Homework 09

IANNwTF

January 11, 2023

This homework's deadline is *23.01., 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 ninth 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: GANs

Welcome back to the 9th and first homework of the year 2023 in IANNWTF. Don't get too excited, there will be a few more before the semester is over ;).

This week you have learned about GANs. Perfect timing! In this homework, we will use your acquired skills to become candle makers to relive the Christmas spirit of the recent holidays!

The goal of this week is to code a simple GAN and train it on an image dataset consisting of sketches—we will only focus on the category 'candle' in this homework. When training, you should visualize your training matrices as well as generate candle images using a random seed to assert the visual quality of your GAN. After training, you should be able to generate acceptable candle images—3,4 or 5 depending on how far it is till Santa Claus is coming to gift

you motivation for the new year.

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 Quick, Draw! Dataset. The Quick Draw Dataset is a collection of 50 million drawings across 345 categories, contributed by players of the game Quick, Draw!.

We will download the data directly from Google this time. As this process is a little more involved, have a look at this Notebook—here. There, you can find an example of how to download the images into a NumPy array to work with. We will restrict ourselves to sketches of candles only, but feel free to toy around with the other image categories later on. You can also find the notebook as a .ipynb in the homework folder on Courseware. Once you have downloaded the data, you should be able to build your data pipeline from the NumPy array as usual. Consider the following steps:

- Construct a tf.Data.Dataset object[1].
- The images come as (1,784) pixel arrays. You should make sure to reshape them correctly, obtaining small images [2].
- Normalize: Bring the images' values into a sensible range.
- Perform other beneficial or necessary processing steps [3]

2.2 The Model

Your task is to implement a GAN.

Create a **Discriminator** class:

- `__init__(self)`: The underlying idea of the discriminator is to perform downsampling and then outputting a vector of probabilities indicating whether the input was fake or real[4].

[1] using `tf.data.Dataset.from_tensor_slices()`
[2] Your resulting images should be of shape (28,28,1)
[3] Batching, Shuffling, etc
[4] This is basically just like a CNN architecture for a binary classification task, thus you have a Dense layer with a single neuron in the end.

- `call(self, x, training)`: This is just a simple call the way you know it, but be aware of the training flag if you decide to use certain optimization and regularization techniques[5].

Create a **Generator** class:

- `__init__(self)`: The generator takes a random point from the latent space[6] and returns a generated image. The latent space comes as a 1d vector, so to receive a 2d image you need to reshape it. To increase the size, use upsampling techniques like Conv2DTranspose[7][8]. In the last layer, perform a convolution with 1 feature map and tanh activation. This will be the output image.

- `call(self, x, training)`: Same as for the Discriminator.

2.3 Training

There is something novel this week: You will actually need to make some major adjustments to your `training_step()` function and the overall training procedure.

In each training step, the generator is fed with random noise[9] and creates images from it.

The discriminator sees a batch of true images and a batch of the generated images. The loss of the discriminator is based on how well the discriminator detected fake images as fake and real images as real[10].

The loss of the generator is estimated by how well the generator was able to fool the discriminator. The more images the discriminator falsely classified as real, the better our generator works and the smaller the Binary Cross Entropy loss between the discriminator's predictions and all labels as true=1.

Training Hyperparameters for orientation can be found in this footnote[11].

[5] BatchNorm and Dropout might be useful.

[6] You can try different latent space sizes. Something around 100 would be a good place to start

[7] It is common to use even-sized kernels in transposed convolutions to avoid checkerboard artefacts, Deconvolution and Checkerboard Artifacts

[8] You can also use additional convolutional layers in between upsampling layers, which gives the generator more parameters to work with

[9] Create random noise, possibly with your own function, using tf.random.normal(). The size of your noise vector is the size of your latent space.

[10] Compute the binary cross entropy between the generator's output on fake images and all labels = 0. Similarly, compute the BCE between the generator's output on the real images and all labels = 1. Add them both to receive the resulting loss of the `discriminator.tf.ones_like()` and `tf.zeros_like()` could be helpful for creating the true labels as a comparison.

[11]

- epochs: 10 is a good starting point, then see if you want to train longer or if you are already satisfied with the quality of the generated images
- optimizer: Adam or RMSprop

We recommend visualizing the losses and output images using TensorBoard. Be aware that, unlike in your previous architectures, the goal is not to get the loss as low as possible. The loss of the generator or discriminator going against zero is actually a sign of training failure.

For evaluation, you should therefore create some random latent vectors before training and feed them through the generator regularly. You can plot the resulting images to evaluate training.

Training GANs is messy, and it is sometimes hard to pinpoint the reason why training fails. But as our input images are not of the greatest quality anyway, your GAN should in the end be very capable of approximating these sketches.

2.3.1 Optional: Is it Christmas?

Now let's have a little fun. For all the non-Germans among you, let's recap a popular Christmas poem:

Advent, advent, a candle is burning
First one, then two, then three, then four,
Then the Christkind is standing in front of the door.
And when the fifth candle is burning
you missed Christmas.

Advent, Advent ein Lichtlein brennt.
Erst eins, dann zwei, dann drei, dann vier.
Dann steht das Christkind vor der Tür.
Und wenn das fünfte Lichtlein brennt,
dann hast du Weihnachten verpennt.

The poem of course is referring to the tradition of advent wreaths.

Now build your own wreath: Write a function called `is_it_christmas(model)` that generates you either 3, 4 or 5 new candles dependent on the current date according to the poem[12].

A possible output for the day of the midterm exam can be seen in figure 1[13]

2.4 Reminder: 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 is just a recap but also includes some specifics for this week:

[12] You can make use of the `datetime` package

[13] You can choose to convert the image output of the GAN back to a white background or leave it black

Figure 1: At the date of the midterm three candles would've been burning

- 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 the topic of this week to someone who doesn't know much about it.)
- Nice visualizations of the losses and accuracies. Also, display generated images for each epoch to assess the visual quality.
- A proper is `is_it_christmas` function.

Soft requirement: **Have lots of fun!**