UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Homework 2: Unsupervised Deep Learning

Neural Networks and Deep Learning

Chelsea Owen
Student ID: 2006036

September 11, 2022

In this homework the task was to implement and test neural networks for solving unsupervised learning problems. The dataset for this homework was the Fashion-MNIST, also used for the classification task of homework 1. The model is a Convolutional AutoEncoder. As in homework 1, in order to find the best model in a random search and cross validation was used. The second part consisted of creating a Variational AutoEncoder (VAE). The PyTorch framework is used throughout.

1 Introduction

Unsupervised learning is characterised by learning without the requirement of labels for the data samples. Instead of referring to labels, the network extracts features and builds its own representations internally. This means that this kind of model can also be used to generate new samples (in this case, new images) because the network can create them by decoding different positions from the representations it has created in the latent space.

To find the best autoencoder model, a random search of the hyper-parameters, namely regularisers, optimisers and dimensionality of the encoded space, was implemented. The best model after training was selected as the one with the minimum average validation loss over the training epochs. This was then fine tuned by adding additional layers and testing using the dataset labels for a classification task. Finally, a Variational AutoEncoder was also implemented, where the encoder instead outputs a probability distribution. All of these tasks were very heavily computationally demanding. The data set is pictures of different items of clothing which are 28 x 28 pixel images. There are ten categories of clothing represented in the data set.

2 Convolutional AutoEncoder

2.1 Methods

The autoencoder network architecture consists first of an *encoder*, which encodes the input into a latent space of a certain, optimisable, dimensionality. The *decoder* then must reconstruct the input. The encoder and the decoder may or may not be symmetric. In this case, they are not. The autoencoder (encoder + decoder) architecture is as follows:

Encoder

1 **Input layer:** one unit.

2 **First Convolutional Layer:** input channels=1, output channels=n_channels. 3 x 3 kernel. Stride=2. Padding=1. ReLU activation. Image dimensions reduced to 14 x 14.

3 **Second Convolutional Layer:** input channels=n_channels, output channels=2*n_channels. 3 x 3 kernel. Stride=2. ReLU activation. Image dimensions reduced to 6 x 6.

4 **Third Convolutional Layer:** input channels=2*n_channels, output channels=3*n_channels. 3 x 3 kernel. ReLU activation. Image dimensions reduced to 4 x 4.

5 **First Linear Layer:** input channels=4*4*3*n_channels, output channels=64.

6 **Output Linear Layer:** input channels=64, output channels=encoded_space_dims.

Decoder

1 **Input Linear Layer:** input channels=encoded_space_dims, output channels=64. ReLU activation.

2 **First Linear Layer:** input channels=64, output channels=4*4*3*n_channels. ReLU activation.

3 **First Deconvolutional Layer:** input channels=3*n_channels, output channels=2*n_channels. 3 x 3 kernel. Image dimensions now 6 x 6. ReLU activation.

4 **Second Deconvolutional Layer:** input channels=2*n_channels, output channels=n_channels. 3 x 3 kernel. Stride=2. Image dimensions now 13 x 13. ReLU activation.

5 **Third Deconvolutional Layer:** input channels=n_channels, output channels=1. 5 x 5 kernel. Stride=2. Padding=1. Image dimensions now 28 x 28. Sigmoid activation so that the output is between 0 and 1, which it must be for a pixel.

The loss function used is CrossEntropyLoss. The images are all transformed into PyTorch tensors. An example of training epochs of the reconstruction of the image over the number of epochs is shown in Figure 1. That is from a model where n_channels was set at 5 and the encoded dimensions set at 10. The tuneable parameters are therefore the number of channels within the hidden layers, along with the encoded dimensions between the encoder and the decoder. Additionally, the optimisation function, the learning rate and weight decay rate will also be varied.

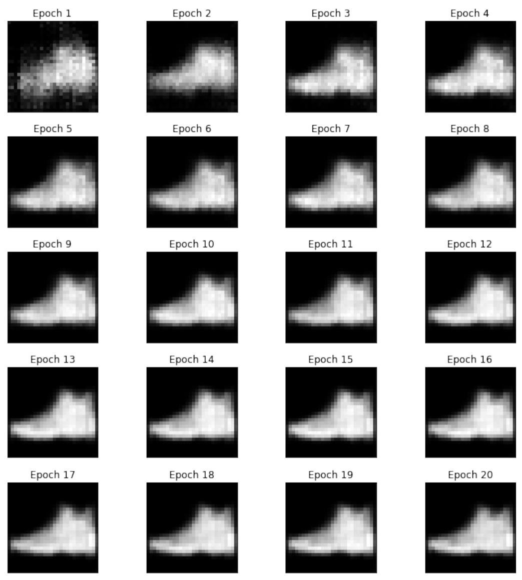

Figure 1: Example of training progress for the AE

The best hyperparameters are found by randomly sampling a new set at every iteration and then training and finding the average validation loss over the training epochs. The lowest average validation loss is used because that implies that is the model which is converging the fastest and has overall the best accuracy.

The set of parameters to be sampled for the random search is as follows:

- **Encoded space dimension:** [2,3,4,5,6,7,8,9,10]
- **n_channels:** [3,4,5,6,7,8,9,10,11,12]
- **Optimiser:** [Adam, SGD]
- **Learning Rate:** [$1e^{-5}$,$5e^{-5}$,$1e^{-4}$,$5e^{-4}$, $1e^{-3}$,$5e^{-3}$,$1e^{-2}$]
- **Number Epochs:** [30]
- **Weight Decay:** [$1e^{-3}$,$1e^{-2}$,$1e^{-1}$,$1e^{0}$]

These parameters were sampled to make 80 sets, which were then initiated and tested as individual trained networks.

2.2 Optimised hyperparameters

The best set of parameters, found after 80 iterations through a reduced training set, as iterating through the entire dataset would have taken an excessive amount of hours. They were found to be as follows:

- **Encoded space dimension:** 10
- **n_channels:** 10
- **Optimiser:** Adam
- **Learning Rate:** 0.01
- **Weight Decay:** 1e-05

The network is then trained using this optimal set of hyperparameters, but with a bigger number of epochs equal to 100. The model performance can be visualised in Figure 2. Some examples of the network are shown in Figure 3

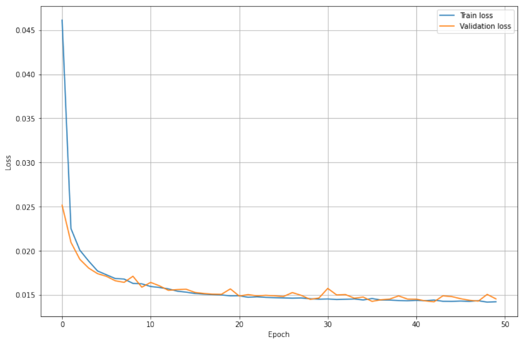

Figure 2: Training and test loss against epochs of best network

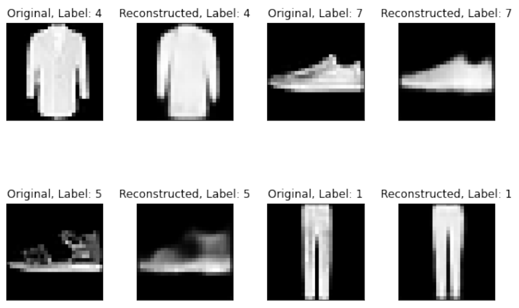

Figure 3: Results of the network trained on the optimised parameters

2.3 Latent space exploration and image generation

Using the t-Distributed Stochastic Neighbour Embedding (t-SNE) transformation, the latent space dimensions can be reduced by minimising the Kullbech-Leibler divergence between the higher and lower dimensional spaces so that they can be plotted and visualised in the requested number of dimensions, aka in two dimensions. The dimensions of the encoded space are ten, and this must be reduced to two. This can be seen in Figure 4.

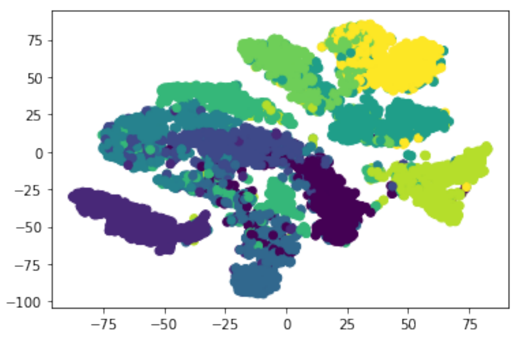

Figure 4: t-SNE data samples, coloured by label

The t-SNE representation shows that the data set is mostly categorised into distinct groups, however there is also some crossover between them. Those are the areas where the network struggles to identify between two (or more) similar categories of image.

We can then sample locations at random from the latent space and use those as input into the decoder in order to generate new images. Some examples can be seen in Figure 5.

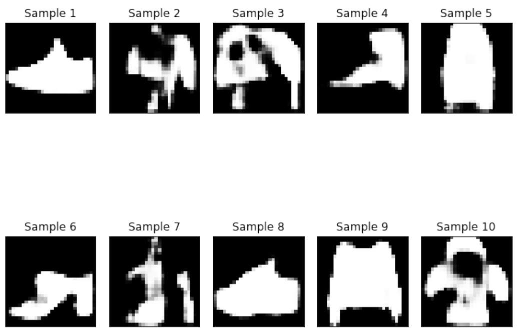

Figure 5: Generated images

These are obviously not very recognisable as images of clothing objects. That is because the sampler is taking a location at random, and it might be in the middle of the cluster for trainers and generate images such as Samples 1 and 8, or it might be somewhere not in any of the clusters. If we choose a specific region to sample from, for example at position $(-60, -50)$ from the reduced dimensionality space, then we would expect to see a clearer image as that is from the centre of a well defined cluster. In that case the decoder produces something which isn't like any of the input categories, as seen in Sample 3 for example.

2.4 Fine tuning

The next task is to fine-tune the autoencoder using a supervised classification task. For this we use the labels provided in the dataset. The encoder part is already trained with the best parameters found previously, so it does not need to be retrained. Two new layers which are added are:

- One hidden fully connected layer with 60 neurons.
- One output fully connected layer with 10 neurons.

Only the new fully connected layers need to be trained which means 1.4k parameters instead of the 40.1k total parameters of the model. Number of epochs was limited to thirty in order to save on computational drain.

The accuracy achieved was 79%. This is only slightly lower than the 85% accuracy achieved in homework one after the whole network in that case had been optimised exclusively for the classification task. This shows the power of the transfer learning technique.

3 Variational AutoEncoder

3.1 Methods

A different implementation of an autoencoder is a Variational AutoEncoder. This instead uses probabilities for mapping the inputs to the latent space which is a way of generating better samples and avoiding the 'nonsense' images seen in Figure 5. The encoder now outputs a probability distribution instead of a point in the latent space. The loss function for the VAE is now two terms, the first is the MSE and the second is the KL divergence between the probability distribution from the encoder and the normal distribution.

$$Loss = |x - x'|^2 + KLdivergence \qquad (1)$$

The decoder thus stays the same, however the encoder must be modified with the new requirements for the Variational Autoencoder.

3.2 Results

The training of the VAE over 50 epochs is seen in Figure 4.

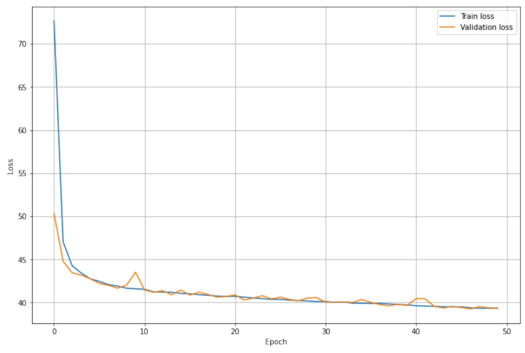

Figure 6: Training of the Variational AutoEncoder

We can see how the network has trained by viewing the input images along with the reconstructed images in Figure 7. Additionally, see Figure 8 to see the latent space of the VAE. The samples are arranged in such a way as to shift from one category of clothing item to another.

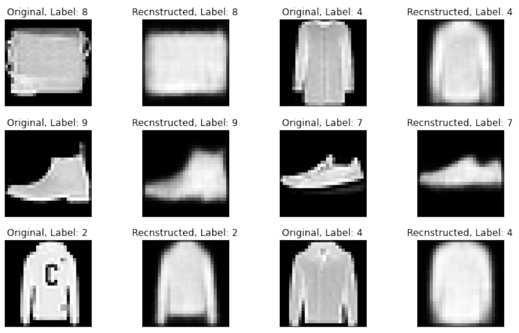

Figure 7: VAE results

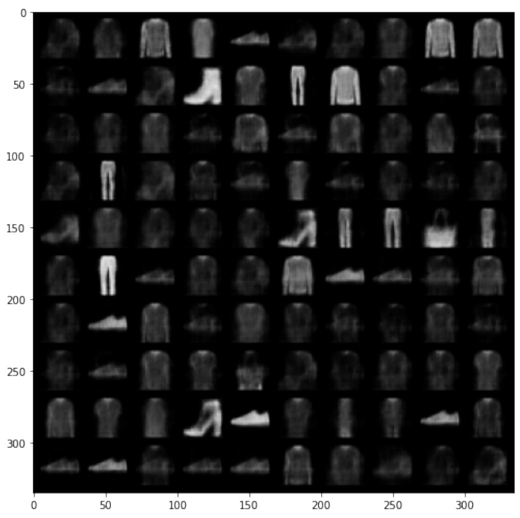

Figure 8: Latent space of the VAE.

Now we can also generate new images from the VAE, Figure 9 and see if they are better than the previous samples generated from the AE, Figure 5.

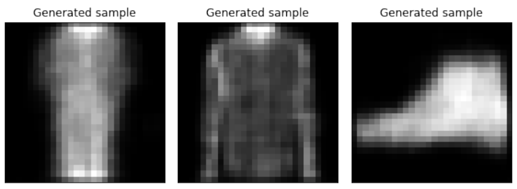

Figure 9: VAE generated images

The images in this case are all definitely recognisable as clothing, with the second two being clearly a shirt and an ankle boot.

4 Conclusion

In conclusion, the optimised AE was able to reconstruct input images successfully. However, generating images from random positions was not a good method as most of the images were not recognisable as items of clothing. Additionally, the classification task of the AE was implemented with a test accuracy of 79% which was only slightly lower than the accuracy achieved by the trained and optimised classifier implemented in the first homework.

Furthermore, the VAE was able to generate images of much better quality. This is due to it using a different type of loss function and encoding probabilities in the latent space rather than exact points.

The training in both cases could be improved by adding noise or rotations to the imput data so that the models can achieved higher generalisation and adaptability.