Homework 08

IANNwTF

December 13, 2022

This homework's deadline is *09.01., 23:59*.
Submit your homework via https://forms.gle/ApAZ5ubY8ewgNmJA9
Remember to review the work done by two of your fellow groups!

Contents

1 Recap on Reviews

Welcome back to your eighth assignment in IANNwTF. Here's your weekly short refresher on how to do reviews.

In addition to handing in your homework, you will have to review last week's homework of two groups and note down which group you reviewed on the homework submission form. This requires you to find two other groups and have a short (10 min) meeting where you go over each others homework submission, discuss it and then write a short summary of what you discussed on each others forum pages. We recommend using the Q&A timeslots for this purpose, but you can meet however and whenever you like. The main review part of this should be the discussion you have together. The written review in the forum should merely be a recap of the main points you discussed so that we can see that you did something and the reviewed group has access to a reminder of the feedback they received. **If there are any open questions regarding your or any other groups code afterwards, please feel invited to discuss this with us in the QnA sessions, so we can help you sort out how a perfect solution would have looked like!** Just to make this obvious: We will not penalize any submission (by you or any of the groups you reviewed) based on such questions. The purpose is helping you understand and code better only.

As to how a discussion could look like, you could for example have the group being reviewed walking the other two groups through their code. The other two groups should try to give feedback, e.g. "I like how you designed your data pipeline, looks very efficient.", "For this function you implemented there actually exists a method in e.g. NumPy or TensorFlow you could've used instead." or "Here you could've used a slightly different network architecture and higher learning rate to probably achieve better results.", and note down their main points in the forum.

Important! Not every member of every group has to be present for this. You could for example implement a rotation where every group member of yours only has to participate in a review every third week.

2 Assignment: 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 to do the same again using a variational autoencoder and then compare your analysis of the two different latent spaces.

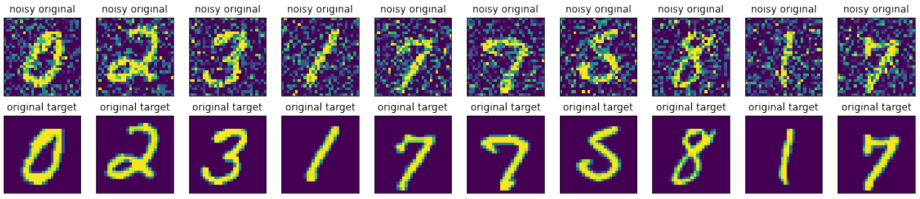

Figure 1: MNIST Variation: First row shows original input with added noise, second row shows the original input to be used as targets.

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 struggle with any of the following tasks, remember to make use of all the resources available to you. Come to our Q&A sessions, post your questions in the forum, and don't forget that most precious resource of them all: your fellow students!

2.1 Prepare the Dataset

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:

- 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 dimensions[1].
- Add noise to your input images[2][3].
- Perform other necessary or beneficial preprocessing steps[4]
- Make sure your data pipeline is efficient[5].

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

[2] Create 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)

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

[4] Shuffling, Batching, Prefetching....

[5] You might want to make use of mapping when normalizing, reshaping and adding noise

2.2 The model

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

2.2.1 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[7][8]. 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.

2.2.2 Outstanding: (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, feel free to come to the Q&A or ask on the forum.

2.3 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

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

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

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

[9] As 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

original image (as target) and the reconstructed image obtained from the noisy input[10]. While training, it is nice to plot some example images from the test set with their reconstructed counterparts to visualize the training progress[11].

2.4 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.

2.5 Reminder: Outstanding Requirements

General Reminder: Don't forget that you have the option to rate your homework as a outstanding submission. 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.)

[10] If you have prepared your dataset correcty, you should be able to recycle your usual testing and trainnig functions without many changes

[11] Training Hyperparameters for orientation:

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

[12] 2t-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. `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.

- 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!**