pi N}(\frac{N}{e})^N}{\sqrt{2\pi M}(\frac{M}{e})^M \cdot \sqrt{2\pi(N-M)}(\frac{N-M}{e})^{N-M}}$$

according to Stirling's approximation. The takeaway from this approximation is that as $M \to \frac{N}{2}$, the number of combinations for M samples increases in variability. Upon any size M greater than half of the entire training dataset, it will slowly resemble the training dataset more. For simpler datasets such as MNISTs, we will not have to consider the size of variation resulting from random sampling compared to EMNIST and KMNIST. For images with greater output classes and image complexity, we suspect that the random selection will result in "poor" prototypes and increase misclassifications greatly.

2.2 K-Means Selection

For a more fleshed-out method of prototype selection, we will utilize K-Means selection that uses the following method:

1. Arbitrarily choose an initial k centers $C = \{c_1, c_2, \ldots, c_k\}$.

2. For each $i \in \{1, \ldots, k\}$, set the cluster C_i to be the set of points in X that are closer to c_i than they are to c_j for all $j \neq i$.

3. For each $i \in \{1, \ldots, k\}$, set c_i to be the center of mass of all points in $C_i : c_i \frac{1}{|C_i|} \sum_{x \in C_i} x$.

4. Repeat Steps 2 and 3 until C no longer changes

The result is that this selection will result in $k = M$ centroids that will serve as prototypes for our learning algorithms in train itself. However, a general drawback of this method is that to determine M prototypes, the algorithm requires that all N data points be passed through. Nevertheless, having clusters based on mathematical calculations should increase accuracy.

2.3 K-Means++ Selection

The general drawback of regular K-Means is that the development of the centroids is done through the initialization of centers through an arbitrary process. As per Lloyd's algorithm, it is a polynomial time problem that does not guarantee global convergence; at best, it will be a "good" cluster. Now to determine a globally perfect cluster, it becomes an NP-Hard problem which makes it insensible to test for. However, to approach "perfection" in this clustering problem, a paper by (David Arthur, 2007) introduces a centroid initialization method that supposedly improves accuracy. The method is as follows:

1a. Take one center c_1, chosen uniformly at random from X.

1b. Take a new center c_i, choosing $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$

1c. Repeat Step 1b. until we have taken k centers altogether.

2-4. Proceed as with the standard k-means algorithm.

The author's claim with their paper is that because the initial point selection is done through a probabilistic stance, it creates more separation. Hence, the actual K-Means convergence occurs with early stopping and higher accuracy. The general issue with this algorithm is that although it has the potential to create better separation, the time complexity drastically increases during the initialization stage. When considering the number of prototypes to develop, it is crucial to consider the complexity of the entire image data set. As we have already discussed, having more prototypes will increase classification accuracy as there is more training data to rely upon. However, the goal of this project is to determine if there is a median between data space size and classification accuracy.

3 Ideas for Machine Learning Algorithms

To determine what type of machine learning algorithm is appropriate for image classification, we must consider how large the prototype dataset is. Some algorithms are greedy, so they simply generate predictions based on the training dataset through memory. Hence, when greedy algorithms do not perform well, it may be a sign to introduce more computationally powerful machine learning methods to increase accuracy.

3.1 K Nearest-Neighbors

For the most rudimentary algorithm, we will consider K Nearest-Neighbors. This will be considered a "greedy" algorithm, as there is no inherent learning process. To generate predictions, the algorithm will return the closest point label based on the average of the Euclidean distance between the unseen point and the closest K points. A great feature of K Nearest-Neighbors is that it exhibits consistency. So, increasing the number of training points $n \to \infty$, the risk (or error) of the algorithm is $R(h_n) \to R^*$. However, it should be noted that n should grow at a higher rate than k.

3.2 Logistic Regression

As a further step from K-Nearest Neighbors, we will also implement multi-class Logistic Regression. Compared to a Nearest-Neighbors algorithm, this takes a probabilistic stance with learned weights from a Gradient Descent algorithm. A key difference is that instead of considering a memory data space that minimizes Euclidean distance, the entire process is parametrized. In general, the process follows these features:

- Label Space: $\mathcal{Y} = \{1, 2, \ldots, k\}$
- Parameterized Classifier: $w_1, \ldots, w_k \in \mathbb{R}^d, b_1, \ldots, b_k \in \mathbb{R}$:

$$P(y = j|x) = \frac{e^{w_j \cdot x + b_j}}{e^{w_1 \cdot x + b_1} + \ldots + e^{w_k \cdot x + b_k}}$$

- Prediction: given a point x, predict label $\arg\max_j (w_j \cdot x + b_j)$
- Learning: Given $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$. Find: $w_1, \ldots, w_k \in \mathbb{R}^d$ and $b_1, \ldots, b_k$ that maximize the likelihood:

$$\prod_i^n P(y^{(i)}|x^{(i)})$$

The potential issue with this algorithm is that although there is far less memory complexity relative to KNN, the learning process requires an epoch across the entire training data. Hence, the objective here is to determine how much we can limit the size of the training data through prototyping.

3.3 Convolutional Neural Network

As for our most advanced model, we will implement a convolutional neural network which is a core model for accurate image machine learning. The components of the neural architecture are as follows:

- Convolutional Layers: Feature Extraction
- Rectified Linear Units: Application of Non-Linearity
- Pooling: Applies down sampling

Through these processes, the model will output multiclass logits that allow classification for the images that we will process.

3.4 Adaptive Learning Rates

For experimental purposes, we will also introduce adaptive learning rates for our Logistic and Neural Networks. In general, the learning process for a non-closed form solution is as follows:

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

In this format, η would need to be set as sufficiently small to ensure convergence in on the convex function will be achievable. But as $\eta \to 0$, we will see that the loss decrease per epoch will be negligible. Hence, the learning algorithm will require many epochs (which is a problem when the training dataset is large and high dimensional). To offer a solution, we will introduce adaptive learning rates based on the Hessian Matrix. The general benefit of using a Hessian is that it considers the curvature of the loss function, so having an adaptive learning rate will net in faster conversions and fewer epochs. Having fewer epochs will be great for training datasets with significant sizes. Training time can drastically decrease and there is a lower penalty for using a larger training dataset to learn weights.

Specifically, we will be using a Lipschitz constant discussed in Nutini et al. (Nutini et al., 2015). The idea is to use a learning rate $\frac{1}{L} \geq \eta$. Where L is:

$$L = \max \lambda_i(\nabla^2 f(x))$$

In this form, λ_i is the eigenvalue for the Hessian matrix.

4 Empirical Results Analysis

4.1 MNIST

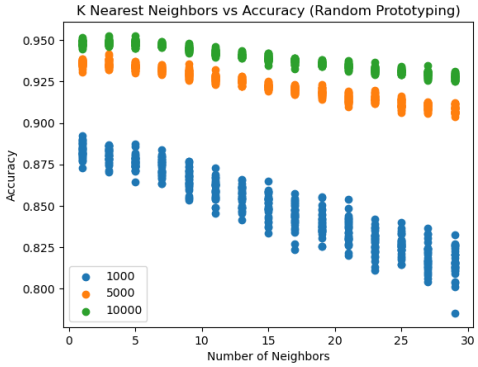
For the MNIST dataset, this represents the least complex data. This is because there is only 10 possible output classes and there are no significant deviations between each possible image. Hence, the model developments will be limited to "greedy" models such as Nearest Neighbors.

4.1.1 Random Selection & KNN

For random selection, it is important that confidence intervals are built into the accuracy metrics being produced. Nevertheless, we will only use KNN as the empirical results state there is no need to.

Subset Size	95% CI
10,000	(0.9488, 0.9488)
5,000	(0.9352, 0.9353)
1,000	(0.8849, 0.8855)

Evidently, we see here that having more prototypes will improve testing accuracy and narrow the confidence interval for accuracy. Therefore, there is great potential even for random prototyping on simple datasets. Even without fleshed-out methods, you can arbitrarily select training data points to achieve good classification.

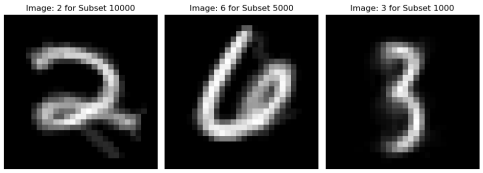


We can see how this model operates beyond 1-NN on random selection. As seen below, there is slight variation within the possible accuracy you can get from using random selection. Nevertheless, they observe the principles of consistency, the more training points, the data will converge towards R^*.

4.1.2 K Means & KNN

We analyze the effectiveness of K-Means-based prototype selection for 1-Nearest Neighbors classification. By leveraging cluster centroids as training prototypes, we analyze classification accuracy

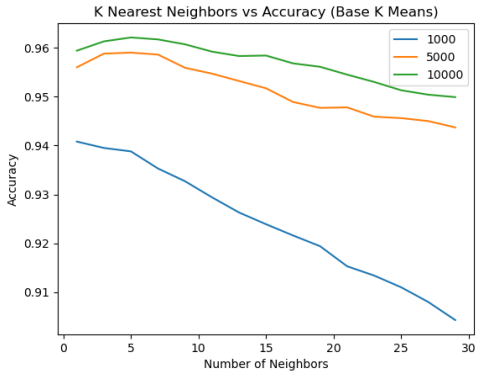
across data subsets of size 1000, 5000, and 10,000. For reference, the centroid images are displayed below:



As the number of prototypes decreases, we can see that it is harder to discern images as they can become blurry. (This principle will follow for all image datasets)

Subset	1-NN Accuracy
10,000	0.9594
5,000	0.9560
1,000	0.9408

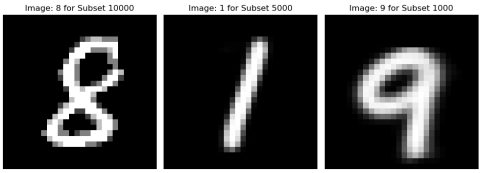
The results demonstrate that this selection method consistently achieves high accuracy with minimal variance across all subsets, highlighting the robustness of centroid-based selection in capturing representative data points for classification. Next, we explore K-Means prototyping for a K-Nearest Neighbor classifier. The graph represents the relationship



between the number of neighbors k and classification accuracy. We observe that accuracy is highest at lower k values (k = 1 to 5) and decreases as k increases across all subset sizes. This decline in accuracy at higher k values is due to over-smoothing, where majority voting among neighbors leads to less distinct decision boundaries. Additionally, larger datasets improve classification performance, as they provide better-distributed and more representative clusters.

4.1.3 K Means++ & KNN

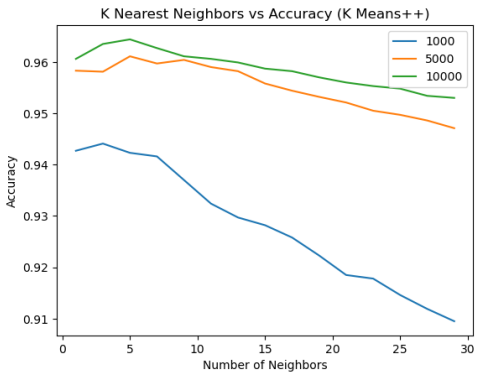
Next, we examine the K-Means++ selection for 1-Nearest Neighbours classification. For reference, the centroid images are shown below:



As the number of prototypes decreases, there is noticeable blur that makes each image harder to interpret by our algorithms. (This principle will follow for all image datasets)

Subset Size	1-NN Accuracy
10,000	0.9606
5,000	0.9583
1,000	0.9427

The findings confirm that K-Means++-based prototype selection is also a viable approach for 1-NN classification. It improves upon standard K-Means by optimising centroid initialisation to produce more consistent clustering and better classification performance, as evidenced by the higher accuracy values. By reducing variability in centroid placement, K-Means++ mitigates classification boundary instability that arises particularly in smaller subsets. We now turn to its application in a K-Nearest Neighbours classifier.



Similar to the K-Means graph, accuracy decreases with increasing k across all subset sizes due to over-smoothing of decision boundaries. Larger subsets maintain higher accuracy throughout, demonstrating how more data improves cluster stability and classification performance. However, under K-Means++, the accuracy gap between the 5,000 and 10,000 subsets is smaller than in standard K-Means as optimised initialisation ensures that even moderate subsets achieve near-optimal performance. The marginal accuracy gain at 10,000 over 5,000 suggests that clustering quality is already optimised.

4.2 EMNIST

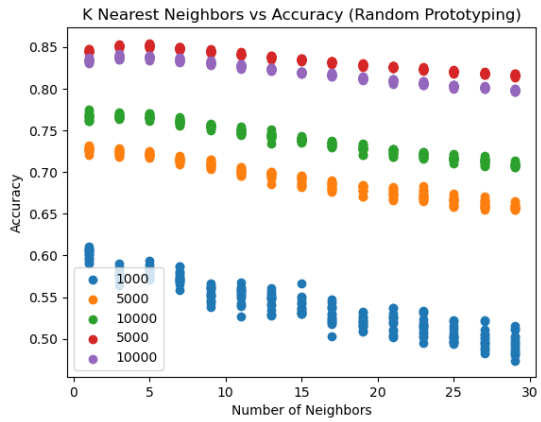
As a step up in image complexity from numbers, we will look at EMNIST using Latin characters A-Z. We notice here that the increase in the available training size and possible output classes shows that the image classification problem will increase in complexity.

4.2.1 Random Selection

As iterated before, we will consider confidence intervals when attempting to view our empirical results. Note: We will not view CI of Lipschitz based learning algorithm as it is not meant to change accuracy heavily.

Subset Size	95% 1NN	95% Logit	95% NN
75,000	(0.842, 0.849)	(0.689, 0.703)	(0.922, 0.934)
50,000	(0.829, 0.838)	(0.681, 0.696)	(0.920, 0.926)
10,000	(0.762, 0.771)	(0.632, 0.676)	(0.862, 0.907)
5,000	(0.722, 0.734)	(0.598, 0.670)	(0.838, 0.876)
1,000	(0.588, 0.612)	(0.539, 0.590)	(0.651, 0.736)

The results point towards the fact that 1NN performs worse on the EMNIST dataset (which was generally expected). This points towards the fact that as the number of output classes increases, and the possible variation between each image increases, greedy models will struggle to maintain high levels of accuracy.



Examining how greedy models perform on random selection, we see that the principles of consistency are exhibited; as you increase the number of points, the error converges to R^*. Moreover, there is more room for variations when the number of prototype points decrease. Nevertheless, we see that aggressive prototyping is starting to become less viable as the number of classes and images increase.

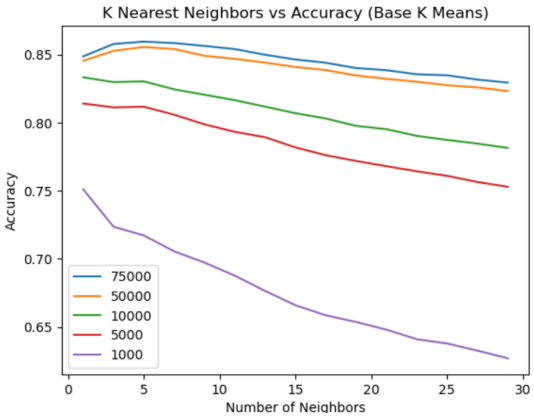
Now examining the learning algorithms, we see that a regular Logistic regression fails to observe high accuracy as they are only at best, linear decision makers. However, we can see that for the Neural Network, you can implement more aggressive prototyping strategies and still achieve high accuracy. This indicates that there is high potential for training size reduction via prototyping on complex datasets.

4.2.2 K Means

We examine K-Means prototyping for five types of classification models. The results show that larger subsets consistently achieve higher accuracy across all classification approaches.

Subset	1-NN	Base Logit	Lip Logit	NN	NN Lip
75,000	0.8486	0.6946	0.6941	0.9261	0.9248
50,000	0.8454	0.6789	0.6855	0.9201	0.9123
10,000	0.8332	0.6520	0.6592	0.8612	0.8619
5,000	0.8140	0.6496	0.6472	0.8313	0.8230
1,000	0.7510	0.6306	0.6319	0.6623	0.6516

1-NN accuracy improves as subset size increases. This implies that K-Means selection benefits from more training data to enhance centroid placement and reduce error. When subset size decreases from 5,000 to 1,000, there is a more noticeable drop in accuracy which highlights the challenge of prototype selection in small datasets. Fewer data points result in less stable clustering and leads to more misclassifications instances.



Moreover, the principle of consistency with KNN is observed; the more points, the risk approaches R^*.

Logit based models show the lowest accuracy compared to the other models, suggesting that logit-based methods do not generalise as well as nearest-neighbour classification under K-Means selection. The difference between logit and Lipschitz logit is minimal, so any improvement from Lipschitz logit

is marginal in this setup. Neural networks (NN) outperform all other models, and NN Lipschitz achieves similar accuracy with minor variations across different subset sizes. The small performance gap between the two indicates that incorporating a Lipschitz constraint enhances stability and robustness without drastically impacting classification accuracy.

4.2.3 K Means++

We also analyse K-Means++ prototype selection for the same five models.

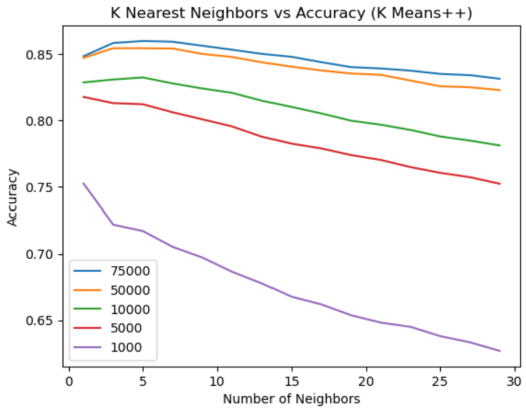
Subset	1-NN	Base Logit	Lip Logit	NN	NN Lip
75,000	0.8539	0.6921	0.6888	0.9244	0.9248
50,000	0.8472	0.6843	0.6866	0.9196	0.9191
10,000	0.8287	0.6403	0.6622	0.8644	0.8774
5,000	0.8178	0.6398	0.6529	0.8215	0.8359
1,000	0.7526	0.6268	0.6270	0.6657	0.7003

Compared to standard K-Means, 1-NN accuracy is slightly improved across all subset sizes which supports that better centroid initialisation helps refine prototyping. Additionally, the accuracy drop between the 1,000 and 5,000 subsets is smaller than standard K-Means, confirming that K-Means++ mitigates the instability in smaller datasets by producing better-separated and more representative centroids. Neural network-based models maintain strong accuracy and continue to outperform other methods. NN Lipschitz performs better on smaller subsets (1,000 - 10,000), demonstrating that Lipschitz constraints can improve classification performance when paired with well-initialised centroids. This suggests that K-Means++ enhances the effectiveness of Lipschitz-constrained networks by providing more stable cluster assignments. Logit-based models still show the lowest accuracy, reinforcing that nearest-neighbour and neural network methods remain superior for prototype selection.

Compared to standard K-Means, K-Means++ maintains slightly higher accuracy across k values, suggesting that better centroid placement improves K-Nearest Neighbours performance. The accuracy drop is more gradual as well which indicates improved stability.

4.3 KMNIST

As for the final dataset, we will look at KMNIST, an image dataset for old Japanese literature. Relative to modern Japanese text, there is greater variance in each character relative to each output type. Hence,



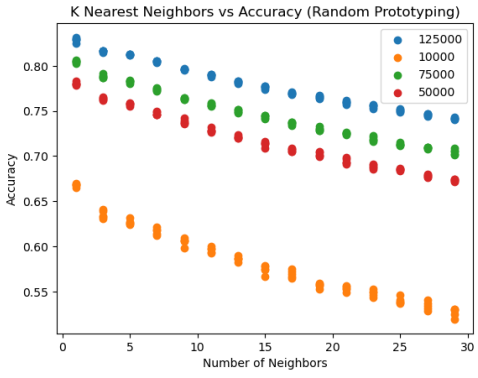
this is where the greatest challenge of prototyping will occur. (Clanuwat et al., 2018)

4.3.1 Random Selection

As mentioned before, we will consider confidence intervals for our Nearest Neighbors and base learning algorithms because that is where we will see insights for prototyping.

Subset Size	95% 1NN	95% Logit	95% NN
125,000	(0.842, 0.849)	(0.549, 0.568)	(0.902, 0.911)
75,000	(0.801, 0.808)	(0.541, 0.562)	(0.882, 0.889)
50,000	(0.778, 0.787)	(0.534, 0.554)	(0.856, 0.874)
10,000	(0.658, 0.676)	(0.488, 0.504)	(0.729, 0.772)

According to our results, we see that we see similar trends in the data as we see CNN methods perform the best and linear methods as the worst. Hence, even in the worst case, we can arbitrarily take a smaller subset of training points and use Neural Networks to classify complex images.



Moreover, we see in random selection for KMNIST that there is a clear downward slope in accuracy when using Nearest-Neighbors algorithms. This deviates from the prior trends when we saw accuracy peak when $k \in \{3, 4, 5\}$. Thus, to achieve the best

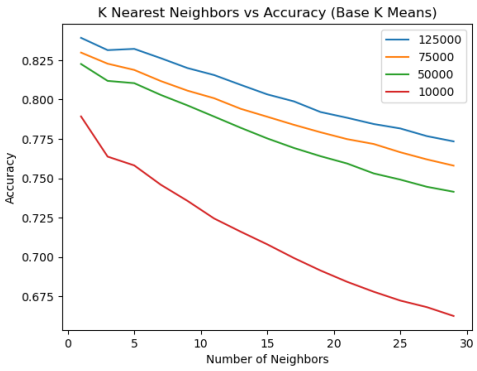
accuracy for more complex images, using a single nearest neighbor appears to be the best. However, for 1NN the principle of consistency is violated. People cannot naively increase the number of training data points and expect accuracy to increase. As long as the nearest data point remains unchanged, statistical theory states that classification on unseen data will be unchanged.

4.3.2 K Means

As a step up from Random Prototype selection, we once again turn to K Means centroids.

Subset	1-NN	Base Logit	Lip Logit	NN	NN Lip
125,000	0.8392	0.6946	0.6941	0.9261	0.9248
75,000	0.8298	0.6789	0.6855	0.9201	0.9123
50,000	0.8226	0.6520	0.6592	0.8612	0.8619
10,000	0.7893	0.6496	0.6472	0.8313	0.8230

Compared to our other experiments, we see that there is a natural decrease in accuracy as the image complexity has increased. Nevertheless, the same trends show that CNN performs the best across the board even with smaller prototypes. Although there is no "strong" difference between the random prototyping case, the fact that there is no randomness will provide a benefit for using K Means prototypes.



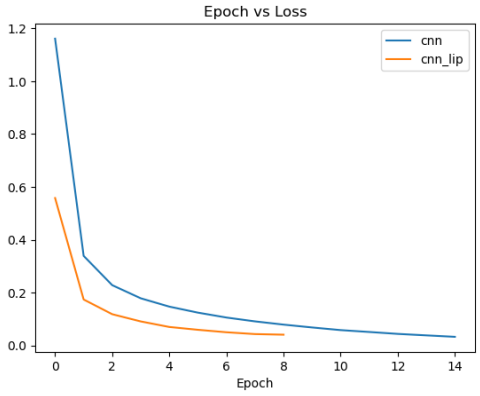
However, one interesting result we have uncovered is that KNN has shown different results when considering varying levels of k. Previously, we have seen that the peak often occurs around 3 or 5 neighbors, whereas now we see 1NN be the best. This is a key point from a theoretical perspective of consistency. Overall, we see consistency across all levels of k, as the number of points increases, accuracy increases, and risk decreases. However, it is important to recognize that 1-Nearest Neighbors is not a consistent algorithm because there is no guarantee that each additional point m will affect the classification result of each test point. So, this provides an argument that even though more training results improve accuracy, this may not hold on a point by point basis.

4.3.3 K Means++

We have previously discussed and viewed the viability of using K Means++ centroid creation. The reason being is that through initial centroid creation based off of a probabilistic Euclidean stance, you can create better separation for the iterative process of creating the centroids. However, when you have high dimensionality against a large dataset, creating pairwise distances becomes infeasible. So we have concluded that K Means++ is not possible in large image datasets.

4.4 Usage of Lipschitz Constants

In our results, we have established that the usage of Lipschitz constants as an adaptive learning rate has no negative on accuracy. Hence, we will take this section to view if it can make learning more efficient. Data collected from our Neural Network on the full KMNIST dataset are as follows:



Overall, we see that the loss convergence on a Lipschitz-based Neural Network has shown faster convergence. This means that there is potential using Lipschitz constants on larger datasets as learning algorithms require less training iterations. However, there are some key caveats when considering the space complexity of using an adaptive learning rate. As a learner, the benefit of using a constant rate is that there is no space or computational complexity that will slow down the algorithm. Considering the dimensionality of image datasets, we will advocate that it will only be practical on low dimensional images. So, the question becomes a trade-off between training data size and learning

time. This is will vary on the image classification task at hand and how imperative model training or accuracy is. For reference, MNIST is $\mathbb{R}^{784}$.

5 Real-world Applications

To extend the ideas of the prior experiments to a greater scope, we will view how certain large data fields may benefit from prototyping.

5.1 Autonomous Vehicles

Autonomous cars rely on vast amounts of image data for real-time decision making, including object detection, lane tracking and obstacle avoidance. Processing these large datasets requires high computational power and increases prediction time, which is a critical limitation in real-world deployment. By selecting representative training prototypes, this project could help develop models that maintain high accuracy while enabling faster decision-making, leading to safer and more efficient autonomous systems.

5.2 Medical Imaging

AI-driven medical imaging models, used in radiology and pathology, demands extensive labeled datasets for accurate diagnosis. However, acquiring and annotating medical images is time-consuming, expensive and often limited due to patient privacy concerns. Leveraging data prototyping techniques allow AI models to be trained on smaller but comparable quality subsets of medical images, making AI-assisted diagnosis more efficient, accessible, and scalable.

5.3 Cybersecurity

Biometric authentication and facial recognition systems also require training on large-scale image datasets, which increases storage demands and computational costs. Additionally, poorly optimised models can lead to biases and misclassifications errors, raising concerns in sensitive security applications like fraud detection and identity verification. This project's data selection approach could help improve efficiency and fairness in biometric authentication by ensuring models learn from a diverse yet computationally feasible dataset, improving both speed and accuracy.

5.4 Satellite Imaging

Satellite image analysis plays a crucial role in climate change tracking, natural disaster monitoring, and defense and security operations. These models process massive amounts of geospatial data, which makes large-scale image analysis computationally intensive. A data prototyping approach could allow satellite models to train on smaller but representative subsets, facilitating faster disaster detection, infrastructure monitoring, and border surveillance. This improves real-time decision making, optimises resource allocation, and ensures more effective deployment of satellite-based solutions across multiple industries.

6 Conclusion

Data prototyping has the potential to enhance computational efficiency, scalability, accessibility, and has the potential to minimise cost by reducing dataset complexity while maintaining accuracy. This approach lowers storage and processing costs, making it more feasible for smaller organisations and real-time decision-making. In fields like climate monitoring, defense, and urban planning, faster processing allows more effective resource allocation and policy implementation. Additionally, optimising data usage decreases energy consumption to promote a more sustainable approach to data analysis.

Nevertheless, these findings should not be taken at face value. Although we have empirically shown that prototyping can be viable when using the appropriate ML models, each organization must value what the cost of misclassifications may be to their organization. Hence, establishing expectations with stakeholders is essential to understanding how aggressive prototyping can be used. Under the case that image prototyping is infeasible, we have empirically shown that adaptions to stochastic learning methods can remedy the problem posed by learning on large datasets. But of course, you must consider the dimensionality and size of the data to consider implementing adaptive learning.

Github Link:

https://github.com/rsm-roguchi/mgta415-project

References

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. 2018. Deep learning for classical japanese literature.

Sergei Vassilvitski David Arthur. 2007. k-means++: The advantages of careful seedings. *Stanford CS*.

Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander, and Hoyt Koepke. 2015. Coordinate decsent converges faster with the gauss-southwell rule than random selection. *JMLR: W&CP*, 37.