

Homework 08

IANNWTF 21/22

Submission until 19 Dec 23:59 via <https://forms.gle/n6ERdhYx3uBPzuGn9>

Welcome back to the 8th homework for IANNWTF. This week's topic was autoencoders.

For this week, we will try to solve a typical autoencoder problem, namely denoising images. For that we will use the MNIST Dataset again with a little bit of preprocessing to obtain noisy images. We split the homework into two parts. The first is to implement a simple convolutional autoencoder that learns embeddings from the noisy images and tries to reconstruct the underlying originals. As a second part, you will be asked to perform some analysis on the obtained latent space.

Spoiler alert: For an outstanding, we will ask you to do the same again using a variational autoencoder and then compare your analysis of the two different latent spaces.

This week we again try to keep the instructions as short as possible, so that you are challenged to try it yourself. Because the best way to learn something is to do it yourself. If you are lost, check the hints for advice or visit us during the coding support Sessions (their purpose is to support you with any trouble you encounter, so really feel free to drop by!). Also, you can always refer to the code snippets and notebooks provided in Courseware.

1 Data set

This week we will work with the MNIST dataset again, with which you will already be familiar. However, this week the preprocessing and constructing the dataset will be slightly different: We will not concern us with the actual labels, but instead we will transform the MNIST images to be noisy and then use the original images as targets. In figure 1 you see an example of how your data should look like after applying the preprocessing.

You should consider the following points when building your data pipeline

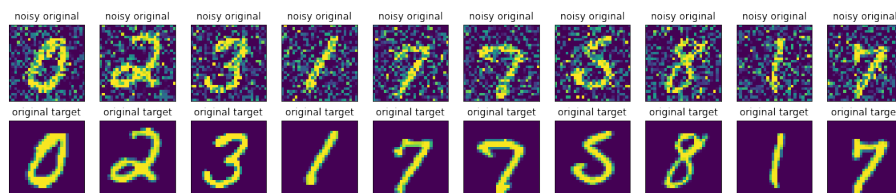


Figure 1: MNIST Variation: First row shows original input with added noise, second row shows the original input to be used as targets.

- Load the dataset and construct a `tf.Data.Dateset` for testing and training using **the images only**.
- Normalize the images and make sure that you have sensible dimensions¹.
- Add noise to your input images²³.
- Perform other necessary or beneficial preprocessing steps⁴.
- Make sure your data pipeline is efficient⁵.

2 Model

Implement a convolutional autoencoder (and a variational autoencoder for an outstanding).

2.1 Convolutional Autoencoder

- The Autoencoder should consist of an encoder and a decoder, which can be called independently. ⁶
- Encoder: The encoder should reduce the size of feature maps like a CNN.⁷⁸ At the end of the encoder, flatten the feature maps and use a dense layer to produce an embedding of a certain size. ⁹
- Decoder: The decoder takes the embedding from the encoder as input. Use a dense layer to restore the dimensionality of the flattened feature maps from the encoder and reshape the resulting vector into feature maps again. Use upsampling or transposed convolutions to mirror your encoder. As an output layer, use a convolutional layer with one filter and sigmoid activation to produce an output image.

2.2 (Variational Autoencoder)

For the implementation of a Variational Autoencoder refer to online sources. There are good tutorials that do a better job at explaining the implementation than we can do in a few lines of text here. There are also different ways to implement VAEs in tensorflow. Here is a [guide](#) using only tensorflow. There is also the [tensorflow probability](#) library, which provides probabilistic layers. Have a look at [this tutorial](#), which explains how to build VAEs with tfp. You can also use any other source that you can dig up. Please explain in detail how sampling and the loss are handled in your implementation so we know you did not just copy and paste without understanding whats going on. If you run into troubles in this part, still feel free to come to the coding class.

3 Training

Then train your network(s). If you are referring to online sources here, do not get confused! Normally, autoencoders are unsupervised and do not deal with targets, thus you would compute the loss between the input and the prediction. As we are working with a de-noising example, however, we actually have a target, namely the original image. We thus want to compute the loss between the original image (as target) and the reconstructed image obtained from the noisy input¹⁰. While training, it is nice to plot some example images from the test set with their reconstructed counterparts to visualize the training progress.

¹¹

4 Latent Space Analysis

Embed the first 1000 images of the test set using the encoder. Reduce the dimensionality of the embeddings to two using the t-SNE algorithm.¹² Then plot the data points, coloured according to their class. Evaluate the result. Is it what you expected? Further, interpolate linearly between the embeddings of two images and plot the reconstructed images.

For the Variational Autoencoder part do the same and compare the results.

5 Outstanding Requirements

General Reminder: Rating yourself as outstanding means that your code could be used as a sample solution for other students. This implies providing clean understandable code, descriptions and types of arguments and explanatory comments when necessary.

This week the main part of the outstanding distinction is to **implement a Variational Autoencoder in addition and to compare the analysis to the standard Autoencoder.**

The following points are still required but you should be able to copy and paste most of it by now.

- A proper Data Pipeline (that clearly, understandably and efficiently performs necessary pre-processing steps and leaves out unnecessary pre-processing steps).
- Clean, understandable implementations of your data set.
- Comments that would help fellow students understand what your code does and why. (There is no exact right way to do this, just try to be empathic and imagine you're explaining topic of this week to someone who doesn't now much about it.)
- Nice visualizations of the losses and accuracies. (You don't have to plot every epoch. Once in the end is enough. Although some print statements to keep track of the performance during training are welcome.)
- Nice visualization and explanations of the latent space analysis and the interpolation.

Soft requirement: Have lots of fun!

Notes

¹Although MNIST Dataset are black and white and have no color channels, you should add third dimension so the autoencoder can handle the input better. Use `tf.expand_dims(, axis=)`

²Create a random noise tensor and add it to the image tensor. You can introduce a hyperparameter to control how noisy your data will be. Make sure to keep the images within the right scale (between 0 and 1)

³You might want to make use of the functions `tf.random.normal(, mean=, stdev=, shape=)` and `tf.clip_by_value(, min=, max=)`

⁴Shuffling, Batching, Prefetching....

⁵You might make use of mapping when normalizing, reshaping and adding noise.

⁶The easiest way to do this is with model subclassing. Define separate models for the encoder and decoder and initialize them in the autoencoder constructor.

⁷You can either use convolutional layers with stride 2 for subsampling or convolutional layers with stride 1 followed by a pooling layer

⁸subsampling two times i.e. to a size of 7x7 should be enough

⁹As a rule of thumb, the smaller the embedding, the harder the training and the more layers required. If experimentation fails you, try 10 as an embedding size.

¹⁰If you have prepared your dataset correctly, you should be able to recycle your usual testing and training functions without many changes.

¹¹Training Hyperparameters for orientation:

- epochs: around 10
- learning rate: try something around 0.001
- optimizer: Adam

¹²t-SNE is a dimensionality reduction algorithm particularly suited for visualization. You don't have to implement it yourself, you can use an existing library, e.g. <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html>.

If you want to know how it works have a look at [this blogpost](#). For some guidance on what can be and what can not be interpreted from the results refer to [this Distill article](#)