EEL6814: Project 2

line 1: 1st Given Name Surname   
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID  
  
Joseph Madden  
*Electrical and Computer Engineering*   
*University of Florida*Gainesville, FL USA  
josephmadden@ufl.edu  
  
line 1: 3rd Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID  
  
line 1: 4th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 5th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 6th Given Name Surname  
line 2: *dept. name of organization (of Affiliation)*  
line 3: *name of organization (of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

*Abstract*—This report presents the development and evaluation of a Stacked Autoencoder Network (SAE) integrated with a Multi-Layer Perceptron (MLP) classifier for application on the Kuzushiji-MNIST dataset. The project aims to explore feature extraction for image classification by experimenting with various SAE configurations, particularly focusing on the dimensions of the bottleneck layer and network hyperparameters. Additions to this model include adding impulsive noise in the input data and employing a correntropy cost function in place of the traditional Mean Squared Error (MSE). Furthermore, the study introduces a distinctive penalty function in the SAE’s reconstruction cost, designed to create discriminative code targets in the latent space, thereby enhancing classification accuracy. Comparative analysis using confusion matrices and computation time evaluations assesses the performance of the SAE+Classifier model against a baseline MLP classifier. The outcomes underscore the model's enhanced noise resilience and superior classification efficiency, underscoring its potential in refining deep learning strategies for image classification tasks.

Keywords—stacked autoencoder network, image classification

# Introduction

In the realm of machine learning, deep learning techniques have revolutionized the approach to complex tasks, particularly in image classification. Among these techniques, Stacked Autoencoder Networks (SAE) have emerged as a powerful tool for feature extraction, enabling the transformation of high-dimensional data into a more manageable and representative form. This report focuses on the design and evaluation of an SAE integrated with a Multi-Layer Perceptron (MLP) classifier, applied to the Kuzushiji-MNIST dataset.

The Kuzushiji-MNIST dataset, a variant of the traditional MNIST dataset, presents unique challenges due to its intricate patterns and historical significance in Japanese literature. This project aims to explore the potential of SAE in enhancing feature extraction capabilities for this complex dataset. Central to this exploration is the investigation of various configurations of the SAE, particularly the impact of the size of the bottleneck layer and the choice of hyperparameters on the overall classification performance.

Moreover, this study delves into the robustness of the SAE under conditions of impulsive noise. By substituting the traditional Mean Squared Error (MSE) with a correntropy cost function, the project evaluates the network's capacity to handle noise in input data, a crucial aspect for real-world applications. Additionally, the introduction of a novel penalty function in the SAE's reconstruction cost aims to foster discriminative feature learning, potentially enhancing classification accuracy.

The comparative analysis of the SAE+Classifier model against a standalone MLP classifier offers insights into the effectiveness of the proposed methods. This report documents the methodology, findings, and implications of these innovations, contributing to the ongoing research in the field of deep learning and image classification.

# Methodology

This study develops and evaluates a SAE with a MLP classifier for the Kuzushiji-MNIST dataset. The approach involves optimizing the SAE, particularly its bottleneck layer and hyperparameters, and adapting it to handle impulsive noise using a correntropy-based cost function. Additionally, a specialized penalty function is implemented to enhance feature discrimination.

## Creating the Stacked Autoencoder Network

The architecture of the Stacked Autoencoder Network (SAE) was crafted using Python, employing the Keras-Tensorflow libraries. The network is structured with an input layer, five hidden layers, and an output layer. The input layer, the first two hidden layers, and the bottleneck layer are all stored in one *Sequential* model and the remaining layers are stored in another. This split design is to make the later SAE+Classifier model easier to design.

The input layer is configured to handle arrays of 784 pixels, which correspond to 28x28 pixel grayscale images, with each pixel normalized within a 0 to 1 range. The output layer is the same, outputting the decoded image. The hidden layers are arranged in the sequence of 800-200-100-200-800 units. The central layer, hereafter known as the bottleneck layer, initially comprises 100 units and is designated as a hyperparameter for experimental variation. The activation function of the hidden layers is the Rectified Linear Unit (ReLU) function.

## Tuning the SAE

Identify applicable funding agency here. If none, delete this text box.

The training process for the Stacked Autoencoder Network (SAE) is designed to optimize its performance using a substantial portion of the image data. Specifically, 70% of the image data is allocated for training, while the remaining 30% is set aside for validation purposes. To ensure a balanced representation, both the training and validation sets are stratified based on their class labels. The KMNIST dataset creators already provide the test set.

Key to the training process are the hyperparameters: the width of the bottleneck layer and the batch size for each training epoch. To efficiently tune these hyperparameters, KerasTuner from the Scikit-Learn library is utilized. The width/height of the bottleneck layer will vary in multiples of 5, ranging from 50 to 100 inclusive. For the batch sizes, the range considered is between powers of two from 32 to 256 inclusive. This approach results in a total of 44 distinct trials, allowing for a comprehensive exploration of the hyperparameter space.

Each model iteration is granted a maximum of 20 epochs for convergence, with the incorporation of callbacks to terminate training early. The specific callback criterion is set to halt training if there's no improvement of at least 0.05 in the loss over a span of 3 epochs. The chosen loss functions for this training are MSE and correntropy (tested separately), and the ADAM optimizer from the Keras-Tensorflow library is employed for learning. The labels for the training data will be the same as the samples for the training data.

## Designing the SAE+Classifier

The encoder extracted from the Stacked Autoencoder Network (SAE) was integrated with a Multi-Layer Perceptron (MLP) classifier, forming a comprehensive classification model. This integration was also facilitated using the Keras-Tensorflow libraries.

The input layer of the MLP is aligned with the width of the bottleneck layer from the SAE, ensuring seamless integration and data flow between the encoder and the classifier. The output layer of the MLP classifier consists of 10 perceptrons, each corresponding to one of the classes in the dataset. The number and width of hidden layers is a hyperparameter and discussed in the following sub-section.

For the hidden layers, the Rectified Linear Unit (ReLU) function is chosen as the activation function. The output layer employs the softmax activation function, standard for classification problems. Additionally, the hidden layers of the MLP classifier are constructed using 'Dense' layers from the Keras-Tensorflow library.

## Tuning the SAE+Classifier

The training process for the integrated SAE+Classifier model is meticulously designed to optimize its classification performance. The classifier component, a Multi-Layer Perceptron (MLP), is configured with several hyperparameters: the number of hidden layers, the number of perceptrons in each layer, and the batch size for training.

For the MLP structure, the number of hidden layers is varied among three options: 1, 2, or 3 layers. The perceptron count in each hidden layer can be one of the following: 5, 10, 15, 20, 100, 200, or 400, with uniformity across all hidden layers in any given trial. This uniformity is a constraint to reduce the search space. The choice in hidden layers is to make comparisons in network size and EEL6814: Project 1 possible. The batch sizes considered for training are 32, 64, 128, and 256.

Same as the SAE, the MLP has a maximum of 20 epochs to converge, with callbacks to end training early. The training will end early if there is not an improvement of at least 0.05 in the loss over a span of 3 epochs. ADAM is the chosen optimizer again.

The encoder from the SAE is kept static—its weights are not trained further during this phase—and its outputs are fed as inputs into the MLP classifier. This approach ensures that the feature extraction capabilities developed in the SAE are not lost while training the MLP classifier.

Moreover, to accommodate the variations in the encoder training, separate tuning instances are established for encoders trained on Mean Squared Error (MSE) and those trained on correntropy. This distinction allows for a comparative analysis of the impact of different loss functions on the overall model performance.

## Adding Noise

As part of the examination of the SAE and its classifier, noise will be added to the training images to observe the effect on the model.

### Modifying the Training Data

The procedure begins by adding impulsive noise to the training images. This is executed with a predefined probability of 10%, targeting the alteration of pixels. The noise introduced is set at a value of 167, representing a mid-range gray, to provide a noticeable yet representative level of distortion.

### Re-Tuning the SAE

Following the noise addition, the SAE undergoes a re-tuning process. This tuning adheres to the same methodology as outlined in the "Tuning the SAE" section. The objective here is to assess and optimize the network’s ability to process and extract features from the noised data, maintaining the integrity of the information despite the added complexity.

### Encoding and MLP Model Training

The encoded values, now derived from the noise-adapted SAE, are subsequently fed into new MLP models. These models are tuned in accordance with the procedures described in the "Tuning the SAE+Classifier" section. The focus here is to evaluate how the noise-influenced encoded data impacts the classification accuracy and to fine-tune the MLP classifiers accordingly.

# Results

## Hyperparameter Tuning without Noise

### SAE with MSE Loss Function

For the SAE trained on MSE, the best hyperparameters were a bottle neck width of 100 PEs and a batch size for training of 64.

A white background with black dots

Description automatically generated

Fig.1. Image input to MSE SAE

A white background with black dots

Description automatically generated

Fig.2. Image output from MSE SAE

### SAE with Correntropy Loss

For the SAE trained on correntropy, the best hyperparameters were a bottle neck width of 100 PEs and a batch size for training of 64. The final validation score for this configuration was a bottle neck width of 100 PEs, a batch size of 32, and a sigma value of 1.0.

A white background with black dots

Description automatically generated

Fig.3. Image input from correntropy SAE

A white background with black dots

Description automatically generated

Fig.4. Image output from correntropy SAE

### MLP Classifier with MSE Encoder

For the MLP classifier appended to the MSE encoder, the classifier had a width of 100 PEs, two hidden layers, a training batch size of 128, and a bottle neck width of 100 to connect to the encoder.

A chart of a number of different colored squares

Description automatically generated with medium confidence

Fig.5. Confusion matrix for MSE SAE+Classifier (No Noise)

### MLP Classifier with Correntropy Encoder

For the MLP classifier appended to the correntropy encoder, the classifier had a width of 100 PEs, one hidden layer, a batch size of 128, and a neck width of 100 to connect to the encoder.

A chart with numbers and labels

Description automatically generated with medium confidence

Fig.6. Confusion matrix for correntropy SAE+Classifier (No Noise)

## Hyperparameter Tuning with Noise

### SAE with MSE Loss Function

For the SAE trained on MSE, the best hyperparameters were a bottle neck width of 100 PEs and a batch size for training of 128.

A white background with black dots

Description automatically generated

Fig.7. Noisy image input to MSE SAE

A white background with black dots

Description automatically generated

Fig.8. Noisy image output from MSE SAE

### SAE with Correntropy Loss

For the SAE trained on correntropy, the best hyperparameters were a bottle neck width of 100 PEs and a batch size for training of 128. The final validation score for this configuration was a bottle neck width of 100 PEs, a batch size of 32, and a sigma value of 1.0.

A white background with black dots

Description automatically generated

Fig.9. Noisy image input to correntropy SAE

A white background with black dots

Description automatically generated

Fig.10. Noisy image output from correntropy SAE

### MLP Classifier with MSE Encoder

For the MLP classifier appended to the MSE encoder, the classifier had a width of 100 PEs, three hidden layers, a training batch size of 64, and a bottle neck width of 100 to connect to the encoder.

A chart with numbers and labels

Description automatically generated with medium confidence

Fig.11. Confusion matrix for MSE SAE+Classifier (Noise)

### MLP Classifier with Correntropy Encoder

For the MLP classifier appended to the correntropy encoder, the classifier had a width of 100 PEs, one hidden layer, a batch size of 64, and a neck width of 100 to connect to the encoder.

A chart with numbers and labels

Description automatically generated with medium confidence

Fig.12. Confusion matrix for correntropy SAE+Classifier (Noise)

## Varying Bottleneck Size

These results are from an MSE SAE+Classifier with all other hyperparameters held constant. The constant values were selected from the values chosen during hyperparameter tuning. A complete list of confusion matrices from 50-100 PEs can be found in the Jupyter Notebooks code.

A chart with numbers and a comparison of the same color

Description automatically generated with medium confidence

Fig.13. Confusion matrix for bottleneck\_width=50

A chart with numbers and a graph

Description automatically generated with medium confidence

Fig.14. Confusion matrix for bottleneck\_width=100

# Discussion

When tuning the different SAE+Classifier models, there were some trends in the hyperparameters worth noting. The 100 PEs in the bottleneck layer was consistently the best choice for that hyperparameter. Worth noting is that other PE amounts yielded similar results (approximately >0.05 difference between them), yet there was a consistent downward trend in loss as more PEs were added. It would seem that 50 PEs in the bottleneck layer is sufficient to get highly predictive features with diminishing returns as the number increases.

This conclusion is further evidenced from the results of testing multiple bottleneck layer widths. The confusion matrices from each configuration were very similar and the accuracy metrics were close as well.

A 100 PEs in the classifier MLP for every model is another trend that makes sense. As referenced in the project 1 paper, a good choice for hidden layers is the sum of the number of inputs and half of the number of outputs [1]. This would be 105, which is closest to 100 in our hyperparameter tuning.

In regards to adding noise to the data, the results show that correntropy outperforms MSE for training the SAE for the SAE+Classifier. This is likely due to the nonlinearity of the correntropy loss function. The gaussian kernel assigns less weight to outliers whereas MSE assigns weight uniformly to outliers.

# Conclusion

SAEs are capable of automatic feature engineering, utilizing nonlinearities to find hierarchical features in the data. In our model, we found a 100 PEs in the bottleneck layer to be a consistently good choice for image classification.

In addition, the correntropy loss function is a great candidate for image classification. It filters noise well due to its nonlinearity and is an efficient measure of similarity between two datasets.

##### Acknowledgment

1. Sheela, K. G., & Deepa, S. N. (2013, June 20). Review on methods to fix number of hidden neurons in neural networks. Mathematical Problems in Engineering. https://www.hindawi.com/journals/mpe/2013/425740/