Image Classification Using Convolutional Neural Network and the effect of Autoencoder on the prediction of images in the Fashion-MNIST Dataset

Astthor Arnar Bragason, Alexandra Meszaros,
Kristina Tomicic, Marwane Ghalila

December 19, 2022

Abstract

Autoencoders have drawn lots of attention in the field of image processing [8]. As the target output of an Autoencoder is the same as its input, Autoencoders can for example be used for data compression and data denoising [6]. In this paper, we focus on trying to find out how the prediction rate of a Convolutional Neural Network is affected after running the data through an Autoencoder. After evaluating the images' before- and after state, we confirm that the Autoencoder influences the predictions negatively.

1 Introduction

Machine learning models such as Convolutional Neural networks (CNNs) [2] are based on deep learning architecture and are specifically used for image recognition tasks [12].

In our research, we use CNN to classify images of clothes using the Fashion MNIST dataset [10]. Images can consist of large amounts of data, can therefore be slow to transfer and it can be expensive to store large amounts of image data. In order to facilitate the transfer process and reduce storage costs, we are interested to see how data that has been reconstructed by an autoencoder might affect the CNN's ability to recognize images. In machine learning, an autoencoder is an unsupervised learning algorithm that encodes an image, reducing its dimensions, and then tries to decode it to the image's original state [6].

1.1 Research question

The focus of this paper is the following research question: How accurately can we predict different clothing examples using a Convolutional Neural Network before and after the data has been processed through an autoencoder? Furthermore, since images may become noisy during transfer, we decided to utilise the autoencoder's other feature, namely denoising, which leads to our research's subquestion: Can we improve the prediction rate of noisy images after using an autoencoder for denoising?

Hypothesis: Processing Fashion MNIST dataset through Autoencoder negatively affects the prediction rate.
Null-hypothesis: Processing the Fashion MNIST dataset through autoencoder does not impact the prediction rate negatively.
Hypothesis 2: Processing noisy images first through autoencoder and then through a CNN, outperforms noisy image classification using just a CNN.
Null-hypothesis 2: Processing noisy images first through autoencoder and then through a CNN, does not outperform noisy image classification using just a CNN.

2 Methods

We use Nunamaker's Multi-methodological framework, which is a systematic way to approach complex research problems [7]. It emphasises the importance of using multiple methods and perspectives to gather and analyse data, and using that information for decision making. Our research process follows Nunamaker's approach, as during the process we go back and forth between the steps of the research; for example after evaluating our first results, we go back to rephrase our research question and hypothesis.

For statistical analysis, we used a method to check the variance between the count of each class of clothing in the training sets. Then for data preparation we split the training dataset into training and validation datasets, using stratification. Stratified sampling is a method often used for partitioning population data into subgroups, as it reduces sampling errors[4].

To classify images we use a Convolutional Neural Network (CNN) which is particularly well suited for image-related tasks, because they are able to automatically learn spatial hierarchies of features from the data [11]. We also explore the topic about autoencoders' two widely used use cases, namely compressing data [1], which helps to reduce the storage usage, and removing noise from the original data. Autoencoders consist of an encoder and a decoder. Encoding an image reduces its dimension, and a decoder restores the number of dimensions based on the encoded data. Another application of autoencoder is image denoising. In this task, we treat the autoencoder as a nonlinear function that can eliminate the effect of noises in the image.

3 Analysis

3.1 Data

We use the Fashion MNIST dataset, which includes a total 70,000 images, and consists of a training set with 60,000 images and a test set with 10,000 images. In order to estimate the performance of a model during training, we divide the training set, using stratified sampling, into training and validation datasets, with 50,000 images and 10,000 images respectively. Each image is a 28x28 grayscale image, which is associated with a label from 10 classes [10].

Descriptive statistics Each image in both the training and test set is assigned to one of the following ten labels: ['T-shirt/Top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle Boot']

Counting how many examples there are of each label class in the training, validation and test datasets, we check for variance between the count of each label class in the datasets. This is to make sure that splitting the training dataset using stratification works as expected, meaning that in each dataset we have the same amount of images in each label class. For example, after splitting the training dataset, we have 5,000 images of each class in the training dataset.

3.2 Structure of models

CNN Our Convolutional Neural Network's architecture is inspired by a previously built model, made by the data scientist Eamon Fleming [5]. Using his model as inspiration we experiment by changing the structure to improve our results as explained in the 'Process' section, and came up with the final structure that can be explained in two parts:

- Feature extraction, which includes four 2D convolutional layers, a single BatchNormalization layer after the first convolutional layer, followed by a MaxPooling2D layer, and finally a Flatten layer.
- The network, which includes three dense layers.

In both parts of the CNN, there is a dropout layer separating each layer, refer to Table 1 to see dropout rate.
As we can see it in Table 1, all layers use the Rectified Linear Unit (ReLU) activation function, except the last 10 neuron output layers representing the classification output, which uses the softmax activation function. Compiling the model, we use the optimizer Adam, and the Categorical Crossentropy loss function. Fitting the model, we use the training and validation

Layer	Type	Nodes	Filter	Activation
1	Conv2D	32	3x3	relu
1	BatchNormalization	-	-	-
1	Dropout	0.2	-	-
2	Conv2D	32	3x3	relu
2	Dropout	0.2	-	-
3	Conv2D	24	3x3	relu
3	Dropout	0.3	-	-
4	Conv2D	64	3x3	relu
4	MaxPool2D	-	2x2	-
4	Dropout	0.2	-	-
4	Flatten	-	-	-
4	Dense	128	-	relu
4	Dropout	0.3	-	-
5	Dense	64	-	relu
6	Dropout	0.05	-	-
6	Dense	10	-	softmax

Table 1: Structure of Convolutional Neural Network

datasets, 30 epochs, 128 batch size and have a list of callbacks. Callbacks consist of a checkpoint, saving the best end-of-epoch model based on the validation accuracy, and a scheduler function that updates the learning rate.

Autoencoder We specify the model's hyper-parameters of each convolution layer that we implement (kernel size, stride, padding), refer to Table 2. It consists of two parts. The first part (encoding) is made of convolution and pooling layers and the second part (decoding) is a mirror version of the encoding, except that instead of pooling layers it has upsampling layers [3].

	Layer	Type	Nodes	Filter	Activation
Encoder	1	Conv2D	64	3x3	relu
	1	Conv2D	32	3x3	relu
	1	MaxPool2D	-	2x2	-
	2	Conv2D	32	3x3	relu
	2	Conv2D	32	3x3	relu
	2	MaxPool2D	-	2x2	-
	3	Conv2D	16	3x3	relu
	3	Conv2D	4	2x2	relu
	3	MaxPool2D	-	2x2	-
Decoder	1	Conv2D	4	2x2	relu
	1	Conv2D	16	3x3	relu
	1	UpSampling2D	-	2x2	-
	2	Conv2D	32	3x3	relu
	2	Conv2D	32	3x3	relu
	1	UpSampling2D	-	2x2	-
	3	Conv2D	32	3x3	relu
	3	Conv2D	64	3x3	relu
	1	UpSampling2D	-	2x2	-

Table 2: Structure of Autoencoder

3.3 Process

Throughout the research process, following Nunamaker's framework [7], we try several different structures for both our CNN and autoencoder models - including experimenting with the number of layers, their number of neurons, the models' activation functions, increased dropout rates in the dropout layers, and more. The final structure provided the best results from our experimentation, while comparing how well the CNN was able to classify images, with clean, noisy, processed or unprocessed examples, as described further here in this section.

Defining the terms here, a 'noisy' example is a term we use for images where we have applied so-called 'salt and pepper' noise to the images. When we refer to examples being 'clean', we mean images where we have not added any noise to the images. The terms processed and unprocessed, refer to whether the examples have been processed through the autoencoder or not.

Following our research question we start by feeding clean images from the test set to the CNN and evaluate the results, and in the next step we first process the images through the autoencoder, then the CNN and we compare the prediction rate of the two results, see 1.

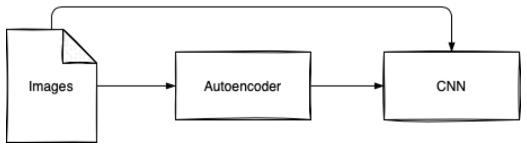

Figure 1: Feeding clear images to CNN and Autoencoder

Furthermore we also want to evaluate how the autoencoder affects the prediction rate with noisy images. In order to do that, additionally to the previous process, we apply salt and pepper noise to the images. In this case, before evaluating the data with our CNN, we use the autoencoder's denoising approach [9], by training it with clean and noisy images so the autoencoder is able to reconstruct the images (see Figure 2).

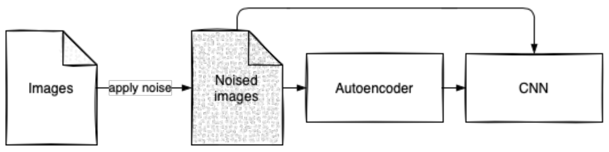

Figure 2: Feeding noisy images to CNN and Autoencoder

The visualisation of the images (Figure 3) before- and after state helps understanding the process.

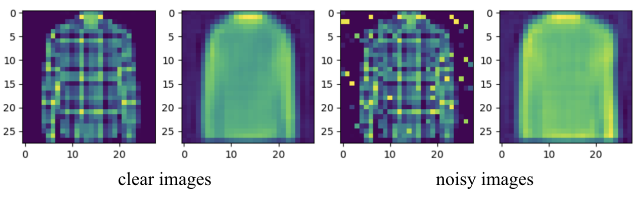

Figure 3: Feeding noisy images to CNN and Autoencoder

4 Findings

Our main objective in this section is to compare unprocessed samples against processed samples, both for clean and noisy examples. Both the clean and noisy samples that we label as 'processed' have been encoded and decoded by the same autoencoder as described in the analysis section.

Loss during the evaluation of the Autoencoder: 4.15%.

When evaluating the prediction accuracy the CNN achieved, we used the best-end-of-epoch model that was saved during training as explained in the analysis section. From the 10,000 examples in the test set, we feed four different datasets of images to the CNN, in two iterations, with two datasets in each iteration. In the first iteration (Figure 1) the clean and noisy examples without using the autoencoder, and in the second round (Figure 2) running the clean and noisy examples through the autoencoder before the CNN, which resulted in deterioration of the prediction rate compared to the first iteration (see Table 3).

	Clean examples	Noisy examples
1st iteration (without using autoencoder)	92.90 %	57.20 %
2nd iteration (using autoencoder)	62.92 %	56.21 %

Table 3: Results of the CNN's prediction rates

Comparing the percentage values seen in Table 3 we can conclude that clean unprocessed examples had a 29.98% higher prediction rate than clean processed examples. While noisy unprocessed samples had a 0.99% higher prediction rate than noisy processed examples.

5 Conclusion

From this experiment, we see deterioration in both clean and noisy images when processing the examples with the convolutional autoencoder. Based on that we accept our first hypothesis, as using the autoencoder negatively affects the CNN's prediction rate, and at the same time we reject the related null-hypothesis. Regarding the second hypothesis about the effect of the autoencoder on classifying noisy images, we reject the hypothesis, as the CNN's prediction rate did not improve after using autoencoder on noisy images. Consequently we accept the second null-hypothesis.

However, as we can see from the findings, evaluating the autoencoder reports some loss, so if we had an autoencoder with less loss, then we could expect the CNN to perform better using processed CNN to perform better using processed noisy images.

References

[1] P. Baldi. “Autoencoders, unsupervised learning, and deep architectures”. In: *In Proceedings of ICML Workshop on Unsupervised and Transfer Learning* (2012), pp. 37–49.

[2] Jason Brownlee. “Deep Learning CNN for Fashion-MNIST Clothing Classification”. In: *In Deep Learning for Computer Vision* (2019).

[3] Jason Brownlee. “How to use the UpSampling2D and Conv2DTranspose Layers in Keras”. In: *In Generative Adversarial Networks* (2019).

[4] Eugen. “Stratified Sampling in Machine Learning”. In: *Baeldung* (2022).

[5] Eamon Fleming. “Dress in layers, Convolutional Neural Net Tuning with the Fashion MNIST Dataset”. In: *Eamon Fleming Blog* (2021).

[6] Harris D Gedeon TD. “Progressive image compression”. In: *In Neural Networks, International Joint Conference on* 4.236 (1992), pp. 403–407.

[7] Jay F. Nunamaker Jr., Minder Chen, and Titus D.M. Purdin. “Systems Development in Information Systems Research”. In: *Journal of Management Information Systems* 7.3 (1990), pp. 89–106.

[8] Mohd. Aquib Ansari Komal Bajaja Dushyant Kumar Singhb. “Autoencoders Based Deep Learner for Image Denoising in Third International Conference on Computing and Network Communications”. In: (2019).

[9] Sijuade Oguntayo Melania Abrahamian. “How to Use Autoencoders for Image Denoising”. In: (2022).

[10] Zalando Research. “Fashion MNIST”. In: *Kaggle.com* (2017).

[11] Devansh Sharma. “Image Classification Using Convolutional Neural Networks: A step by step guide”. In: *In Data Science Blogathon* (2021).

[12] Christopher Thomas. “An introduction to Convolutional Neural Networks”. In: *Towards Data Science* (2019).