Enhancing Feature Discrimination in Stacked Autoencoders: Comparative Analysis and Robustness to Noise in Image Classification

Tess O. Christensen
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL USA
tolivia.christen@ufl.edu
Completed and evaluated part 2 and aided in the production of this paper

Catalina B. Murray
Department of Electrical and Computer Engineering
University of Florida
Gainesville, FL USA
catalinamurray@ufl.edu
Completed part 1 and the foundation for part 2 and aided in the production of this paper

***Abstract*—**
In this study, we examine how effective stacked autoencoder networks (SAEs) are at pulling out features and classifying data, especially when it comes to making distinctions in a hidden part of the network called the latent space. We use the Kuzushiji-MNIST dataset to compare two methods. First, we look at a standard SAE with five hidden layers trained using mean squared error (MSE).

We then investigate the SAE's robustness to noise by introducing impulsive noise to the input images and training the SAE with a correntropy cost function. This approach's effectiveness in noise reduction and its impact on classification accuracy is quantitatively analyzed and compared against the traditional MSE-based SAE.

In the second part of our research, we propose and implement a novel penalty function designed to enhance discrimination in the SAE's latent space. We discovered that the SAE-SVM model using a regularizer proved more accurate than the original model. These enhancements suggest that such an approach could benefit other complex classification tasks beyond the Kuzushiji-MNIST dataset.

I. Introduction

The rise of deep learning, particularly in pattern recognition and feature extraction, has been aided by the introduction of Stacked Autoencoders (SAEs). SAEs enable the learning of hierarchical representations without labeled data, making them pivotal in the domain [4]. This research explores the potential of SAEs in image classification using the Kuzushiji-MNIST dataset, a challenging benchmark in handwritten character recognition. Unlike our previous study comparing Convolutional Networks (CNNs) and Multilayer Perceptrons (MLPs) on the same dataset, this investigation focuses specifically on evaluating the efficacy of SAEs on the KMNIST set

The autoencoder has two key components: the encoder and the decoder. The *encoder* maps the input to a hidden representation, the bottleneck layer. The *decoder* takes this hidden representation and maps it to the output.

Autoencoders can be trained end-to-end, or layer by layer, in a *stacked* format. In this project, we look closer at the *stacked* autoencoder (SAE), which consists of several layers of autoencoders. Part two delves into regularized SAEs, enhancing generalization by focusing on the most salient features and reducing computational load. This efficiency benefits tasks like feature learning, anomaly detection, and data compression [7]. Unlike sparse autoencoders, which prioritize efficiency and sparsity, SAEs emphasize depth and hierarchical feature learning [6]. Incorporating a regularizer in the SAE's reconstruction cost function optimizes extracted codes, by minimizing their distance to a predefined set of 'constellation' targets, akin to communication systems.

In both project segments, we applied bottleneck regularization, compressing the representation dimensions smaller than the input [1]. In the first part, a deep SAE with five layers, utilizing Mean Squared Error (MSE), projected data into a subspace. Extracted features from the bottleneck layer served as inputs to a support Vector Machine (SVM) classifier. We conducted a series of experiments to ascertain the best configuration, comparing it against the MLP classifier results from the previous project.

To enhance complexity, impulsive noise was introduced. The SAE was evaluated with both the MSE and Correntropy cost functions. Using a correntropy cost function improved noise cleanup, validated by better lower reconstruction error, as well as higher accuracy when combined with the SVM classifier. Correntropy, a specialized form of cross entropy, excels in handling outliers with a proper kernel bandwidth [2]." It is a nonlinear and local comparison measure that shows the similarity between two random variables.

At the culmination of our research, we reduced the bottleneck size to 10, directly using the codes as class assignments in the test set.

This collaborative effort, shared between two students, found that the first part of the project obtained a test accuracy of 92.6% for the original SAE, 92.2% with added noise, 92.2% with added noise and correntropy, 93% with the regularizer, and 78% with only 10 as the bottleneck size. We employ confusion matrices to elucidate the performance differences among the various models and methodologies. Additionally, we assess the computational efficiency of each approach.

This paper is organized as follows: Section II details the precise methodology used in our experimentation process. Section III describes the results of the experiment. Section IV provides a qualitative and quantitative analysis of the results, listing limitations and future work. Finally, Section V concludes this paper.

II. Methodology

Our methodology involves two primary components. Firstly, we design and train a traditional SAE with five hidden layers, optimizing its bottleneck layer for feature extraction, which feeds into a Support Vector Machine (SVM) classifier. A series of systematic experiments guide the optimization of the bottleneck layer and other hyperparameters. Secondly, we introduce a penalty function into the SAE's reconstruction cost to enhance discrimination in the latent space.

A. *Part 1*

B. *Standard Architecture*

The standard SAE architecture comprised 5 hidden layers with ReLu activation (800-200-xxx units), where 'xxx' denotes the bottleneck layer size, a hyperparameter. The ReLu activation function is often chosen because it increases sparsity. The Input and output size was 784 (flattened images). Early stopping (patience of 5) prevented overfitting, with a maximum of 100 epochs. The SAE was combined with an SVM classifier, leveraging the SVM's advantages in high-dimensional spaces through the kernel trick. Exploration mainly focused on the SAE, evaluating two key SVM parameters (C and gamma) with other parameters set to standard values. Sci-kit learn's SVM library, employing an rbf kernel with a degree of 3 and an L2 regularizer, facilitated implementation.

1) hyperparameter search: To optimize SAE results and ensure consistency across experiments, hyperparameters were evaluated initially for the original SAE. Once determined, these hyperparameters were fixed for subsequent models, facilitating a standardized evaluation process.

- Bottleneck size: Given that there are 10 classes, the bottleneck layer size ranged from the smallest permissible value of 10 to a maximum constrained by the previous hidden layer, which is 200. To streamline the search and enhance efficiency, square values within this range were chosen, allowing for interval evaluations. The choice of square values facilitated both visualization and comprehensive evaluation, assessing the bottleneck layer not only for reconstruction error but also its efficacy in image classification using an SVM classifier.
- Batch size: To enhance computational efficiency while ensuring robust reconstruction, we systematically assessed batch sizes from 32 to 448 at intervals of 32. The choice of a 32-unit interval aligns with hardware considerations, as it is often more efficient to work with batch sizes in this increment. Using this large of an interval also accelerates the exploration of a diverse range of batch sizes within a condensed time frame. The upper bound of the search region, set at 448, was strategically chosen to result in approximately 100 passes per epoch. This marked a significant reduction from the 1500 passes associated with a batch size of 32. The goal was to identify the point at which training time could be minimized without compromising accuracy. The selected range was anticipated to be expansive enough to capture this optimization.
- Optimization Algorithm and Learning Rate: The choice of the Adam optimization algorithm was motivated by its adaptive learning rate capability, a key factor in ensuring model convergence. Adam's three main parameters—Learning Rate, Beta1, and Beta2— were evaluated. The Learning Rate sets the initial rate to be adjusted by Adam, striking a balance between achieving faster convergence and mitigating the risk of overshooting the minimum. Beta1 influences the mean of the exponential decay rate, controlling the influence of recent gradients on new updates, while Beta2 controls the variance of the decay rate, aiding in adjusting the step size for each update. Learning rates in the range of [0.001, to 1.0] were evaluated, capturing both relatively low and high learning rates. For Beta1, values in the range [0.7, 0.9] were assessed, and for Beta2, values within [0.8, 0.999] were explored. These value ranges slightly deviate from standard values as the primary aim was to investigate whether nonstandard values could yield improved results.
- Classifier: The two hyperparamaters that were evaluated for the SVM classifier is C and gamma. C controls the weight of the L2 regularizer and is inversely proportional to the regularizer. Gamma is the coefficient of the 'rbf' kernel and influences the effect each training sample has on the decision boundary, a smaller gamma will result in a smoother decision boundary. Gamma Values between 0.001 and 10 were evaluated for both Gamma and C.

2) Add impulsive noise: To assess the impact of introduced noise on input images, a curated set of images was generated by incorporating impulsive noise. Specifically, images were altered with a 10% probability of introducing impulsive noise using a mid-range grey color. Examples of the unaffected images and noisy images can be seen. In the context of this experiment, the input images for the (SAE) comprise the noisy images, while the target images correspond to their noiseless counterparts. Once again, the codes created from the bottleneck layer were fed into a SVM classifier. The reconstruction error from the SAE as well as the classification results were evaluated.

3) Cost function: Two different cost functions were evaluated for the Noisy Image SAE. The first cost function was Mean Squared error and was used in the design of the original SAE as well as the SAE with the noisy images. The second cost function which was used in the SAE with the noisy images was Correntropy (equation 1).

$$V(X,Y) = \int_e k(e) f(e)\, de \tag{1}$$

Where K denotes the Gaussian Kernel. The kernel size is controlled by the sigma parameter, which indicates the bandwidth of the Gaussian kernel. As sigma increases the Gaussian curve becomes smoother and covers a broader range of points. A larger sigma can lead to a more flexible model that may be prone to capturing noise which can also increase the risk of over-fitting. Silverman's method was applied, which is often used to determine bandwidth selection in kernel density estimation. Silverman's method resulted in a value of 3.4. Values surrounding this were evaluated from 0.001 to 10. In the literature it suggest using mse or the first 10 to 20 epochs since correntropy is convex and can be slow to converge. In this method the first 10 epochs were evaluated using mse any training beyond that was done using correntropy.

C. Part 2

In the second part of our study, we augmented our approach by incorporating a novel regularization strategy into the SAE's cost function. This approach aimed to amplify the discriminative power within the autoencoder's latent space by minimizing the distance between the learned representations and a set of predetermined targets, akin to the concept of constellations used in communication systems.

The algorithmic steps for implementing this enhanced methodology were as follows:

1) Initialization of the Enhanced Autoencoder with a bottleneck layer, maintaining the same size determined in Part 1.
2) Preprocess the input data through normalization and flattening methods to prepare it for processing by the autoencoder.
3) Introduce the regularizer
   - Define a custom loss function incorporating the reconstruction error and the regularization term.
   - The regularization term computes the distance between the bottleneck layer's current code and its target constellation point, modulated by a lambda (λ) parameter.
4) Compute the target constellation points for each class by averaging the bottleneck features of all samples belonging to that class.
5) Build the model training function using the combined cost function, which includes the MSE reconstruction loss and the regularization term.
6) Implement early stopping based on validation loss.
7) Conduct a hyperparameter grid search, experimenting with various lambda values and distance metrics to identify the combination that yields the lowest validation loss.
8) Plot the impact of different lambda values on the loss using 3D visualization techniques to elucidate the trade-offs and interactions.
9) With the optimal parameters identified, retrain the autoencoder on the full training dataset.
10) Calculate the reconstruction error on the training set.
11) Train an SVM classifier on the bottleneck features derived from the autoencoder.
12) Evaluate the autoencoder and SVM classifier on the test set.

This model was architecturally defined with a specific focus on its bottleneck layer, and it was trained using a custom loss function (Step 3). This function was a blend of MSE (Equation 2) for reconstruction, which measures reconstruction fidelity, and a regularization term (Equation 3). This term penalizes the distance between the bottleneck representations and predetermined target points for each class. The latter was a strategic addition, quantifying the distance between the bottleneck features and predefined target points for each class, referred to as the *constellation targets*.

$$L = \frac{1}{N} \sum_{i=1}^{N} ||x_i - \hat{x}||_2^2 \tag{2}$$

The above formula (Equation 1) represents reconstruction loss using MSE where x is the original input and $\hat{x}$ is the reconstructed output. The formula below (Equation 3) represents the regularization term where K is the batch size, and d is the distance metric to be systematically selected by the coder.

$$R = \frac{1}{K} \sum_{i=1}^{K} d_i \tag{3}$$

The result of combining these two gives us the new cost function:

$$J_{new} = L + \lambda R \tag{4}$$

To calculate the target constellation points for each class (Step 4), we averaged the bottleneck features of all training samples belonging to that particular class. These points served as representations in the latent space, guiding the regularization process. For Step 5, the model training function was built using our custom loss function. The regularization term's impact was controlled by a lambda parameter, tuned in Step 7, and finding the most effective distance metric. Using a grid search, we tested a range of values for lambda between 0.001 and .5 against the three most commonly used distance metrics in machine learning:

$$\textbf{MSE}: \; d(b_i, c_{y_i}) = ||b_i - c_{y_i}||_2^2 \tag{5}$$

$$\textbf{L1}: \; d(b_i, c_{y_i}) = ||b_i - c_{y_i}||_1 \tag{6}$$

$$\textbf{Cosine}: \; d(b_i, c_{y_i}) = 1 - \frac{b_i \cdot c_{y_i}}{||b_i|| \cdot ||c_{y_i}||} \tag{7}$$

Typically, small lambda values give more weight to reducing the training error, while larger lambda values emphasize smaller model parameters, which is why we tested the range of values that we did.

The classifier was trained on the bottleneck features to perform the final classification task. We evaluated the classifier's accuracy using standard metrics (i.e., accuracy score) and confusion matrices, providing a clear picture of the model's classification performance. We then repeated all of these steps (aside from the hyperparameter search) for a model with a bottleneck size of 10.

III. Results

A. Part 1 Results

1) Hyperparameter Search: Each parameter underwent evaluation based on two key metrics: computation time and reconstruction loss. A grid search methodology was employed for the bottleneck layer and batch size, comprehensively exploring different hyperparameter combinations. The initial phase considered these parameters together, followed by a focused analysis evaluating each hyperparameter individually while keeping others constant. This dual approach provided insights into the distinct impact of each hyperparameter on the specified metrics, offering a comprehensive understanding.

- Bottleneck size: As the bottleneck size increased, model size and computation time rose, and reconstruction error decreased. However, classification accuracy did not significantly improve beyond a bottleneck size of 81, as seen in Figure 1. Confusion matrices in Figure 2 revealed that a bottleneck layer of 196 provided near-perfect results, with minimal change at size 81. Below 36, misclassifications notably increased. In Figure 3, increasing the bottleneck size beyond 100 led to a substantial rise in computation time (200-500 seconds) with marginal reconstruction loss reduction (approximately 0.0025 to 0.0005). The increase in bottleneck size significantly raised model parameters, posing a risk of overfitting. Notably, the grid search corroborated the trends observed in the individual search. The grid search aligned with individual search trends, highlighting a significant decrease in the slope at a bottleneck size of 121. Considering model size, computation time, reconstruction error, and SVM accuracy, the optimal bottleneck size was 81. This choice balances minimal reconstruction loss, manageable computation time, and high-accuracy results.
- Batch size: As batch size increases, computation time decreases. The optimal batch size, chosen just before reconstruction error began to decrease, was determined to be 320, as illustrated in Figure 4.
- Optimization Algorithm: The learning rate was chosen to be 0.001 because it struck a balance between lower reconstruction error and improved computational efficiency. As the learning rate increased, the computation time decreased exponentially, as shown in Figure 5. Both beta 1 and Beta 2 were also evaluated against reconstruction error. It was found that the standard values of 0.9 for beta 1 and 0.999 for beta 2 were optimal 6.
- SVM parameters: The regularizer's parameter, C, and kernel parameter, gamma, were evaluated based on the SVM's accuracy results. The optimal C value was 10 with a gamma value of 0.1. These results can be found in figure 7.
- Correntropy Kernel Size: The resulting plot 8 shows that as sigma increased, reconstruction error also increased. A value of 0.5 was chosen as the optimal sigma value since it resulted in the lowest reconstruction error.

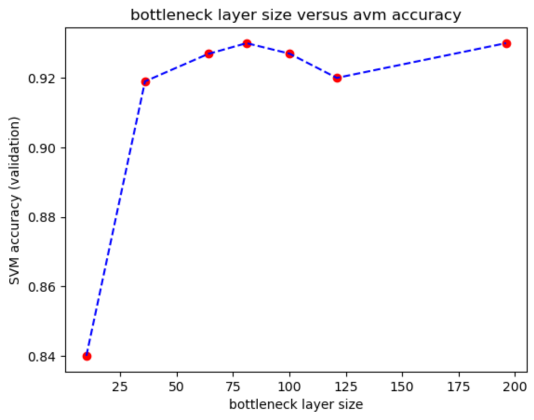

Fig. 1: Scatter plot representing the change in the classifier's accuracy as the bottleneck layer size changes.

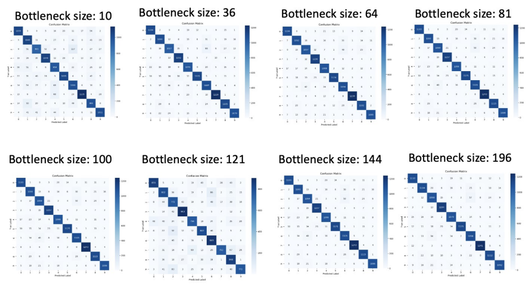

Fig. 2: Confusion Matrices describing different classification results for various bottleneck sizes.

B. Test Results

Once the hyper-parameters were optimized, each of the SAE's bottleneck layer output codes were utilized to train an SVM Classifier. The test set was then used to evaluate the SAE and the classifier. The confusion matrices for the test set can be found in figure 9, and training and test results can be found in I.

C. Part 2 Results

In our comprehensive hyperparameter search, we evaluated various lambda values and distance metrics to optimize the performance of our autoencoder. The tested lambda values ranged from 0.001 to 0.5, and we experimented with three different distance metrics: Mean Squared Error (MSE), L1 norm, and Cosine distance. The combination of these parameters was assessed based on the loss they achieved.

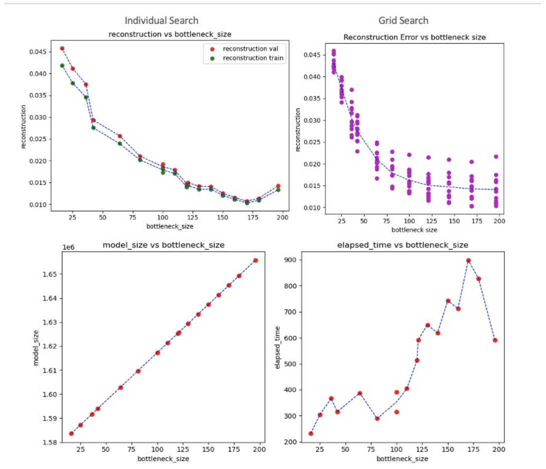

Fig. 3: Scatter plots representing the individual and grid searches for bottleneck size versus computation size, computation time, and reconstruction loss.

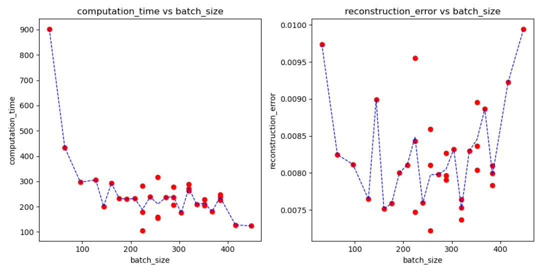

Fig. 4: Scatter plots representing the individual search for batch size versus computation time and reconstruction loss.

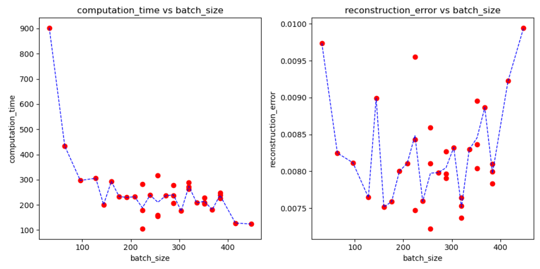

Fig. 5: Scatter plots representing the individual search for Learning Rate versus computation time and reconstruction loss.

The optimal parameters were identified as a lambda of 0.001 and the MSE distance metric, achieving the lowest loss of 2.2%. Utilizing these parameters, our model demonstrated a final loss of 0.14% with a bottleneck size of 81 (obtained from part 1). In examining Fig 10, the differences in loss aren't overly significant, but we can still see that for lower values of lambda, we achieved minimal loss with the MSE function. Additionally, we provide a three-dimensional visualization of the relationship between the lambda parameter, the regularizer R, and the reconstruction error in our stacked autoencoder model 11. Data points across the plot show how varying levels of lambda and R correlate with changes in the reconstruction error, indicated by the color gradient, with darker colors representing lower errors.

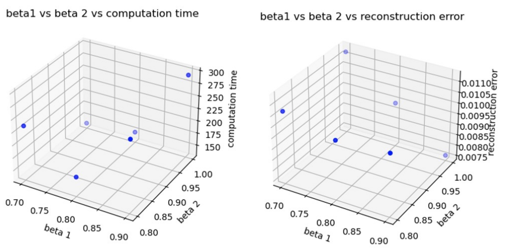

Fig. 6: 3D plots representing Beta 1 and Beta 2, for Adam optimizer, exploration

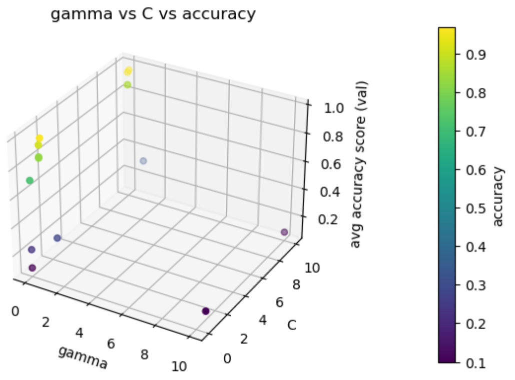

Fig. 7: 3D plot representing the explored gamma and C values versus the classifier accuracy.

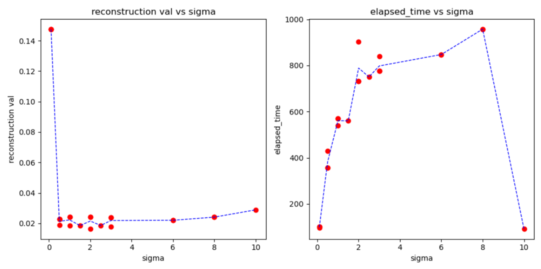

Fig. 8: Scatter plots representing the explored sigma values for correntropy.

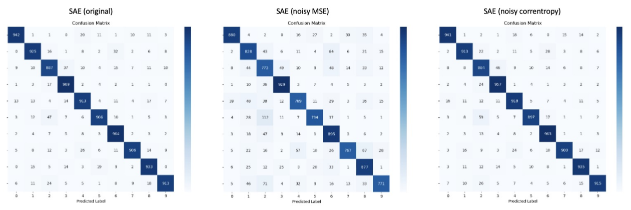

Fig. 9: Confusion Matrices for the three methods in Part 1 (Original SAE, SAE trained with Noisy images and MSE, SAE trained with noisy images and correntropy)

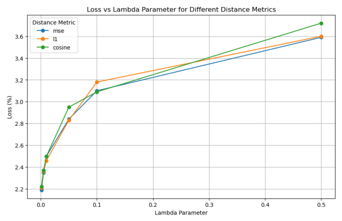

Fig. 10: Scatterplot representing the accuracies per each hyperparameter

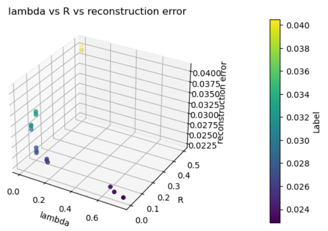

Fig. 11: A 3D plot showing how the choice of lambda and R affects the error in rebuilding data in our model, with darker colors meaning less error.

After training the model with these optimal values, the average reconstruction loss recorded was 0.02%. Further, when assessed through a Support Vector Machine (SVM), the trained model's performance exhibited high accuracy. The accuracy on the training data reached 99.9%, while the test set accuracy was 93.2%.

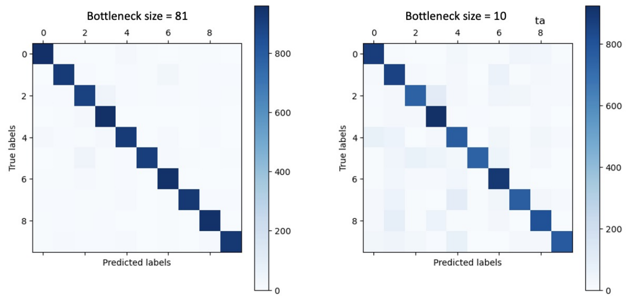

Fig. 12: Confusion Matrices utilizing test set for Part 2

To explore the impact of the bottleneck size on model performance, we conducted an additional experiment with a bottleneck size of 10. In this setup, the final loss increased to 4.62%, and the average reconstruction loss was 2.00%. Correspondingly, the SVM accuracy on the training data decreased to 91.7%, and the test accuracy reduced to 78.0%.

Overall, the SAE with regularization performed the best, so long as it was trained with the optimal bottleneck size. Table I shows a visualization of all accuracies. The confusion matrices in figure 12 compare the classification performance of two stacked autoencoder models with different bottleneck sizes. The left matrix, with a bottleneck size of 81, shows a higher concentration of values along the diagonal, indicating more accurate predictions. With a bottleneck size of 10, the right matrix exhibits lighter shades along the diagonal and more spread, suggesting less accurate classification outcomes.

D. Run Time

The runtimes for each method in parts 1 and 2 were compared in table II. The techniques with the regularizers increased the training time significantly. The training was fastest for the original SAE in part 1 and longest for the SAE with the new regularizer in part 2. The SVM training times were all relatively similar, except for the method with the bottleneck size of 10, illustrating how valuable it is to fine-tune your hyperparameters.

IV. Discussion

This project's primary objective was to enhance feature extraction and classification accuracy on the Kuzushiji-MNIST dataset through advanced autoencoder architectures. Both Part 1 and Part 2 incorporated class labels in the training process but approached the task with different methodologies and enhancements.

A. Part 1

In Part 1, a Stacked Autoencoder (SAE) with variable bottleneck layers was employed, with the SAE outputs used as inputs for SVM or MLP classifiers. This part of the study was crucial in establishing baseline performance and understanding the impact of bottleneck layer dimensionality on feature extraction efficacy.

1) SAE + Classifier compared to MLP: Both Project One's MLP and the SAE featured 7-layer architectures, with the MLP having three hidden layers and two dropout regularization layers. With five hidden layers, the SAE had significantly more parameters (over 800,000 in the autoencoder) than the MLP's 250,000. The SAE+classifier achieved higher accuracies in training and test. 97% compared to MLP's 93% in training and 92.5% compared to MLP's 89% in test. However, the SAE+Classifier exhibited a more significant drop in accuracy between training and test, suggesting potential overfitting. Additional regularization may benefit the SAE+Classifier model.

2) MSE compared to correntropy: Adding impulsive noise and using the correntropy cost function illustrated the robustness of the SAE model against data imperfections, a common challenge in real-world datasets. Correntropy effectively improved the image reconstructions (figure 13) and classification accuracy for several reasons. First, correntropy is particularly effective at dealing with non-Gaussian noise, like impulsive noise, which is common in real-world data [3]. Additionally, by incorporating correntropy, the SAE learns to extract features that are not just representative of the data but also resilient to noise, leading to improved generalization in noisy environments.

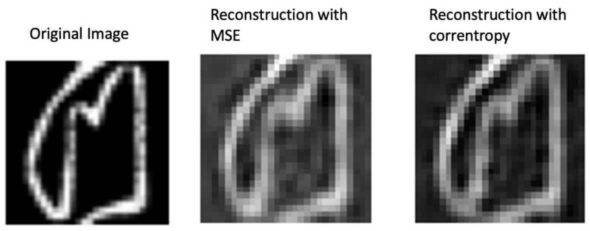

Fig. 13: Image reconstruction examples using the MSE cost function and the correntropy cost

3) Part 1 methods compared: Looking at the accuracy metrics in Table I and the confusion matrices for each of the methods in part 1, it is clear the original SAE + Classifier had the least amount of misclassified i mages, w hich i s probably

because the test images were similar to the training images. The generalization of the models from adding the noise would be more apparent if we were to utilize unrelated testing images that contained noise. Moreover, the correntropy cost function had better results than the MSE cost function for the SAE with noisy images.

4) Target Constellations: The target constellations were created using each class's mean of the codes. Another approach would be utilizing a latent space clustering method to create maximally discriminative codes. This method was evaluated using kmeans, resulting in multiple codes for some labels and no codes for others. This is because the parameters are optimized to improve image reconstruction, which does not necessarily correlate to class separability. This was an important takeaway because our initial thought was that lower reconstruction error in the SAE would lead to better accuracy results. However, this is not always true. Therefore, the classifier had to be considered in the search for the optimal parameters. This is also why the regularizer in part 2 benefits the classification results.

5) Part 2 methods compared: The success of Part 2 in achieving better results than Part 1 can be attributed to a more targeted approach in organizing and discriminating the feature space, which is a testament to the effectiveness of informed, class-based regularizations in autoencoder training. Additionally, regularization often adds a penalty for complexity, encouraging the model to learn more efficient and essential features, reducing the risk of over-fitting to the training data [5]. Looking at the confusion matrices for the part 2 methods, it is clear the method with a bottleneck layer of 81 had fewer misclassified images, and a bottleneck layer of 10 achieved optimal results. This makes sense because using a bottleneck layer of 10 is essentially finding one grayscale value to determine the entire class of images, which could be based on how much writing is in each image but could also be significantly affected by things like light. This low dimensional space led to severe overfitting and resulted in inferior results for the test set.

It should also be noted that for all these methods, the order in which data samples were presented in the SAE and SVM were random to help produce more generalized results. Additionally, since all of these methods are nonlinear, the initialization of each method could lead to a different optimization. Each method was initialized multiple times to ensure the change in the results was not significant due to the initialization.

B. Limitations

The most significant limitation of this project was the time constraint. Provided more time, we could have performed a more exhaustive hyperparameter search to improve model accuracy. Hyperparameter tuning can be time-consuming and may not yield optimal results for different datasets or tasks. While effective for this specific study, the chosen parameters might not be universally applicable.

dataset. While this dataset provides a robust foundation for analysis, the results may not generalize to datasets with different characteristics or real-world scenarios with more complex or varied data. Finally, while the model showed improved performance with the introduction of impulsive noise and a correntropy cost function, it remains untested against other noise or data corruption types, which could affect its practical utility. To evaluate several parameters in a short period, in general, the hyperparameters were separately assessed; however, I would like to do more evaluation of the interaction of the hyperparameters together.

V. Conclusion

The findings o f t his p roject u nderscore t he i mportance of class-informed training in autoencoder-based feature extraction models. While both parts of the project utilized class labels, Part 2's method of directly influencing t he latent space organization through a regularizer proved to be more effective. This approach enhanced classification a ccuracy and contributed to a more structured and meaningful latent representation.

Our experiments with different noise levels and cost functions highlight the adaptability of the SAE to varying data conditions, a crucial aspect for practical applications. Using correntropy in noise management and adding a regularizer in the latent space are both innovative steps that significantly enhance the model's capability.

This study contributes to the ongoing feature extraction and classification research, providing insights into the efficient design of autoencoder architectures for improved classification performance. The success of the regularized SAE in Part 2 opens up new avenues for future research, particularly in developing more sophisticated regularizers that can further refine the feature extraction process for various machine learning tasks.

References

[1] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, pages 353–374, 2023.

[2] Jin Chen and Sheng Guan. Correntropy-based doa estimation algorithm under impulsive noise environments. *EURASIP Journal on Wireless Communications and Networking*, 2020(1):1–20, 2020.

[3] Weifeng Liu, Puskal P Pokharel, and Jose C Principe. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on signal processing*, 55(11):5286–5298, 2007.

[4] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[5] Harald Steck and Dario Garcia Garcia. On the regularization of autoencoders. *arXiv preprint arXiv:2110.11402*, 2021.

[6] Jianbo Yu, Xiaoyun Zheng, and Shijin Wang. A deep autoencoder feature learning method for process pattern recognition. *Journal of Process Control*, 79:1–15, 2019.

[7] Qiang Zhao and Fakhri Karray. Anomaly detection for images using autoencoder based sparse representation. In *Image Analysis and Recognition: 17th International Conference, ICIAR 2020, Póvoa de Varzim, Portugal, June 24–26, 2020, Proceedings, Part II 17*, pages 144–153. Springer, 2020.

Method	**Final Loss (%)**	**Avg. Reconstruction Loss (%)**	**Train SVM Accuracy (%)**	**Test SVM Accuracy (%)**
SAE (Original)	2.5	2.5	97.3	**92.6**
SAE (with noise, MSE)	2.2	2.2	97.2	**92.2**
SAE (with noise, Correntropy)	1.9	1.8	97.2	**92.4**
SAE with regularizer (81)	0.14	0.02	99.9	**93.2**
SAE with regularizer (10)	0.5	2.0	91.7	**78**

TABLE I: Comparison of Model Accuracies in SAE models

Method	**Avg. Train Time SAE**	**Avg. Train Time SVM**
SAE (Original)	5.3 min	5 min
SAE (with noise, MSE)	8.9 min	5.2 min
SAE (with noise, Correntropy)	13.3 min	5.3 min
SAE with regularizer (81)	15.34 min	5.64 min
SAE with regularizer (10)	8.51 min	1.42 min

TABLE II: Comparison of Training times for each method