CS554 Introduction to Machine Learning Convolutional Auto Encoder

Mert Erkol

May 24, 2023

1 Introduction

In this study Single Layer Perceptron(SLP) and Multi Layer Perceptron(MLP) with different number of hidden units fitted to the given custom 1D dataset and their results evaluated with Mean Square of Error.

In this study Convolutional Auto Encoder(CAE) implemented with PyTorch and Python to generate the second half of the image from given first half image in the FashionMNIST dataset.

2 Background

The goal is to fit a model to FashionMNIST image data that generalizes the problem and generates the second image half when first half image input is given. In this section Convolutional layers, encoder and decoder architecture explained in detail.

2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of artificial neural network designed for processing grid-like data, including images and speech signals. A CNN is composed of one or more convolutional layers, often followed by pooling layers, and then fully connected layers as in a standard multi-layer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image or other 2D inputs [1].

The shape calculation of next layer in CNN is:

$$O = \frac{(W - K + 2P)}{S} + 1 \tag{1}$$

where O is the size of output feature maps, W is the input or feature map size, K is the size of the kernel, P amount of padding applied and S is the amount of stride

The key components of a CNN are:

- **Convolutional Layer:** This layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter.
- **Pooling Layer:** This layer progressively reduces the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layers operate independently on every depth slice of the input and resize it spatially.
- **Layer parameters:** Convolutinal layers has optional parameters like stride, step iteration of the filter. Padding, extra artificial pixels that created to control the image size. Kernel size, shape specifications of the convolution filter most common's are 3x3 and 5x5.

In this way, CNNs are able to effectively learn hierarchical representations of the input data, making them very effective for a wide range of tasks in computer vision, speech recognition, and other areas. Example CNN architecture can seen below in 1

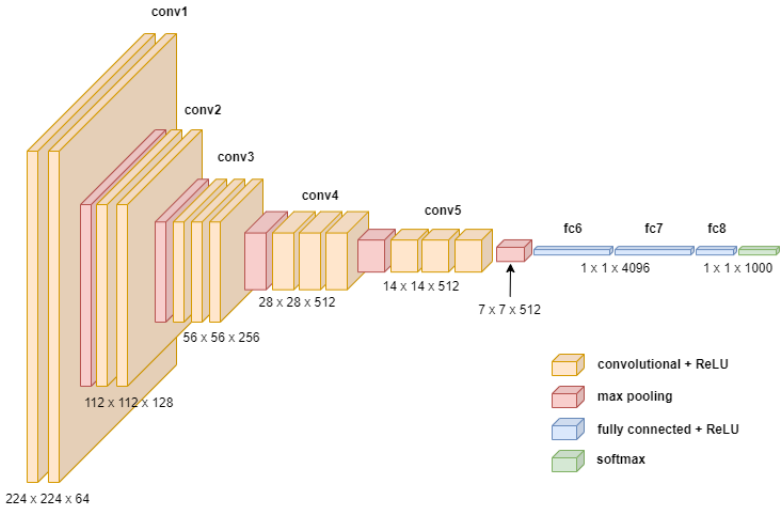

Figure 1: Example CNN architecture with 5 conv layers and 3 fc layers from CNN

2.2 Encoder and Decoder

The encoder-decoder style architecture is very commonly used in Machine Learning at NLP or Computed Vision related tasks. The idea is there are two main component architectures symmetrical to each other. Encoder compresses the data to a latent space that can be represented in a flatten small vector. Decoder decompress that vector and make it an image again. Single loss function is optimized during this process. Either convolutional or fully connected layers can be used in both architectures or both of them. Example architecture of an auto-encoder can seen in 2

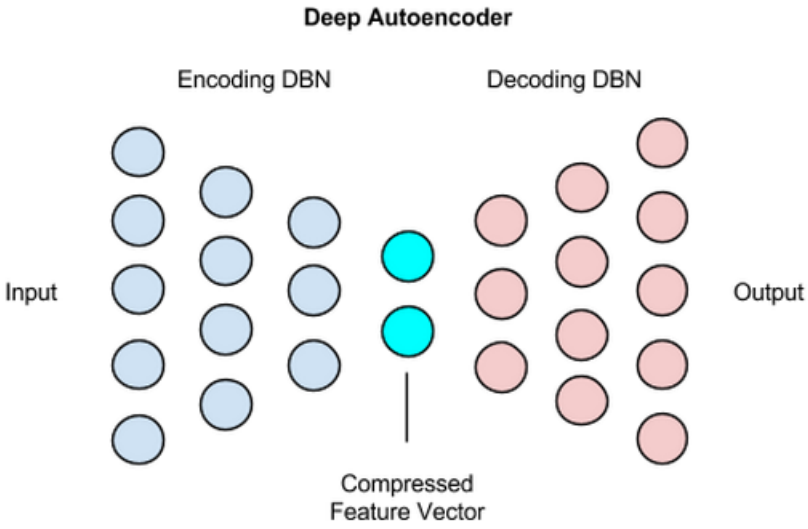

Figure 2: Deep Autoencoder arcitecture from Deep Autoencoder

2.3 Activation function

An activation function is a mathematical operation that is applied to an output of a neuron. It provides non-linearity to the neural network. There are several activation functions is available but for this work LeakyReLU activation function has been used as follows

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \tag{2}$$

where x is the either output of the neuron or the input, α is a very small constant if x is less than 0, usually has been set to 0.01.

2.4 Loss function

The Mean Square Error(MSE) is a common loss function for evaluating reconstruction problems. It calculates the distance between the prediction and the desired value squares and calculates the mean for all data points pixels in this case.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{3}$$

where n is the number of samples in the dataset. $\hat{y}_i$ is the i_th prediction and y_i is the i_th desired output.

2.5 FashionMNIST

FashionMNIST is a popular dataset can be used in classification, image generation, image denoising tasks. It contains 50,000 training and 10,000 test images with size 28x28x1 and 10 different classes; T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle Boot[2].

3 Methodology

Utilizing Python 3.9 as the programming language of choice, this project is a comprehensive exploration of Convolutional Autoencoders (CAEs) applied to the FashionMNIST dataset. The PyTorch library served as the principal tool for designing and training the neural networks, while Matplotlib was employed for the visualization of generated results.

The procedure begins with loading the FashionMNIST train and test data. The CAE is then trained using vertically split images, where the left side of each image is provided as input, complemented with a learning rate scheduler. Subsequently, the model's performance is evaluated on both the training and validation set. This evaluation is carried out using the Mean Squared Error (MSE) loss function, which takes the right-hand side of each image as input for each batch, eventually producing an aggregate value per epoch.

The project also includes a comprehensive analysis of multiple CAE variants. These variants differ in terms of layer depth, filter count, learning rate, optimizer, and loss function, each carefully tested and tuned for optimal performance. In the Results section, the best performing CAE has been presented from a multitude of potential configurations, showcasing the effectiveness of the tuning process.

Here are the specific details of the CAE:

- **Architecture Details:** The CAE has a total of 3 Convolutional and 2 FC layers. Kernel size is (4, 2) for all convolutional layers. Out features of convolutional layers are 32, 64, 128 by order. Strided convolutional technique with same padding has been used without needing a pooling operation. Batch normalization is applied at first two layers. LeakyRELU activation function has been used for both convolutional and FC layers. Hidden dimension size is 256 and the latent space size is 64.

- **Loss Function:** The loss function used in the CAE is the Mean Squarred Error (MSE) function.

- **Learning Rate:** The learning rate used for training the CAE is 0.001 but learning rate has been decreased by ReduceLRonPlateu scheduler if the validation loss has not been decreased in last 2 epochs. Decrease factor was 0.5.
- **Training details:** Network has trained for 20 epochs with a batch size of 512 on a NVIDIA RTX 2060 6 GB VRAM GPU.

4 Results

In this section, the result of model's image generation has been presented. For each of the ten classes, five samples have been selected. These samples display the original left-hand side of the image alongside the model-generated right-hand side, providing a clear comparison between the original and the reconstructed portions of the image.

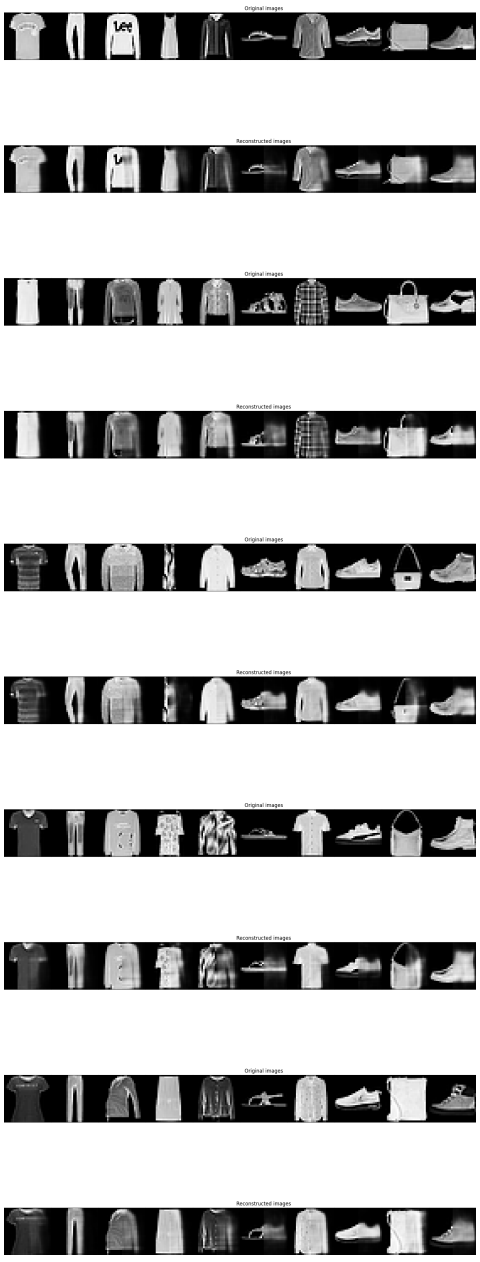

Figure 3: Original and Reconstructed images from the testset FashionMNIST

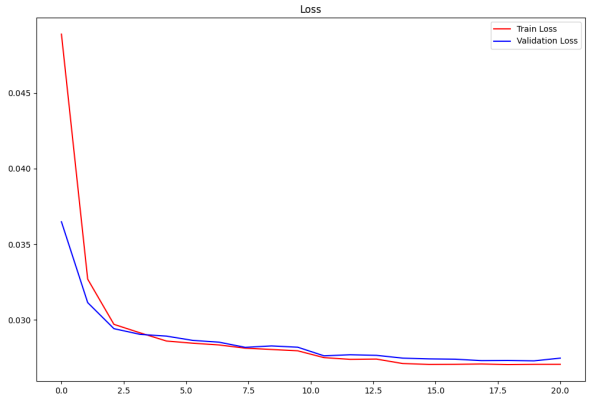

Figure 4: Train and Validation loss by epoch

5 Discussion

As depicted in Figure 4, both the training and validation loss converges around a value of 0.0273 by the 17th epoch. Beyond this point, further training of the model becomes redundant and could potentially lead to overfitting on the training set. To facilitate reaching the global minimum, the learning rate was gradually decreased four times from an initial value of 0.001 during the training process.

Upon examining the visual results in Figure 3, it's apparent that our model exhibits excellent performance on certain classes, such as Trousers, T-shirt, and Dress. However, the quality of reconstruction is less satisfactory for classes like Sneaker and Ankle Boot. This discrepancy can be attributed to the fact that the first group of classes consists of symmetrical objects, which are inherently easier to reconstruct.

Unfortunately, the same cannot be said for the latter group of classes, which are not symmetrical. Moreover, it's noteworthy that the reconstruction quality tends to fall if a non-symmetrical image is oriented from left to right. This is likely due to a bias in the dataset, where non-symmetrical objects are predominantly positioned from right to left.

6 Conclusion

Our exploration of Convolutional Autoencoders (CAEs) has shown their effectiveness in image reconstruction, particularly with symmetric patterns in the FashionMNIST dataset. Challenges remain in reconstructing non-symmetrical classes and managing dataset biases.

Future work may involve advanced techniques, such as variational autoencoders or generative adversarial networks, to improve reconstruction quality. Additionally, addressing dataset biases and increase in dataset size could yield better results.

In summary, CAEs have demonstrated potential in image reconstruction, with potential applications extending to other machine learning tasks. Their exploration serves as a vital step towards advancements in the field of deep learning.

References

[1] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2014.

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*. cite arxiv:1708.07747Comment: Dataset is freely available at https://github.com/zalandoresearch/fashion-mnist Benchmark is available at http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/. 2017. URL: http://arxiv.org/abs/1708.07747.