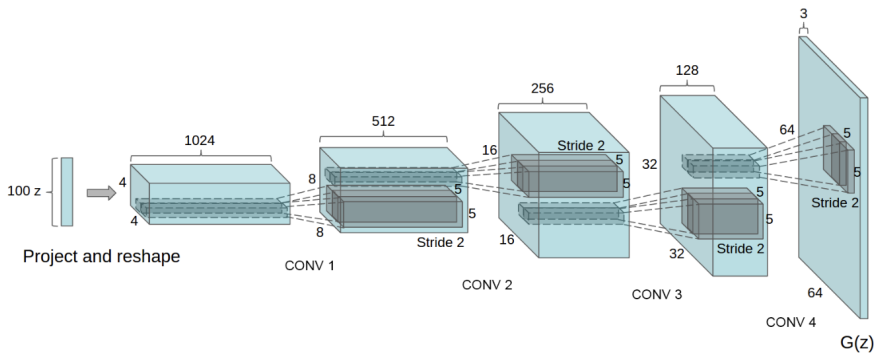
1 Summary

Since I was able to successfully load the dataset and explore the dataset I have started implementing the GAN. I have looked at the GAN paper and figured out some details that might help me in implementing the GAN. I am using `pytorch` library to implement the generator and discriminator but faced some issue in setting up the models. Mainly the issues are with shape mismatch in vector related operations. The `pytorch` documentation has been really helpful in fixing the mistakes based on the type of inputs that the functions accept. I am taking a reference from `DCGAN` implementation from pytorch examples repository[1]. One thing that I am planning to do after loading the data is to normalize the input data (understood normalization through the lecture).

2 Model Architecture

2.1 Generator

The generator architecture is inspired from the architecture given in the `DCGAN`[2] paper.



This is how it's set up using `pytorch`, number of channels in various hidden layers is different than given in the above architecture. I am inputting 100 latent features to the generator and it generates a 64×64 single channel (greyscale) image.

```
nn.Sequential(
    # input is Z, going into a convolution
    nn.ConvTranspose2d(     nz, ngf * 8, 4, 1, 0, bias=False),
    nn.BatchNorm2d(ngf * 8),
    nn.ReLU(True),
    # state size. (ngf*8) x 4 x 4
    nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
    nn.BatchNorm2d(ngf * 4),
    nn.ReLU(True),
    # state size. (ngf*4) x 8 x 8
```

```
    nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
    nn.BatchNorm2d(ngf * 2),
    nn.ReLU(True),
    # state size. (ngf*2) x 16 x 16
    nn.ConvTranspose2d(ngf * 2,     ngf, 4, 2, 1, bias=False),
    nn.BatchNorm2d(ngf),
    nn.ReLU(True),
    # state size. (ngf) x 32 x 32
    nn.ConvTranspose2d(    ngf,      nc, 4, 2, 1, bias=False),
    nn.Tanh()
    # state size. (nc) x 64 x 64
)
```

2.2 Discriminator

Discriminator has a kind of similar architecture as generator but reversed. It takes a single channel 64×64 image and outputs a single value through a sigmoid function.

3 Next Steps

Now that I have set up the higher level details of the model, I need to set up the forward propagation as well as decide on the optimizer to use and how I am going to do the back propagation and perform training of the network. I also have to decide various hyper-parameters of the model and see which values yield the best results. Hopefully I will be able to generate some images after I setup the initial training.

References

[1] `https://github.com/pytorch/examples/tree/main/dcgan`

[2] `https://arxiv.org/pdf/1511.06434.pdf`