Learning to Unpermute

Muhammad Zakriya Shah Sarwar, Nehmiya Shikur

Abstract

This paper investigates the effectiveness of deep learning models in matching distributions, using the problem of unpermuting images as a case study. Unpermuting an image refers to the process of reversing the effects of image permutation, which involves randomly shuffling the pixels of an image. The research is divided into two phases. The first phase involves the development of a supervised learning model that trains a neural network on a paired data set of original and permuted images. The second phase focuses on the development of an unsupervised deep learning model that trains the network on unpaired images. The study demonstrates the ability of deep learning models to match distributions and address the problem of unpermuting images. The findings of this study can provide insights into the general problem of distribution matching in deep learning.

1 Introduction

Deep learning models have achieved remarkable success in learning to classify images and objects, enabling significant advancements in the field of computer vision. However, a less explored area is how deep learning models can learn to match the distributions of data sets. To investigate this topic, we chose the problem of unpermuting images as a toy example. Unpermuting images involves reversing the effects of image permutation, which is the process of shuffling the pixels of an image in a random order. In this paper, we demonstrate how deep learning techniques can be used to train neural networks in supervised and unsupervised settings to match the distributions of original and permuted images, thus addressing the problem of unpermuting images.

In the initial stage of our study, we concentrated on building supervised learning models. This involved training a neural network using a paired data set that consists of original and permuted images in a matched manner. The aim of training these models was to enable them to accurately generate the original image from their permuted counterpart by minimizing the difference between the unpermuted and original images. Through this approach, we sought to demonstrate the ability of deep learning models to learn the distributions in the images and to successfully unpermute them. The development of these supervised learning models laid the groundwork for our subsequent investigation into unsupervised deep learning models.

In the second phase of our study, we aimed to develop unsupervised deep learning models with an ambitious goal of training the network using an unpaired image data sets. The data sets consists of original and permuted images, where they are not matched to each other. This approach is challenging as it requires the model to learn to match the distributions of two sets of images without direct supervision.

In summary, section 2 of our paper provides a detailed overview of our image permutation techniques and the formation of our paired and unpaired image data sets. In sections 3 and 4, we discuss our supervised and unsupervised learning models, consecutively. Finally, section 5 presents the findings and interesting aspects of our study.

2 Data

For our study, we utilized both gray scale and color images, all of which underwent the same permutation process. Subsequently, we organized these permuted images into paired and unpaired formats for supervised and unsupervised models.

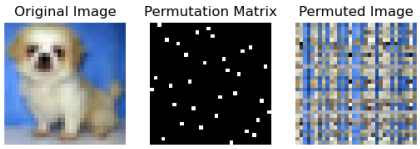
2.1 Permutation

Permuting an image involves rearranging the pixels of an image to create a new image that looks different from the original image. There are several methods to permute an image. For example, we could divide an image into smaller grid and then shuffle those grids. The other option is to scramble all pixels of the image, so that it is completely unrecognizable for human's eye. We took the later approach.

To permute the images on pixel levels we multiply the image with a permutation matrix from left and right. A permutation matrix is a square matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

(a) An example of 5×5 permutation matrix



(b) The permutation matrix in the middle shown as a binary image. Permuted Image is generated by multiplying original image with permutation matrix from left and right.

Figure 1

Let $\mathbf{P}$ be the permutation matrix and $\mathbf{X}$ an image. Then our permuted image $\mathbf{Y}$ can be described by equation (1).

$$\mathbf{Y} = \mathbf{PXP} \tag{1}$$

Where $\mathbf{X}, \mathbf{P}, \mathbf{Y} \in \mathbb{R}^{m \times m}$. Multiplying the image only from right side will permute the columns of the image and multiplying from left side will permute only rows. So we first multiply image from one side and then from other side with the same permutation matrix. Order of multiplication does not matter.

If the permutation matrix is known, then reverting to the original image is very simple.

$$\mathbf{X} = \mathbf{P^{-1}YP^{-1}} \tag{2}$$

where

$$\mathbf{P^{-1}} = \mathbf{P^{T}} \tag{3}$$

if the permutation matrix is not known, then to the best of our knowledge, it is not possible to solve equation (1) for $\mathbf{P}$. One way to figure out $\mathbf{P^{-1}}$ in equation (2), to get the original image back, can be trying all possible permutation matrices. But for a $m \times m$ permutation matrix, there are $m!$ possible different configurations (of which half matrices are transpose of the other half). For matrices of relatively large size trying to find the P^{-1} by going through all possible configurations will take huge time.

2.2 Paired and unpaired data sets

Paired data sets refer to a type of data sets in which permuted images are paired with their original images. In all training iterations, loss is calculated between the output image of the unpermuting model and the original image which was used to permute the input image of the model.

On the other hand, unpaired data sets do not have paired permuted and original images. Thus during training we can not match the output of the unpermuting model directly to it's original counterpart. Instead, a more sophisticated approach had to be developed which is presented in section 4.

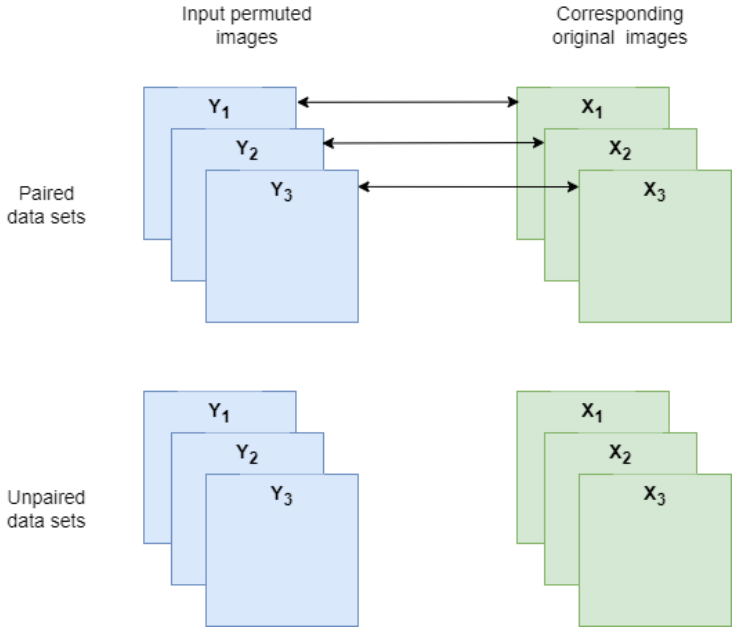


Figure 2: Illustration of paired and unpaired data sets.

Given that the images in the paired data sets are accurately matched, it enable us to use a supervised learning approach to train our models. In other words, we are able to provide the model with both the input (permuted image) and the expected output (original image). By doing so, we can optimize the model's parameters to minimize the error between the output and the expected output. On the other hand, for the unpaired data sets, we have to use an unsupervised learning approach since we don't have the matching expected output.

3 Supervised approach for paired data sets

In this stage of our research we utilized a paired data set to map permuted images to their corresponding original versions. This approach laid the foundation for our subsequent work in the unsupervised setting, and ensured the effectiveness of our models.

We have used supervised deep learning models, such as Single Layer Perceptron (SLP) [6] and Auto-encoders [3, p. 499], to address this problem by learning a mapping from permuted images to the original images. We trained the model on a large data set of permuted images, along with their corresponding original images. These models have been successful in recognizing patterns in the permuted images and were able to map them back to the original ones.

3.1 Classification

To understand and demonstrate the complexity of different type of Image data sets, we initially sought to develop several basic supervised learning models capable of classifying permuted images. In this pursuit, a cross-entropy loss function [2], as in equation (4), was utilized to evaluate the ability of a multi layer perceptron (MLP) [3, p. 164] model to correctly label permuted images, which correspond to their semantic content, for example 0-0-9 digits in MNIST data set [1]. This function employs the softmax [3, p. 180] function to predict the probability of various target categories and compares the distribution of the true labels c to that of the predictions $\hat{c}$.

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} log(\hat{c}_{ij}), \tag{4}$$

where N is the total number of examples and M is the total number of classes.

In this stage we trained four different models, namely Multi-Layer Perceptron (MLP), convolutional Neural Network (CNN) [3, p. 326], MLP with dropouts [7], and CNN with batch normalization[4]. These models were trained on three separate image data sets, namely MNIST [1], FashionMNIST [8] and CIFAR-10 [5], with the objective of classifying permuted images. MNIST and FashionMNIST has 70000 images each of size 28×28, of which 60000 images are used to train the models and 10000 images are used for testing. CIFAR-10 consists of 60,000 color images, each sized at $32 \times 32 \times 3$, distributed across 10 classes, with 6,000 images per class. To evaluate their performance, the training loss of each model on each data set was measured and plotted in a graph, which is shown below. By examining this graph, we can compare the training performance of the four models and determine which one performs the best for the task of classifying permuted images.

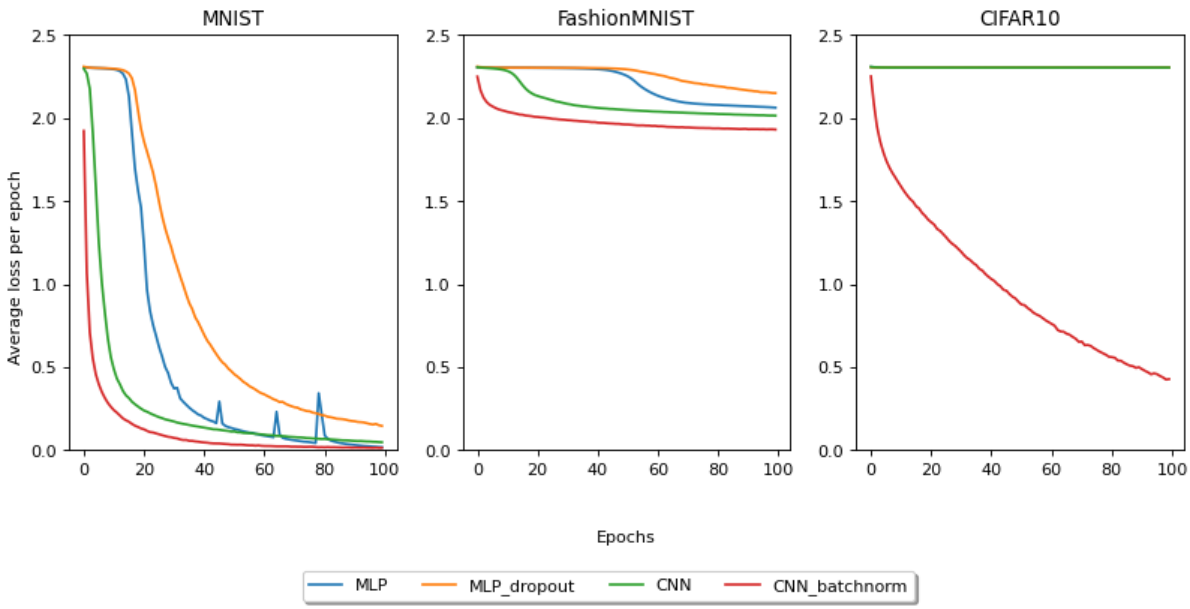


Figure 3: Classification Losses

We also sought to evaluate the accuracy of our classification models on the MNIST [1], FashionMNIST [8], and CIFAR-10 [5] data sets, in addition to analyzing their training performance. The table presented below indicates that all four models performed well in terms of accuracy, but the accuracy of the models on CIFAR-10 images was lower than that of MNIST and FashionMNIST. From these results, we can conclude that CIFAR-10 is a rather complex data set, as it has three color channels, while MNIST and FashionMNIST only have one. And also because CIFAR10 consists of realistic images and MNIST and FashionMNIST are synthetic sketches.

Model's Accuracy			
Model	**MNIST**	**FashionMNIST**	**CIFAR10**
MLP	0.9656	0.6441	0.0999
MLP + Dropout	0.9461	0.3668	0.1013
CNN	$\boxed{0.9728}$	0.7735	0.0999
CNN + Batch Norm	0.9581	$\boxed{0.7748}$	$\boxed{0.4885}$

Table 1: Our models accuracy on test data of different data sets. Best results are highlighted with boxes.

3.2 Unpermuting

Given that we have developed models capable of classifying various permuted images, the next step is to construct models that can reconstruct these permuted images to their originals. Our initial attempt at this involved using a Single-Layer Perceptron (SLP) model with the same dimensions for its input and output. We trained the SLP model using stochastic gradient descent [3, p. 149] and mean squared error [3, p. 131] as the loss metric to reconstruct the original image from its permuted form.

$$L = \sum_{i=1}^{N} \|\hat{\mathbf{X}}_\mathbf{i} - \mathbf{X}_\mathbf{i}\|_2^2, \tag{5}$$

where $\hat{\mathbf{X}}$ is model's output.

The SLP models underwent a training to reconstruct both gray-scale and RGB images, starting with MNIST and eventually progressing to CIFAR-10. Impressively, it was able to accurately reconstruct the permuted images that were given to it, as evidenced by Figure 4 and Figure 5.

Figure 4: Unpermuting single channel MNIST images with an SLP model. The images in the first column represent the original images while the images in the second and third column represent their corresponding permuted and unpermuted versions consecutively.

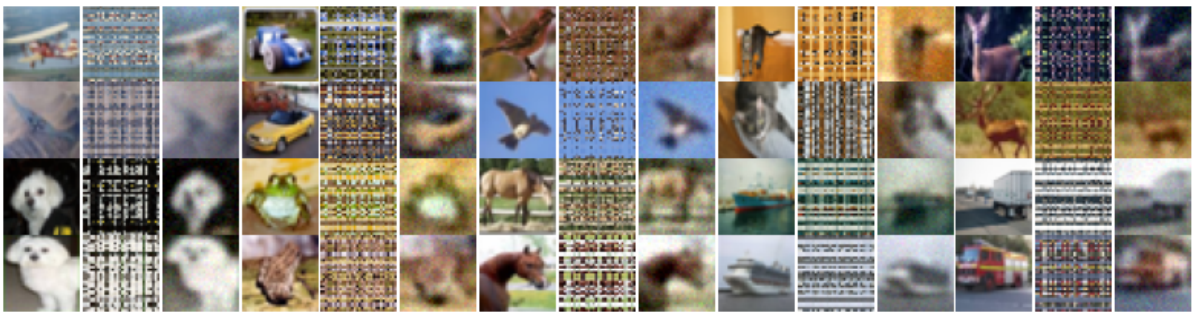


Figure 5: unpermuting multiple channel CIFAR10 images with an SLP model

In addition to the SLP model, an auto-encoder model that encodes the pixel values of the image by reducing it into a smaller latent representation and decodes this representation into a dimension that is equal to the input, as shown in Figure 8, were trained and tested on MNIST, FashionMNIST and CIFAR-10 data-sets. The model had the same input and output dimensions with a parameterized latent representation. Additionally, the mean squared error loss function metric as seen in equation (5) was used for the training of the model.

The auto-encoder model we built showed promising results in unpermuting images across different data sets, as depicted in Figure 5 and 6 for MNIST and FashionMNIST data sets, respectively. These results

demonstrate the model's ability to reconstruct permuted images, which further validates the effectiveness of our approach in handling permuted image classification and reconstruction tasks. Remarkably, the model achieved this despite being trained for only a few epochs. This is a significant finding that highlights the efficiency of our model and its potential for further development and improvement. Overall, our auto-encoder model has demonstrated its ability to perform well on permuted image reconstruction tasks.

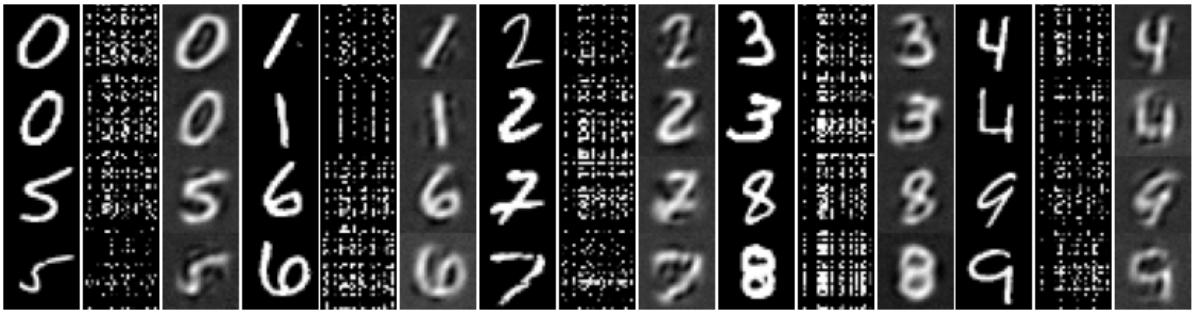


Figure 6: Unpermuting images from MNIST data set using an auto-encoder model trained for only 20 epochs

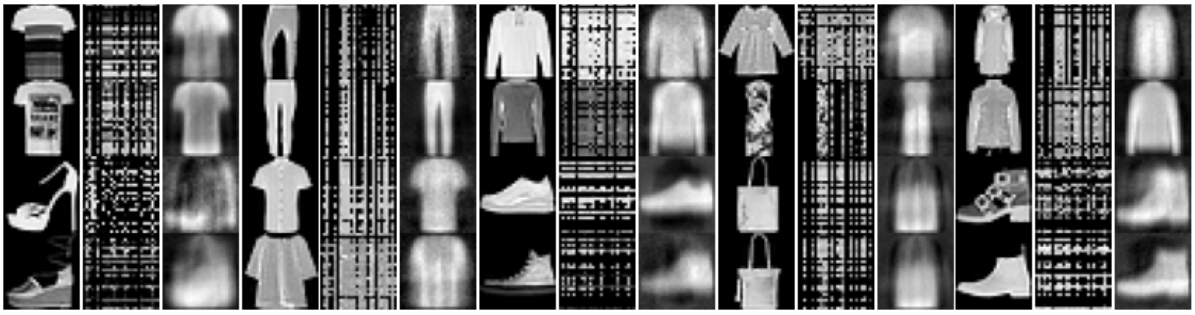


Figure 7: Unpermuting images from FashionMNIST data set using an auto-encoder model trained for only 20 epochs

Our models exhibited varied levels of success in unpermuting different types of images. While MNIST and FashionMNIST data sets were relatively easier for the models to unpermute, CIFAR-10 presented a greater difficulty. Surprisingly, the models trained on MNIST and FashionMNIST images were able to successfully unpermute each other; however, they struggled with images from CIFAR-10. We hypothesize that the sparsity of the image matrix in CIFAR-10 images likely contributed to the models' difficulty in learning and unpermuting them. For more details and visualizations of our experiments, please refer to the appendix section of our paper.

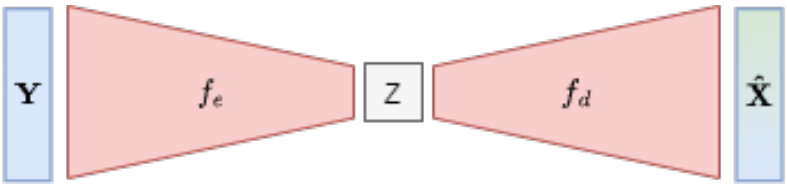


Figure 8: An Auto-encoder model, that uses a lower-dimensional latent representation z to encode the input data, which is then decoded back to reconstruct the original input.

3.3 Classifier and Generator

By dividing the previously mentioned auto-encoder model into two distinct models, a classifier and a generator, we were able to generate images from integer values, corresponding to the classes of the images. The two models were trained separately, which gave us more flexibility and control over the generated images. The classifier model is used to determine the class of the image, while the generator model is responsible for generating the images based on the input values. Overall, this approach proved to be effective in generating images of given classes from integer-valueed class labels.

The Classifier model is trained on the permuted images to classify them, while the Generator model takes the output values of the Classifier model, which are integer values, as input and is trained to construct an image matrix that is later plotted as an image. The Generator model uses mean squared error as a loss function, which takes a randomly selected image that corresponds to the label of the original image. This approach allowed us to generate new images by feeding integer values into the Generator model, enabling us to explore the space of possible images. The use of the Classifier model as an intermediate step ensures that the generated images belong to the same class as the original image, while the use of mean squared error as the loss function ensures that the generated images resemble the original image. The results of this approach can be seen in Figure 9.

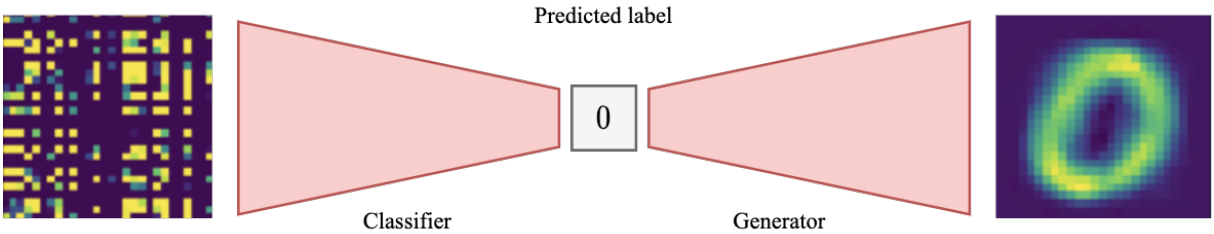


Figure 9: Generating images from integer values using our classifier and generator model

Now that we have successfully developed models that can classify, unpermute and generate images by learning their distributions, it is worth noting that all these tasks were possible due to the availability of original images that could be used for learning to reconstruct and unpermute the permuted images in a supervised learning setting. In the next chapter of the paper, we will explore different techniques for unpermuting images without any prior information about the original images or labels. This presents a more challenging and realistic scenario, where we will need to rely on unsupervised learning techniques to infer the structure of the data and its underlying patterns. This would help to address the real-world problem of dealing with permuted images where we do not have access to the original images or their corresponding labels.

4 Unsupervised

We are now focused on addressing the challenge of unpermuting images in the absence of their labels. This is the next stage of our work, where we aim to develop a solution for unpermuting images without having access to their original or unpermuted versions. This presents a new challenge, as the models we developed earlier were trained on the original images to unpermute them. Without prior knowledge of the original images, we need to develop new techniques to unpermute the images in a completely unsupervised manner. This will require novel approaches and creative thinking to overcome this new challenge. We will discuss our proposed methods and the results in the next section of the paper.

A crucial challenge in addressing the issue of unpermuting an image in an unsupervised learning setting is determining the appropriate loss metric for training our model. To address this, we employed a Cycle Consistency loss [9]. This loss metric is designed to ensure that our generator model can generate an image and then reconstruct it back to its original form as closely as possible. In other words, we aim to ensure that the generated image is both realistic and accurate, and that the process of generating and reconstructing it does not introduce any unexpected changes. Our use of Cycle Consistency loss enables

our model to learn the underlying structure of the image, and to use this knowledge to unpermute images in the absence of the original images.

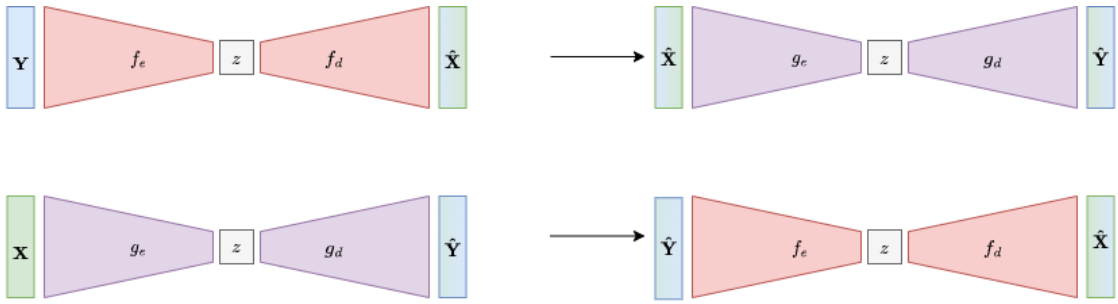


Figure 10: During the training cycle, an unpaired permuted image $\mathbf{Y}$ is processed through the model f and used as input for the model g. Likewise, an unpaired original image, which is denoted as $\mathbf{X}$, is fed into the model g and its output is passed through the model f. Moreover, f_e and g_e represent the encoder, while f_d and g_d represent the decoder sections of the auto-encoder models.

This experiment involved training two auto-encoders simultaneously using the Cycle Consistency loss function outlined in (6) and (7). The two models, denoted as g and f, were trained to process an original image and an unpaired permuted image, respectively. The model g was given the original image as input, and its output was passed to f. The resulting output of f was then compared to the original image using the mean squared error (MSE) loss, denoted as L_o in Equation (6). Conversely, the model f was given the permuted image as input, and its output was passed to g. The output of g was compared to the permuted image using the MSE loss, denoted as L_p in Equation (7).

$$L_o = E_{x \sim X}(\|f(g(x)) - x\|_2^2) \tag{6}$$

$$L_p = E_{y \sim Y}(\|g(f(y)) - y\|_2^2) \tag{7}$$

The total loss function, denoted as L in Equation (8), was defined as the sum of the two MSE losses. This total loss function was then used for back-propagation to update both models.

$$L = L_o + L_p \tag{8}$$

Once both models were trained, permuted images were passed through f to obtain the unpermuted images.

In our experiments, we trained models to unpermute images from both MNIST and FashionMNIST data sets. As shown in Figure 11, the models were able to successfully match the distributions of the permuted and the original images which resulted in unpermuting of the images to their corresponding correct classes. However, for FashionMNIST images, the results as shown in Figure 12, suggest that the models learned to map the distribution of permuted images to original images but it struggled to match them to their actual classes. The reason why certain unpermuted images in a given class may not exactly match their original counterparts is because, when it comes to the FashionMNIST data set, the models have not fully learned the correct mapping. In contrast, for the MNIST data set, the models have learned the correct mapping to a much greater extent. Therefore, in the case of FashionMNIST, many images are either being misclassified into the wrong class or are being transformed into slightly different images within the same class, rather than being an exact replica of the original image.

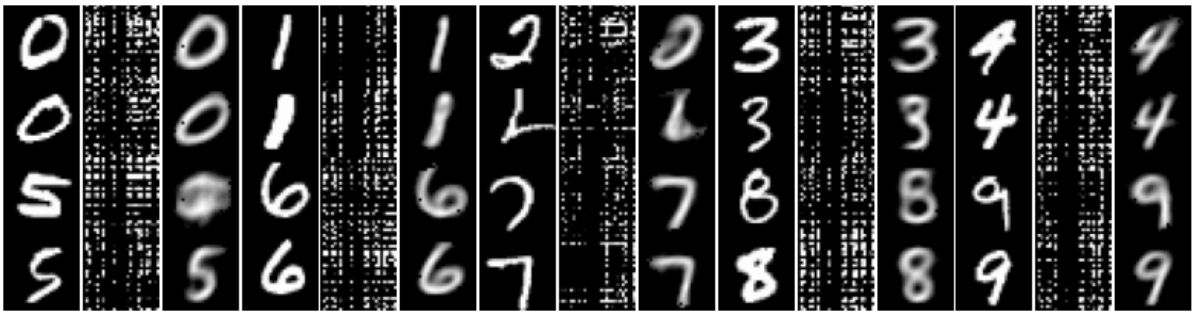


Figure 11: unpaired MNIST data unpermuted using 2 auto-encoders

To evaluate the accuracy of our models, we employed a CNN model with batch normalization layers that we developed during the supervised stage of our research. Using this model, we assessed the number of permuted images that were correctly unpermuted to their respective class. For the MNIST unpermuted images, the resulting accuracy was 69.89%. This metric provided us with valuable insights into the performance of our models and allowed us to further optimize our approach to improve accuracy.

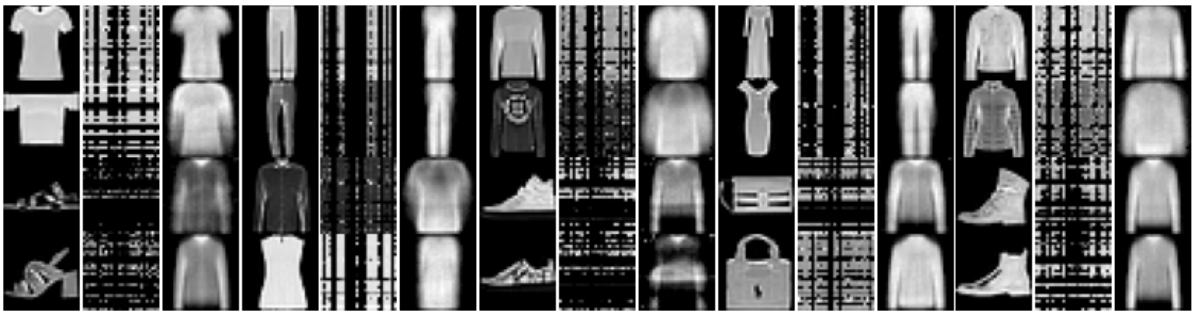


Figure 12: unpaired FashionMNIST data unpermuted using 2 auto-encoders

5 Discussion

In our study, we were able to match the distributions of permuted images with their original counterparts using our models, which effectively addressed the issue of unpermuting images in a supervised setting. Through experimenting with different data sets we trained single-layer models and auto-encoders and found that increasing the number of training epochs resulted in better outcomes. However, we observed that our models trained on the MNIST data set had limited success in unpermuting images beyond MNIST. This was due to the sparse nature of MNIST images, where only a few pixels have different values, making it difficult for the network to unpermute the entire image. In contrast, our models trained on the CIFAR10 data set were able to unpermute any image.

We also explored the unsupervised setting and were able to unpermute MNIST images. However, we noticed that our models were not able to learn without the addition of a dropout layer. When we attempted to apply the same model to CIFAR10 images, the results were not satisfactory. We propose that the use of models with more layers or convolutional neural networks could lead to better performance in unpermuting CIFAR-10 images. Further details on our experiments, including visualizations, can be found in the appendix section of this paper.

Acknowledgement: This work was supervised by Professor *Magda Gregorova*, from *Faculty of Computer Science and Business Information Systems,Technische Hochschule Würzburg-Schweinfurt. This paper have benefited by the English and grammatical assistance offered by ChatGPT,OpenAI*

References

[1] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[2] I. J. Good. "Rational Decisions". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 14.1 (1952), pp. 107–114.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* `http://www.deeplearningbook.org`. MIT Press, 2016.

[4] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015.

[5] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images.* Tech. rep. 0. Toronto, Ontario: University of Toronto, 2009.

[6] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para.* Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.

[7] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.

[8] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.* Aug. 28, 2017.

[9] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2223–2232.

Muhammad Zakriya Shah Sarwar
Technische Hochschule Würzburg-Schweinfurt
FIW
Sanderheinrichsleitenweg 20 97074 Würzburg
Germany
E-mail: *muhammadzakriya.shahsarwar@study.thws.de*

Nehmiya Shikur
Technische Hochschule Würzburg-Schweinfurt
FIW
Sanderheinrichsleitenweg 20 97074 Würzburg
Germany
E-mail: *nehmiya.shikur@study.thws.de*

A Appendix

In this section, we evaluated the generalization performance of our SLP models trained to unpermute images from various data sets. We tested the models on data sets they were not trained on to assess their ability to perform on unseen data. The objective was to verify if our models could handle new data sets beyond their original training data.

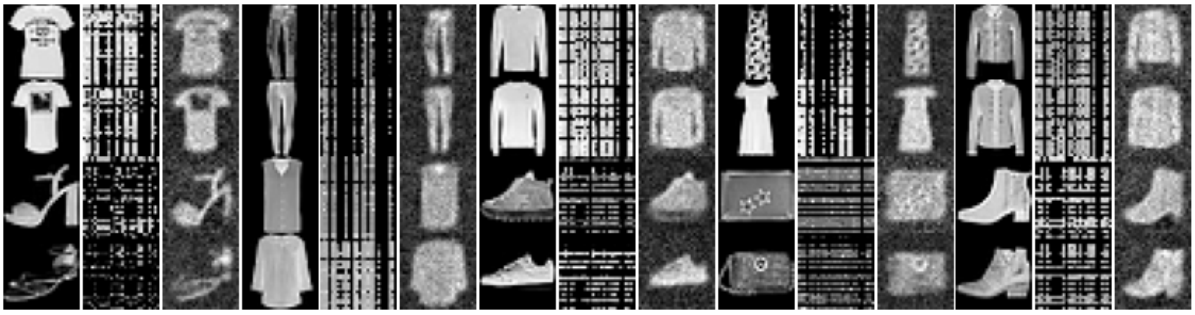


Figure 13: The result of applying an SLP model trained on the MNIST data set to images from the FashionMNIST data set. The output displays the model's performance in unpermuting the FashionMNIST images.

Figure 14: The SLP model trained on MNIST dataset was tested on images from CIFAR-10 dataset, but failed to unpermute them, indicating that the model cannot generalize well to other types of images.

Figure 15: The generated output from the SLP model, trained on the FashionMNIST data set, for the images from the MNIST data set demonstrates the generalization ability of the model. This indicates that the model is capable of unpermuting images from data sets that it has not been trained on, which further validates its effectiveness.

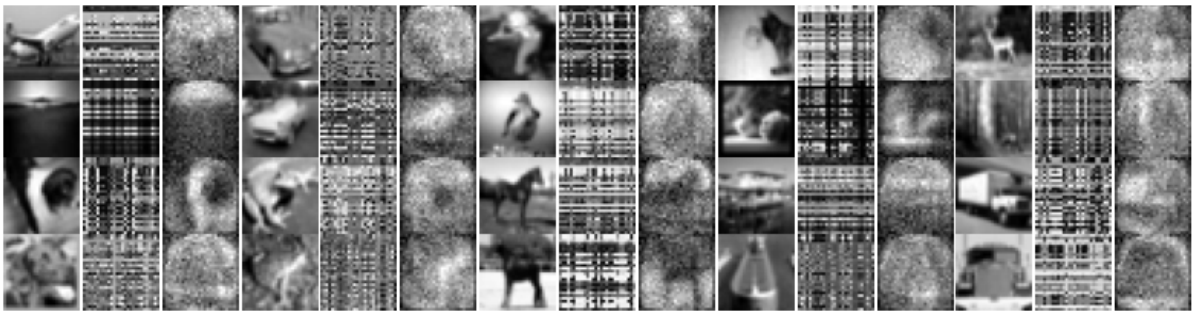


Figure 16: The SLP model trained on FashionMNIST data set was also tested on CIFAR-10 data set and it showed better results compared to the SLP model trained on the MNIST data set when unpermuting the images.

Figure 17: The SLP model trained on CIFAR-10 data set demonstrated a high level of proficiency in unpermuting images from the MNIST data set, achieving good results.

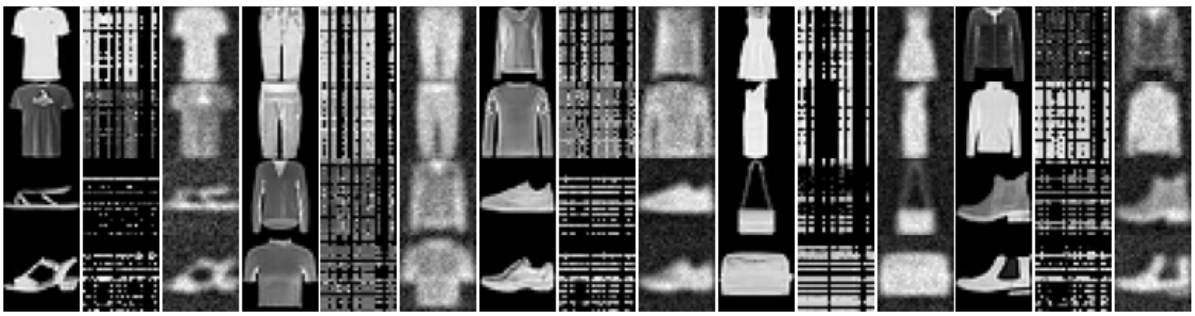


Figure 18: The output of the SLP model trained on the CIFAR-10 data set for images from FashionMNIST data set was very encouraging, as it was able to successfully unpermute the images with high accuracy. This indicates that the model has a good generalization ability and can be applied to various data sets.