EE 452 - COMPUTER VISION

Assignment 2: Generative Adversarial Network (GAN) Application

---

Instructor - Dr. Muhammad Farhan

Authors - Kuldeep Dileep (kl04008)

Contents

1 Question 1 **3**

1.1 Part 1 . . . 3

1.2 Part 2 . . . 3

1.3 Part 3 . . . 3

1.4 Part 4 . . . 4

1.4.1 Results with Stanford car Dataset . . . 4

1.4.2 Results with Cifar-10 Dataset . . . 5

1.5 Part 5 . . . 6

1.6 Part 6 . . . 6

1.7 Part 7 . . . 7

1.7.1 Part a . . . 7

1.7.2 Part b . . . 7

1.7.3 Part c . . . 8

2 Question 2 **10**

2.1 Part 1 . . . 10

2.2 Part 2 . . . 10

2.3 Part 3 . . . 10

3 Question 3 **13**

4 References **14**

1 Question 1

1.1 Part 1

Originally, a notebook was provided to load Stanford dataset but then due to some technical glitch the Stanford car dataset was no longer available on their official website so I had to load it from Kaggle. To do so I followed the standard procedure of loading the API of the dataset on Kaggle. Once when the rar folder was unzipped and images were saved in the given directory then I preformed pre-processing on data such as scaling all the imagess to 32x32 dimension and normalizing the image to [0,1] range.
To load Cifar data, I used the built-in function provided by the tensorflow-keras which automatically loads 50,000 train images and 10,000 test images.

1.2 Part 2

Discriminator of a Generative Adversarial Network (GAN) is just like any other binary classifier. In DCGAN, discriminator comprises of 2D convolutional layers. The model has an input convolutional layer followed by 3 convolutional layers that downsamples the image from 32x32x3 to 4x4x3 where the number of neurons increase as we go deeper from 64 to 128 to finally 256. I have used a simple architecture because creating a complex and deeper architecture could lead to vanishing gradient problem.

1.3 Part 3

In most of the literature, 100-d vector was used as the latent vector which was carried on by the future researchers in their work because latent vector with the dimension of 100 gave satisfactory results therefore, others used it as is in their work since, dimension of the latent space is just like any other hyper parameter which needs fine tuning but since 100-d was already fine tuned therefore, it wasn't changed. I tried changing it to observe the affect of the dimension of the latent space on the images generated but I didn't notice any difference

maybe because of the less complexity of the dataset. Figure 1.1 shows the resulting loss curves where figure (a) is when the latent dimension is 20, figure (b) is when the latent dimension is 100, and figure (c) is when the latent dimension is 500. As it is evident from figure 1.1 that there is no significant difference in the performance of the adversarial network.

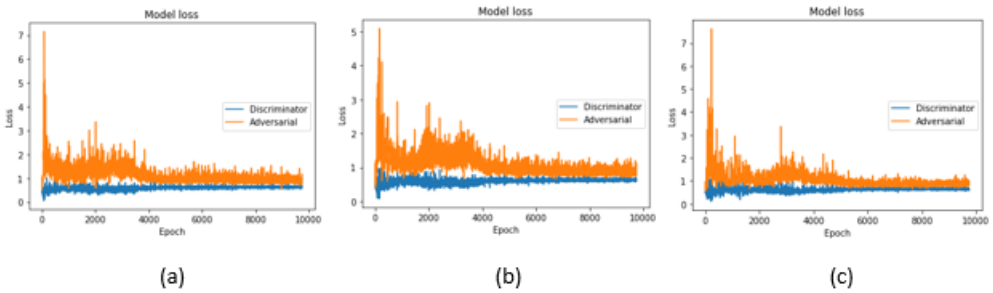

Figure 1.1: Loss curves at varying latent dimension

1.4 Part 4

1.4.1 Results with Stanford car Dataset

Figure 1.2 shows the result achieved on Stanford car dataset with the adversarial networks as defined in the previous subsections. The result seems good enough as one can identify cars in them. I intended to train my GAN for 1000 epochs at batch size of 16 but it was taking too long to train and the notebook's runtime kept disconnecting so the result shown is at 180 epochs because that is the maximum I got to which took more than 4 hours and after that the runtime would disconnect. I also tested it on CIFAR-10 dataset and results are shown in next section.

Figure 1.2: Result on Stanford car dataset

1.4.2 Results with Cifar-10 Dataset

Figure 1.3 shows the result achieved on Cifar-10 dataset with the adversarial networks defined in the previous subsection. They seem like a decent results. Similar scheme of hyper-parameter tuning was followed as discussed in previous sections.

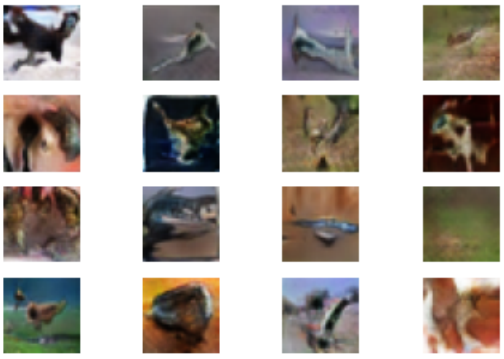

Figure 1.3: Result on cifar-10 dataset

1.5 Part 5

Figure 1.4 shows the loss curves of discriminator (in blue) and adversarial network (in orange). From the graph we can conclude that the loss in both the adversarial networks is reducing with the increase in epochs.

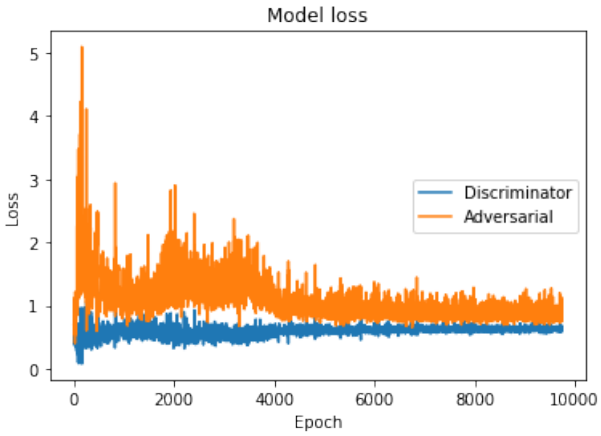

Figure 1.4: Loss curve on Cifar-10 dataset

1.6 Part 6

Figure 1.5 shows the 20 images generated by the GAN trained using the function generate samples.

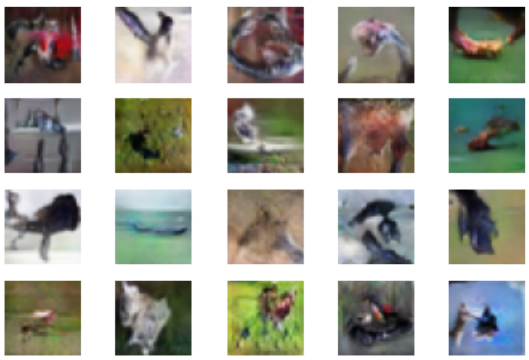

Figure 1.5: Sample images generated from the trained GAN

1.7 Part 7

1.7.1 Part a

Yes, we can use VGG-16 network as the discriminator in this GAN. To do so we will load the pre-trained VGG16 model by Keras and we will freeze all the layers to preserve the original weights. We will not include the original top layer which has a default input size of 224x224 whereas, we will change the input shape of the input layer to 64x64x3. Moreover, we will also remove the last layer as by default in VGG16 it has 1000 output classes whereas, in our case we only require single neuron. We will UpSample the incoming input image from 32x32 to 64x64. Then we will define a sequential model and add VGG16 model to it and will finally add the output layer with single neuron.

The resulting network as described above will not be a DCGAN because the discriminator of a DCGAN only includes convolutional layers whereas, the last 3 layers of VGG16 are fully connected (dense layer) therefore, to make this a DCGAN we will have to remove the dense layers of VGG16 and to achieve this we will iterate through each layer of VGG16 and add only Convolutional layers of VGG16 to the sequential model to make this a DCGAN.

1.7.2 Part b

VGG-16 can be considered as a discriminative network because discriminator is essentially a classifier that learns the feature of a real image sample and then maps the features of a generated image on it to classify it as a real or fake image which can also be achieved by VGG16 as it also uses convolutional layers to extract features with increasing number of filters as we go deeper into network. Detailed description of how VGG16 can be used a discriminator is provided in the previous subsection. On the contrary, VGG16 can't be used as a generative network because in generative model we UpSample the latent space to reach the dimension of the original image which can't be achieved by VGG16 because it's architecture doesn't include up-sampling or Conv2DTranspose layer.

1.7.3 Part c

Figure 1.6 shows the resulting loss curves of the adversarial networks when implemented as discussed in previous sections. One can notice that the loss of discriminator decreases but the loss of adversarial network starts increasing after few epochs which is not desired therefore, as a fine tuning measure, I added batchnormalization and drop out layers at each layer of the generator which resulted in the loss curves as shown in figure 1.7 which also doesn't seem ideal. After browsing about possible ways to solve this I found a suggestion of adding Gaussian noise after each layer of generator to resolve this issue. After doing so, I received the curves as shown in figure 1.8 which also don't seem ideal. All in all, I spent days trying to fine tune my VGG16 model but couldn't achieve good results. A possible reason that I could think of to justify such behavior of my VGG16 based DCGAN is that since the VGG16 is trained on different categories therefore, it has learnt different set of features and since, the images of Cifar10 don't lie in that feature space therefore, VGG16 fails to map the given image to the targeted feature space. Since, the VGG16 doesn't know what a real image looks like therefore, it fails to discriminate the genereated image as real or fake and therefore, generator doesn't learn anything.

Moreover, another possible justification could be that the VGG16 has a complex architecture which makes it too good and therefore, it wins all the time and the generator doesn't learn anything which could lead to vanishing gradient problem. To resolve this I tried increasing the complexity of generator by increasing the number of filters at each layer of generator to 256 but the run-time kept crashing.

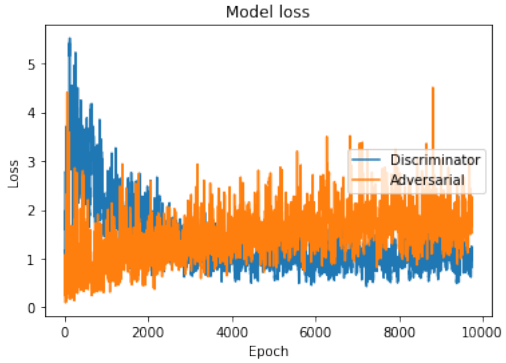

Figure 1.6: Loss curves of VGG16

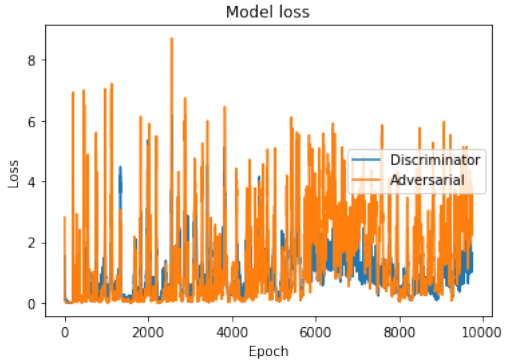

Figure 1.7: Loss curves VGG16

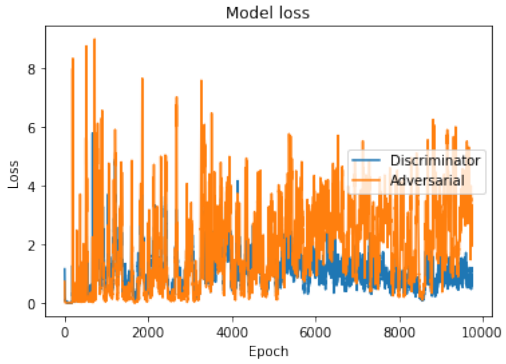

Figure 1.8: Loss curves of VGG16

2 Question 2

2.1 Part 1

For this purpose we will use Super Resolution GAN (SRGAN) because SRGAN uses adversarial network to produce high resolution images. It takes a high resolution image and down-samples it to a low resolution image and then uses this pair to train its network. The generator up-samples the low resolution image to produce high resolution image which is then fed to a discriminator to classify it as high resolution image and the loss is propagated back to generator. In this scenario, the student had applied large median filter to blur the image resulting in a low-resolution image and lost the original high resolution image therefore, SRGAN can be used to restore the original HD images.

2.2 Part 2

For this purpose we will use Cycle-Consistent Adversarial Networks because it captures and learns characteristics of an image from domain X and maps it to an image from domain Y in the absence of paired images of the two domain. In this scenario, the intern aims to perform Image-to-Image translation from domain x i.e. panda's skin texture to domain y i.e. bear's skin texture and vice versa but the intern doesn't has paired images of the two domain that could have been used to achieve a direct mapping between them therefore, intern must use Cycle-Consistent Adversarial Network proposed by authors in [1].

2.3 Part 3

For this purpose we will use Style transfer technique using Cycle-Consistent GAN because in style transfer we use two images one is referred as content image and the other as style reference image and we blend them such that the resulting image looks like the content image but is painted in the style of the style reference image. This is achieved by using convolutional network to extract content statistics of the content image and style statistics of the style

reference image and combining them into a single image. For instance, in figure 2.1 image (a) is the dog image referred as the content image, image (b) is the painting of Wassily Kandinsky referred as the style reference image and image (c) is the resulting image of style transferred from image (b) to (a) which looks like if Wassily Kandinsky has painted it.

Figure 2.1: Example of style transfer

Similarly, in the given scenario the daughter wants her family photos to look like they have been painted by Von Gogh so to achieve this style transfer must be performed using Cycle-Consistent GAN.

To perform style transfer using Cycle-Consistent GAN, we don't require paired images instead we will use unpaired images from domain x and y to train our network. Unlike traditional GAN, CycleGAN trains two generators and two discriminators. Generator A (GenA) learns mapping from domain X to Y such that it takes an image from domain X and converts it into an image that resembles image from the target domain Y. The other generator, Generator B (GenB), learns the mapping from domain Y to X such that it takes an image from the target domain Y and converts it into image that resembles image from the source domain X. Similarly, there are two discriminators, Discriminator A (DisA) is responsible to discriminate images generated by GenA and the images from source domain X whereas, Discriminator B (DisB) is responsible to discriminate images generated by GenB and the images from target domain Y. This cycle between source and target domain allows CycleGAN to reconstruct source image and vice-versa.

Lastly, CycleGAN employes two types of loss functions namely Adversarial loss and Cycle

Consistency loss. Adversarial loss is the loss between real images from domain X and Y and generated images from GenA and GenB whereas, Cycle Consistency loss is the loss of the real and reconstructed images that is when image from domain X is passed on to GenA it generates image that resembles an image from domain Y and when the generated image is passed on to GenB it must generate back the original source image. Figure 2.2 shows the forward Cycle-Consistent mapping where 'G' is GenA and 'F' is GenB:

$$x \rightarrow G(x) \rightarrow F(G(x)) \approx x$$

Figure 2.2: Example of style transfer

3 Question 3

Unlike conventional CNN, Recurrent Neural Network (RNN) is used to deal with sequential data where along with spatial information the temporal information is also equally important to make future prediction. Video data with fight scene sequences is one such example where both spatial and temporal information is required to make future scene prediction therefore, one can't use a traditional Neural Network because in it all the inputs are independent of each other but when predicting the future frames in a video the knowledge of the previous frames up until that point is required to understand the context of the video which can be incorporated by the memory in RNN that holds the information of the past inputs and uses it along with the current input to make the future scene prediction.

Generative Adverserial Netowork (GAN) employs deep learning techniques to train generative models to generate new synthetic images which is classified by the discriminator as a real or fake image. A traditional GAN lacks the ability to generate a specific type of image therefore, a new type of GAN was introduced called Conditional GAN (cGAN). In cGAN image generation can be conditioned on class label.

To solve this problem, I propose a combination of RNN and cGAN that will require a sequence of images and score of the next move as the input and it will give the image of the next move as the output. If the score from the user is less than and equal to 2.5 then the class label is bad move otherwise it's good move. The generator here will be a Recurrent Neural Network which will receive the sequence of images along with the class label and it will produce the next fight move based on the score. The generated image along with the class label and a real image image from the same class is fed to the discriminator which classifies the generated image as real or fake and propagates the loss back to generator to improve the quality of image generation.

4 References

[1] J. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2242-2251, doi: 10.1109/ICCV.2017.244.

[2] The following site was referred to understand the working of Cycle-Consistent Adversarial Network: https://medium.datadriveninvestor.com/style-transferring-of-image-using-cyclegan-3cc7aff4fe61