Generalizing Image Reconstruction from Individual Patches Using K-Means

John Kincaid, Henry Chou, Cheng-Hao Tsai, Bishane Sagal
Electrical and Computer Engineering
University of California, Davis
Davis, USA

Github Link: https://github.com/bchou9/eec289Ahw/tree/main

***Index Terms*—Unsupervised Learning, K-means**

I. Introduction

In this assignment, we aim to use K-means clustering of image patches of the MNIST dataset to understand the underlying relationships between the number of clusters needed and its accuracy in reconstructing the entire patch space, interpret the meanings, if any, of the patches, as well as unearth the ways in which the patches used to reconstruct the original digit. Although there are a myriad of other well known, faster approaches to clustering [1]–[3], such as through the use of geometric reasoning [4], spectral relaxation [5], adaptive distance bounds [6], as well as deep clustering of local manifolds [7], we used the iterative refinement technique (the "naive" K-Means approach) for this assignment due to its ease of implementation. We also implemented the Mini-Batch version of K-Means, allowing us to compare and contrast the resulting reconstructed images along with its accuracy between these two fundamental clustering methods.

For the assignment step of the k-clustering algorithm, we first assign each 5 by 5 image patch (a subset of which is depicted in Figure 1) to the nearest cluster using Euclidean distance, and then for the update step, we recalculate the means (*i.e.*, centriods) until the k means algorithm converges to its optimal point (*e.g.*, when the clusters no longer change), as it is done in the actual k means clustering algorithm [8]. Ultimately, our analysis of K-Means Clustering could reveal insights with regards to how to choose the best hyper-parameters - the number of clusters, degree of sparsity, number of patches, and patch size.

II. Results

Figure 2 shows that the error is decreasing as the iteration times increase, suggesting an exponential negative correlational relationship between these two variables. This means that although the chosen number of iterations must be large enough for the algorithm to converge to the most accurate reconstruction of the digit, there is a diminishing return as we attend to iteration values that are excessively high. Thus, the results from Figure 2 suggests that we choose a number of clusters around 2^8 (256) in order to achieve an optimal balance between performance and reconstruction accuracy.

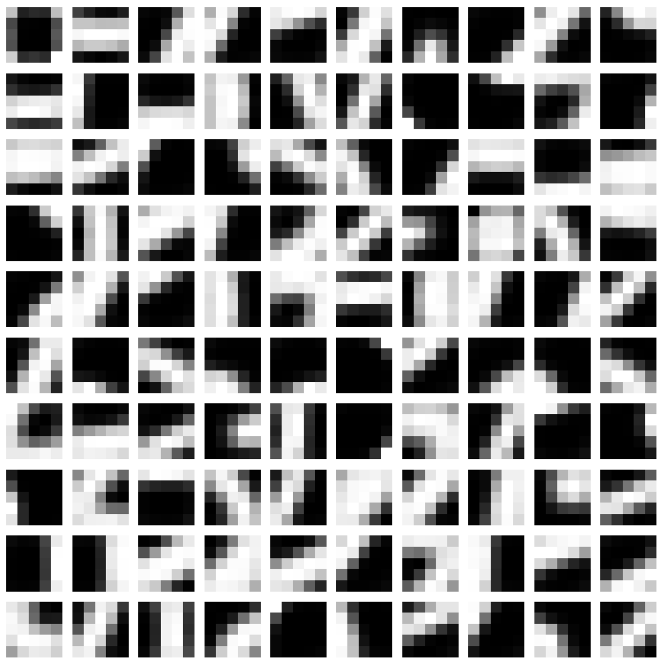
Fig. 1. 100 clusters, with each individual cluster being of size 5 by 5

Figure 3 shows that the average error is decreasing when the numbers of clusters increase. Initially, the average error is very high, but it decreases dramatically when the number of clusters increase until about 2048 clusters. Once the number of clusters reach 2048 or greater, we again see a diminishing return with regards to minimizing the average error. In fact, once we reach extremely high cluster counts (16384 or greater) we reach a saturation point with 0.09 for our average error. The results reveal that the clustering model more accurately fits the data when the number of clusters increase. However, it also shows that more is not necessarily significantly better, as adding more clusters on top of already very high cluster counts results

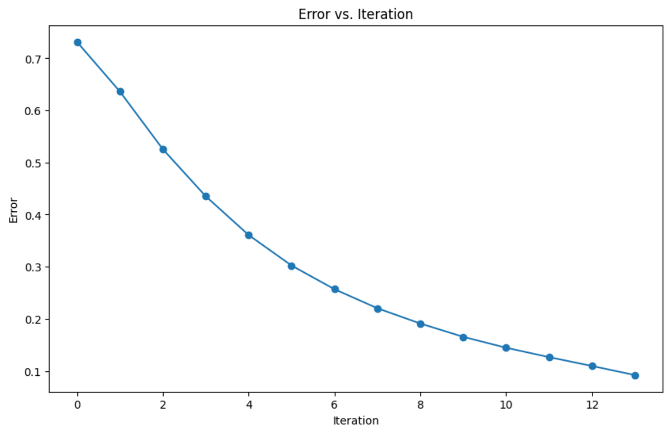

Fig. 2. Relationship between the error rate vs number of iterations

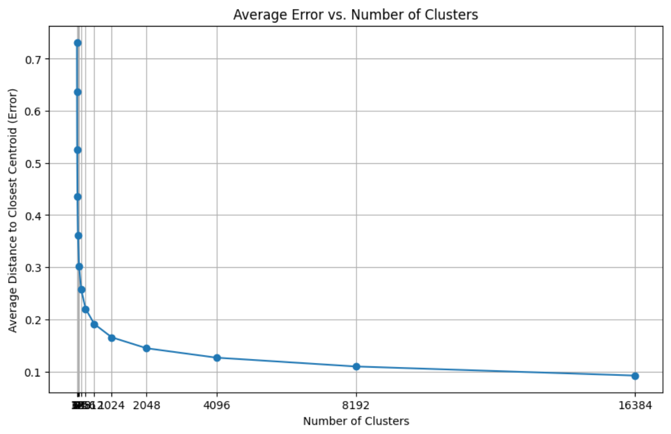

Fig. 3. Relationship between the average error vs number of clusters

in only a marginally lower average error, and marginal visual improvement of reconstructed digits, such as the one shown in Figure 8.

Zooming in, when we varied the number of clusters from 24 to 256 (Figure 6), the most significant drop in error occurred as the number of clusters increased from 24 to about 64. This suggests that the increase in the number of clusters can significantly improve the clustering accuracy of the digits at hand during the initial stage of the K-Means Clustering.

Nevertheless, the slope of error reduction begin to flatten out dramatically as the cluster counts increase from 64 to 256. This suggests that the clustering model is nearing the optimal number of clusters and that 128 seems to be an optimal point for our clustering implementation, since the error rate only decreases marginally beyond 128 clusters. Similarly, this diminishing error improvement is also confirmed by the negligible visual improvements in terms of reconstruction accuracy, when going from 256 to 1024 centroids (Figures 4 and 5).

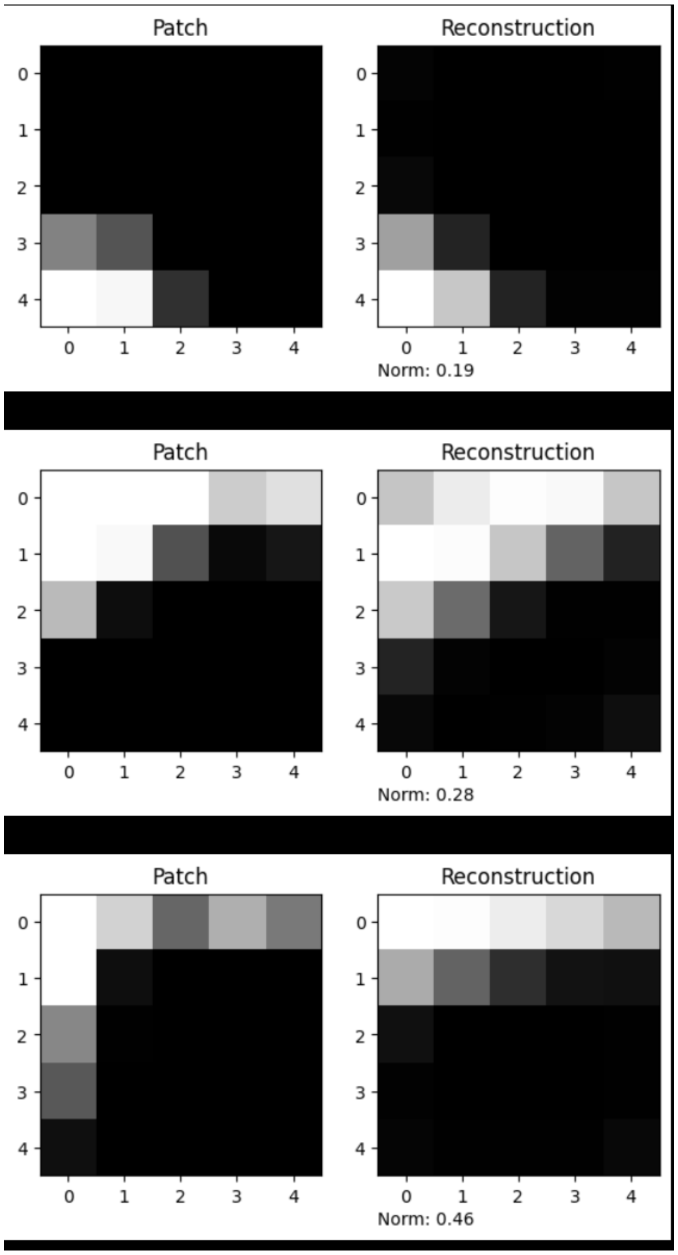

Fig. 4. A sample of three original patches out of the 210,000 total number of patches, and their respective reconstructed patch using 256 cluster centroids.

III. Discussion

A. *Generalizing dictionary reconstruction to novel images*

We found that 128 clusters sufficiently covered the patch space as they provided an adequate reconstruction of patches for all digits across both vanilla K-Means and the Sparse Encoding variant, according to Figure 6. The elbow of the loss curve occurs at this mark and the reduction in error is marginal with increasing clusters. The clusters are a vector for which assigned patch vectors have minimum dissimilarity. A physical interpretation of the clusters is that they represent a common "piece" of the stroke of

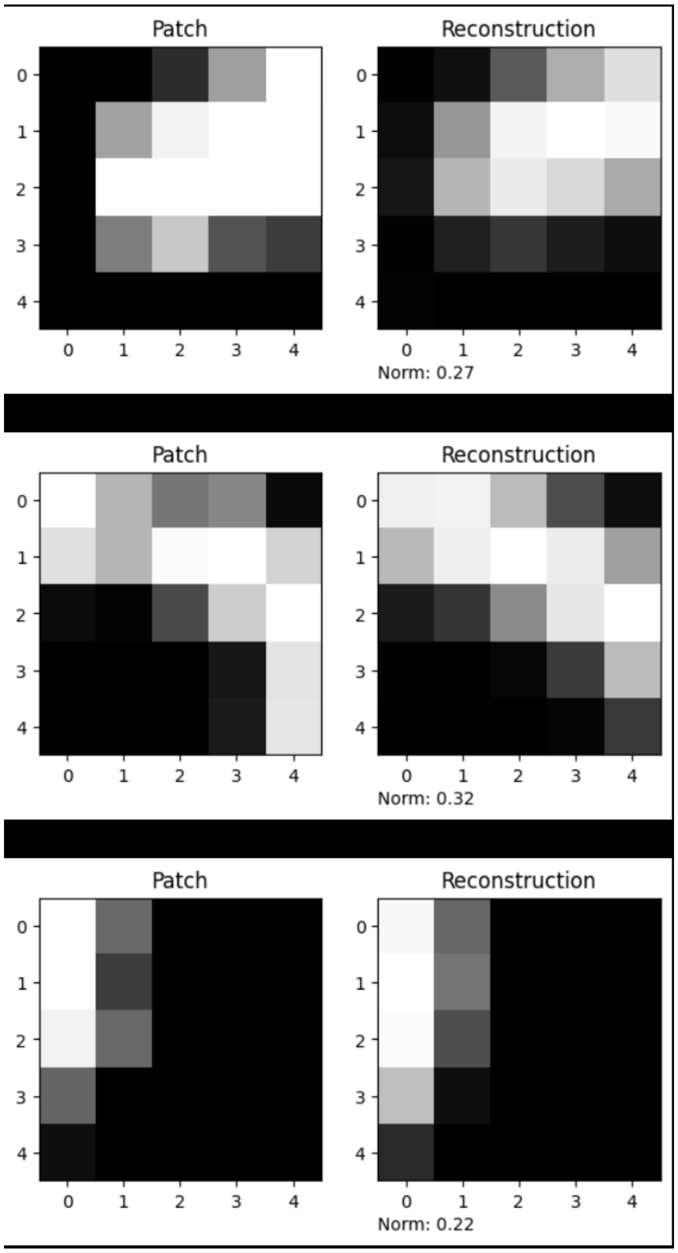

Fig. 5. A sample of three original patches out of the 210,000 total number of patches, and their respective reconstructed patch using 1024 cluster centroids.

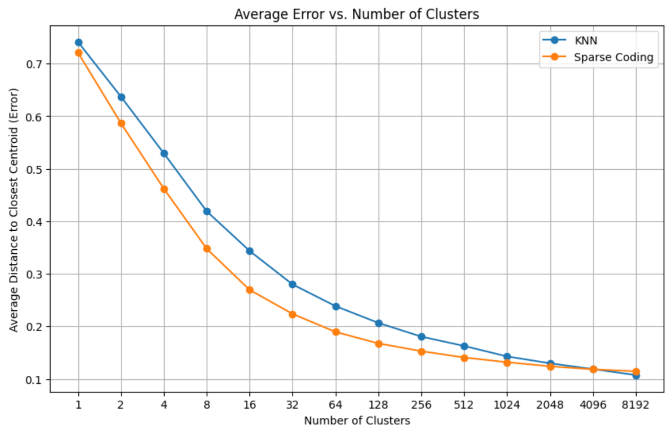

Fig. 6. Comparison of the relationship between the error rate versus the number of clusters, between vanilla k-means and sparse encoding clustering. The sparsity is Less Sparse (sparsity parameter = 0.1). Using sparse encoding, the same average error can be approximately obtained with half the number of dictionary elements. The best gains from sparse encoding happen between 8 and 128 clusters, when the dictionary is close to covering the patch space, and converge to no gain, or even worse performance, at large dictionaries (1024+).

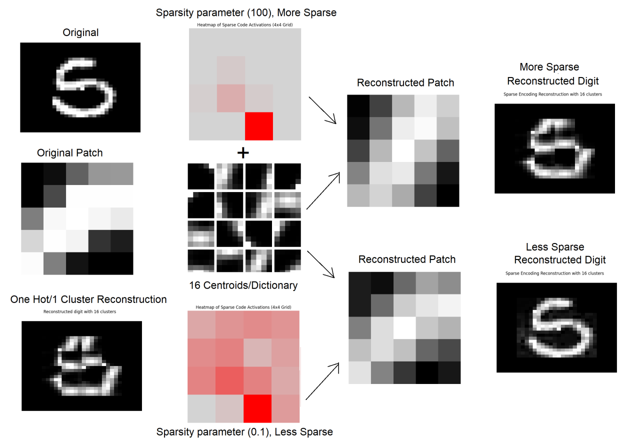

Fig. 7. Reconstructions of a specific patch and it source digit using a More Sparse and Less Sparse encoding from a dictionary of 16 clusters. The top and bottom heatmaps shows the weights and activation of the centroids for the More Sparse encoding and the Less Sparse encoding, respectively. The Less Sparse encoding reconstructs the image very well, comparable to the 64 cluster one-hot encoding. The More Sparse encoding is not as clear, but still better than the 1 cluster reconstruction, shown in the bottom left.

handwriting. As the writing utensil draws out a shape, in this case a digit between 0 and 9, the continuous line can be broken apart into patches that are the building blocks of the full image.

Although a combination of the patches (with overlap) can reconstruct the image, not all of the patches are unique to a single digit; there can be common patches between digits. One such example is the similarity between a 4 and 1. The straight vertical lines in both images can contain very similar patches from the patch space. The centroid for patches contained in these strokes could be the same and might be used in reconstructing both digits.

This raised an interesting question: could these centroids be building blocks for not only handwritten digits, but also handwriting in general or possibly even other images? Many different characters in handwriting can overlap with each other in term of their appearance, especially when it comes to those in East Asian languages [9], [10], and this also is true for many hand drawn images as well. Applying the same reasoning from the earlier example, a 1 and a lower case "l" may have similar building blocks from the patch space, and we would like to see if we can exploit this to create more accurate image reconstructions.

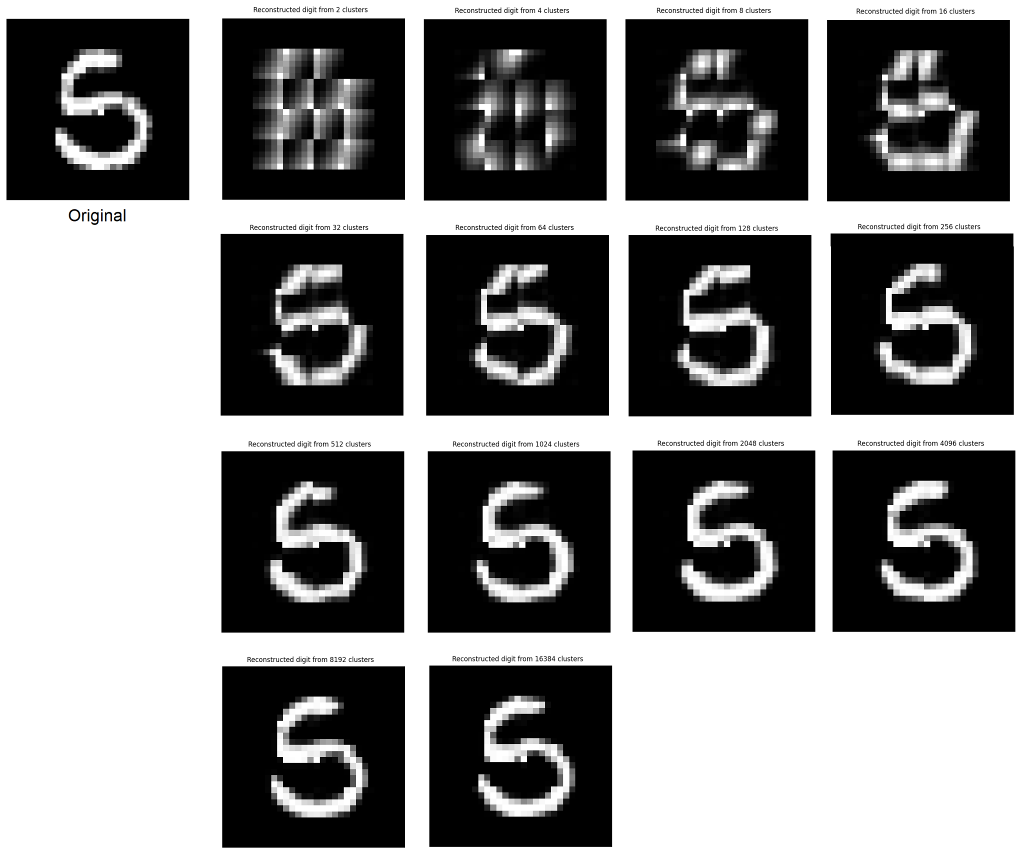
Original
Reconstructed digit from 2 clusters
Reconstructed digit from 4 clusters
Reconstructed digit from 8 clusters
Reconstructed digit from 16 clusters
Reconstructed digit from 32 clusters
Reconstructed digit from 64 clusters
Reconstructed digit from 128 clusters
Reconstructed digit from 256 clusters
Reconstructed digit from 512 clusters
Reconstructed digit from 1024 clusters
Reconstructed digit from 2048 clusters
Reconstructed digit from 4096 clusters
Reconstructed digit from 8192 clusters
Reconstructed digit from 16384 clusters

Fig. 8. Reconstruction of a padded '5' from single clusters. The 30x30 padded image is divided into 5x5 patches, which are then 'reconstructed' and glued back together. Note that a special 'null' centroid has been inserted into the dictionary to handle empty patches. We can use this chart to qualify the numerical error we calculated for the different centroid numbers. The written '5' is well-formed between 128-256 clusters. For clusters counts 1024 and beyond, the difference between the reconstruct digits are almost indistinguishable to the eye.

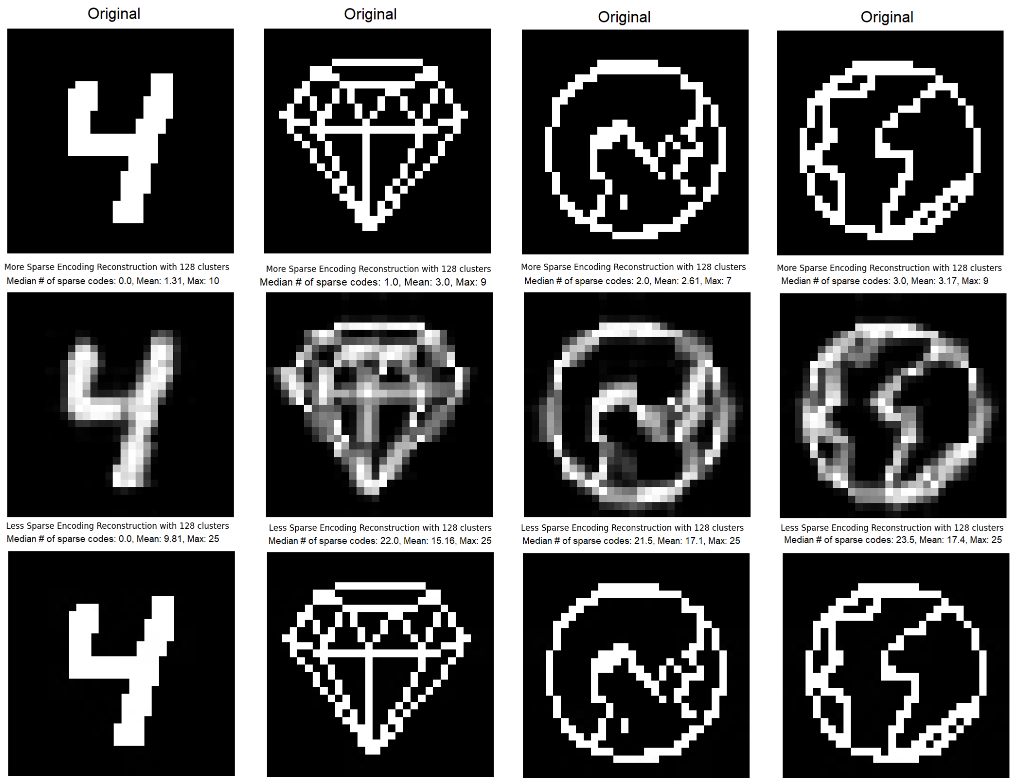
Original
Original
Original
Original
More Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 0.0, Mean: 1.31, Max: 10
More Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 1.0, Mean: 3.0, Max: 9
More Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 2.0, Mean: 2.61, Max: 7
More Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 3.0, Mean: 3.17, Max: 9
Less Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 0.0, Mean: 9.81, Max: 25
Less Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 22.0, Mean: 15.16, Max: 25
Less Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 21.5, Mean: 17.1, Max: 25
Less Sparse Encoding Reconstruction with 128 clusters
Median # of sparse codes: 23.5, Mean: 17.4, Max: 25

Fig. 9. Generalization of the reconstruction of images to those beyond simply numerical digits, which are reconstructed using higher sparsity parameter values of 100 (less sparse) as well as lower values of 0.1 (more sparse).

We explored this question using novel images and sparse coding, confirming our suspicion. The novel images were created in Microsoft Paint and saved as 28x28 bitmaps. The images were put through the same reconstruction function as for the MNIST dataset. Thus, they are also padded to a size of 30 by 30, broken into 36 individual 5 by 5 patches, and then reconstructed which in return is used to form a new image that is made up of those reconstructed patches. A dictionary of 128 clusters learned from the MNIST dataset patches was able to reconstruct novel image types. There seems to be a minimum of 128 patches needed to accurately reconstruct general 28x28 images, however, such patches are not only building blocks for handwritten digits; the 128 patches from the patch space can be used to generate different images (Figure 9).

B. Addressing the drawbacks of hard classifiers

During our experiments, we also confirmed the fact that K-Means suffers from the drawbacks associated with hard classifiers [8]. Points that are assigned to a particular cluster regardless of their distance to that cluster's centroid are viewed as equals to other points within that cluster [8], regardless of their actual distance between themselves and the centroid of the cluster that the point was assigned to. This means that even points that are near two or more different clusters are still assigned into only one of the clusters, and viewed as an equal point to all others in that cluster even though this point could have very well influenced the other clusters as well. Consequently, this could affect the accuracy of the reconstructed image from the patches, thus causing a sharp drop in terms of error when increasing from just a few clusters to 128. More clusters could help alleviate inaccuracies with the spatial representation of each patch during cluster assignment.

C. Comparing the results between clustering methods

Shifting focus to methodology, we used standard K-Means, Mini-Batch K-Means, and Dictionary Learning (sparse coding). Using all three K-Means methodologies we were able to achieve low error between centroids and patches, and fairly accurate reconstruction of digits. Mini-Batch K-Means converged faster, but the patches themselves and consequently, quality of the reconstructed images, was not as good as standard K-Means. The error for Mini-Batch was not as low as standard K-Means for centroid selection. Even for a large number of centroids (1000), the resulting images were lackluster. The shape of the image was accurate, but the resulting gray-scale image intensities did not match well with the original images. The white pixels were far grayer in the Mini-Batch K-Means whereas this was not as much of a problem for our standard K-Means reconstructions; this was true even with less centroids. However, neither standard K-Means nor Mini-Batch K-Means was able to create the full whites (255 values) reliably, but while experimenting with sparse coding we found that it was able to accurately and reliably reconstruct white patches and full images. In our case, we see a trade-off in Mini-Batch K-Means with the standard algorithm: faster convergence with reduced quality of results. However, other researchers have shown that Mini-Batch K-Means was orders of magnitude faster at converging to more accurate results when it comes to sparse cluster centers [11], [12]. Thus, in the context of patch reconstruction, mini-batch could generate a more accurate reconstruction in a shorter amount of time in cases where a lower sparsity parameter is used, as in the case of Figure 7, using low sparsity parameter values.

IV. Member Contributions

John Kincaid - Implemented dictionary learning (sparse encoding) and standard version of K-Means clustering, captions for Figure 1, 6, 7, and 8. Generated majority of figures, including Figure 9. **Henry Chou** - Wrote introduction and results sections of the report, explained the theory, second half of discussion section, captions for Figures 4, 5, and 9, as well as proofread and revise the report as needed. **Cheng-Hao Tsai** - Wrote Figure 2 and 3 description in results section of the paper. **Bishane Sagal** - Implemented Mini-Batch K-Means clustering (code on GitHub - "EEC289AHW1"), wrote item A and majority of item C in the Discussion section, and posed question about reconstructing novel/general images.

References

[1] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings."

[2] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, Jul. 2002.

[3] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics.* UC Press, Jan. 1967, vol. 5.1, pp. 281–298.

[4] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '99. New York, NY, USA: Association for Computing Machinery, Aug. 1999, pp. 277–281.

[5] H. Zha, X. He, C. Ding, H. Simon, and M. Gu, "Spectral Relaxation for K-means Clustering."

[6] J. Drake and G. Hamerly, "Accelerated k-means with adaptive distance bounds."

[7] R. McConville, R. Santos-Rodriguez, R. J. Piechocki, and I. Craddock, "N2D: (Not Too) Deep Clustering via Clustering the Local Manifold of an Autoencoded Embedding," Jun. 2020.

[8] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, 22nd ed. Cambridge University Press, 2019.

[9] D. Keysers, T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune, "Multi-Language Online Handwriting Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1180–1194, Jun. 2017.

[10] K. Chellapilla and P. Simard, "A New Radical Based Approach to Offline Handwritten East-Asian Character Recognition," in *Tenth International Workshop on Frontiers in Handwriting Recognition.* Suvisoft, Oct. 2006.

[11] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: Association for Computing Machinery, Apr. 2010, pp. 1177–1178.

[12] J. Béjar Alonso, "K-means vs Mini Batch K-means: A comparison," May 2013.