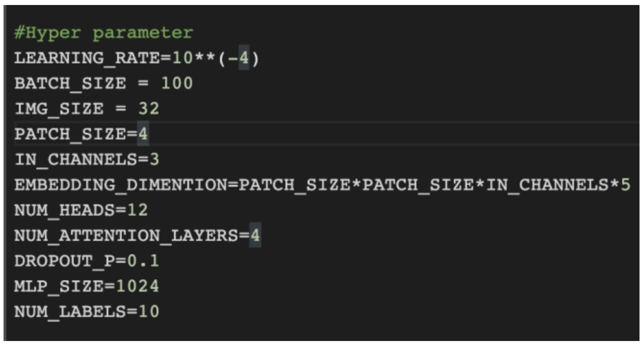
Vision Transformer Report

Puspak Priyabrata Sahu 21676 MTECH CSA

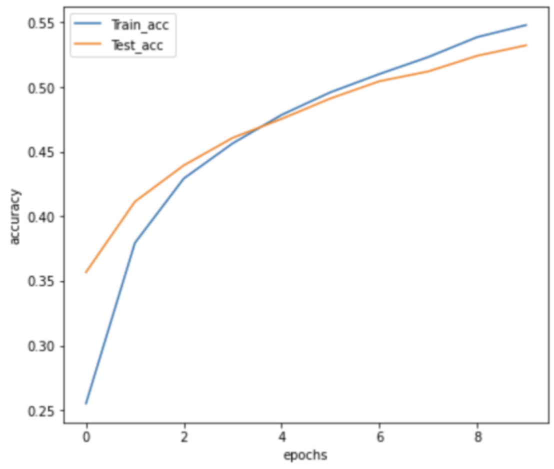
A. *Implemented a basic version of **Vision transformers**, that first divides an image into patches and then passes them through a set of multi-head self-attention modules to perform classification.* *(As given in paper)*

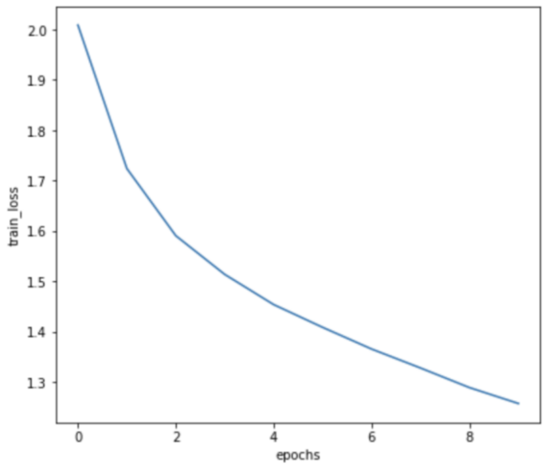


```
#Hyper parameter
LEARNING_RATE=10**(-4)
BATCH_SIZE = 100
IMG_SIZE = 32
PATCH_SIZE=4
IN_CHANNELS=3
EMBEDDING_DIMENTION=PATCH_SIZE*PATCH_SIZE*IN_CHANNELS*5
NUM_HEADS=12
NUM_ATTENTION_LAYERS=4
DROPOUT_P=0.1
MLP_SIZE=1024
NUM_LABELS=10
```

B. **Trained with CIFAR-10** with above Hyper parameters and number of encoder layers – 4 for 10 epochs and got the following results.

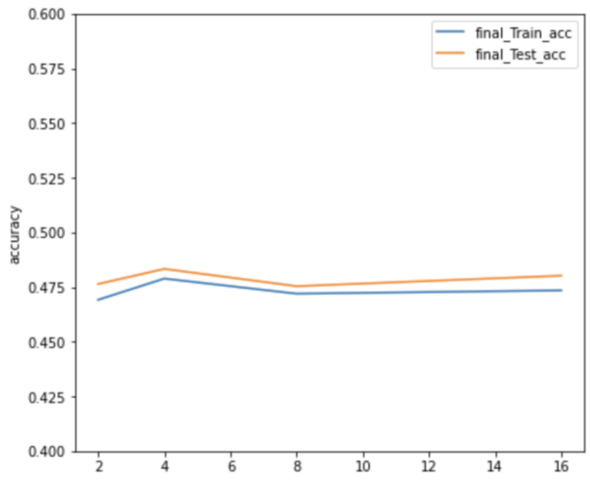
- As shown below, we can see in the first few epochs the train-accuracy < test-accuracy but as we go on the model learns the training data and overfits. That's why the train-accuracy goes on increasing linearly but the test-accuracy becomes stagnant.
- As I increased the epochs, **train-accuracy went to 1 but test accuracy stayed near to 0.6 on higher epochs**. It was remembering the train data as whole.
- Reason could be smaller dataset CIFAR-10 (Only 50K data) and non-high quality image only(32 X 32).
- Transformers usually perform better on larger datasets; on smaller datasets they perform worse than CCNs.
- **OBSERVATIONS:** Using higher embedding size and MLP size was time consuming but good accuracies were observed.

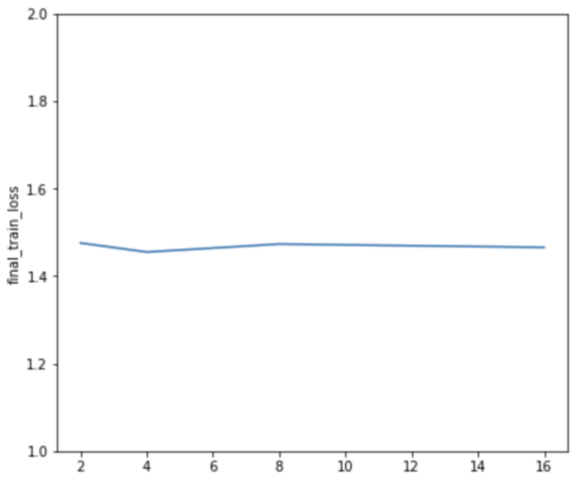




C. Patch-Size Experiment

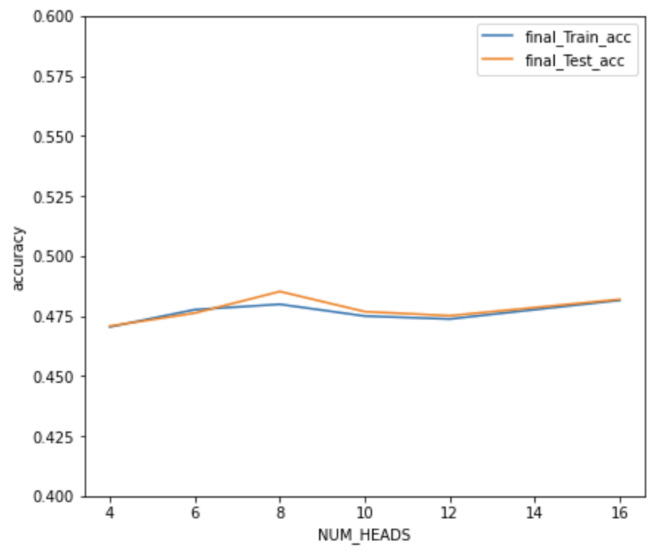
- The patch size is an important hyperparameter in the ViT model because it determines the **granularity of the image feature**s that the model can learn
- If the patch size is too large, the model may miss important details or patterns in the image, since the information in each patch is averaged over a larger area. On the other hand, if the patch size is too small, the model may become computationally expensive or may be overfit to the noise in the image.
- Using a larger patch size can increase the receptive field of the model and allow it to capture more global context information from the image
- I experimented with patch-size **2*2,4*4,8*8 and 16*16.**
- Got almost similar accuracies in all sizes, but with **patch-size 4 X 4 , slightly better accuracies were found in train and test data.**
- For higher epochs significant changes were seen. Due to lack of computational resources, I could not plot it. It was taking way more time.

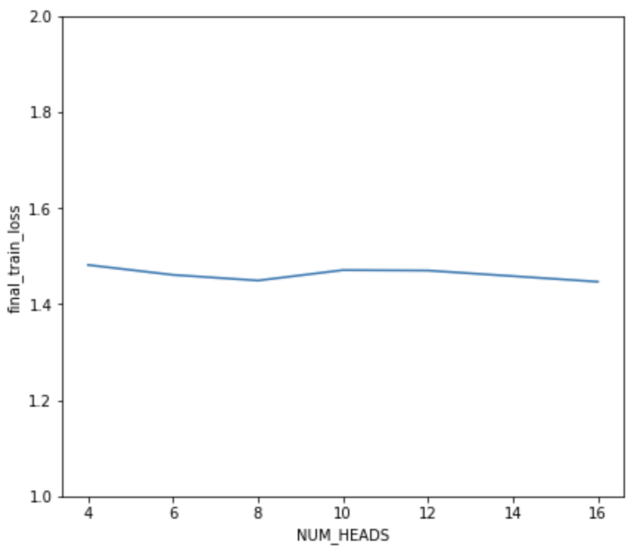




D. EXPERIMENT ON ATTENTION HEADS

- The attention heads play a critical role in allowing the ViT to model **complex relationships between different parts** of the image.
- By using multiple attention heads in parallel, the model can learn different aspects of the input image simultaneously, allowing it to capture a diverse set of features and relationships
- Just like words learns attention with respect to each other's in NLP, here ViT learns relation of patches among themselves.
- I experimented with number of attention heads **4,6,8,10,12,16** as number of attention heads should be divisible by embedding dimension.

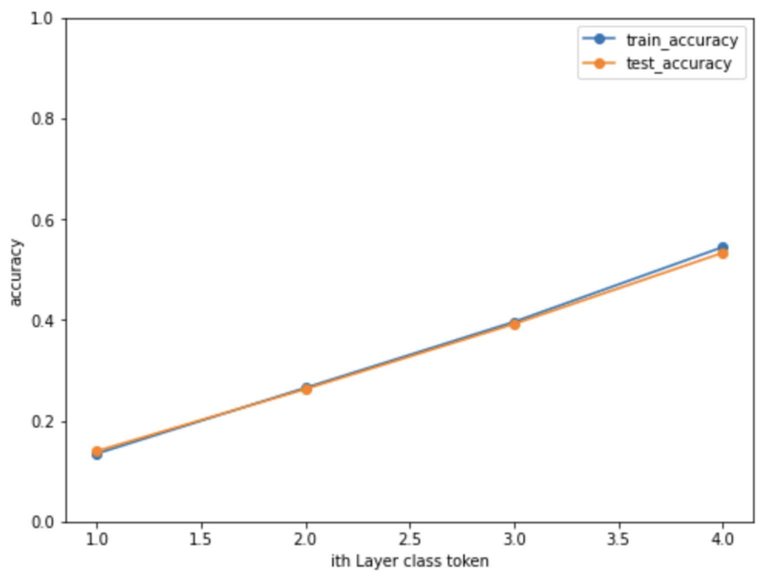




- The results were similar. The model was not showing any particular pattern as such.
- Increasing the number of attention heads definitely increases the complexity of model, but the size of the images we are testing is 32 X 32. So increasing number of attention heads doesn't make sense.
- Theoretically, using more attention heads can allow the model to capture a more diverse set of image features and relationships, leading to improved performance on tasks that require modelling complex interactions between image patches.
- But my **results do not depict that**, there are similar patterns.
- Maybe at significantly higher epochs and more data, the results might be more convincing.

E. EXPERIMENT BY CLASSIFYING WITH CLASS TOKEN AT DIFFERENT LAYER

- The class token serves as a way to inject **information about the image class** into the ViT model, and to allow the model to learn to make class-specific predictions.
- I have used 4 encoder layers to my Vision transformer as suggested in Question.
- I took out class token at each level and found accuracy on train and test data.
- As expected, as we go **up in the layers, the class token gets better results** and higher accuracies.



DEEP CONVOLUTIONAL GAN

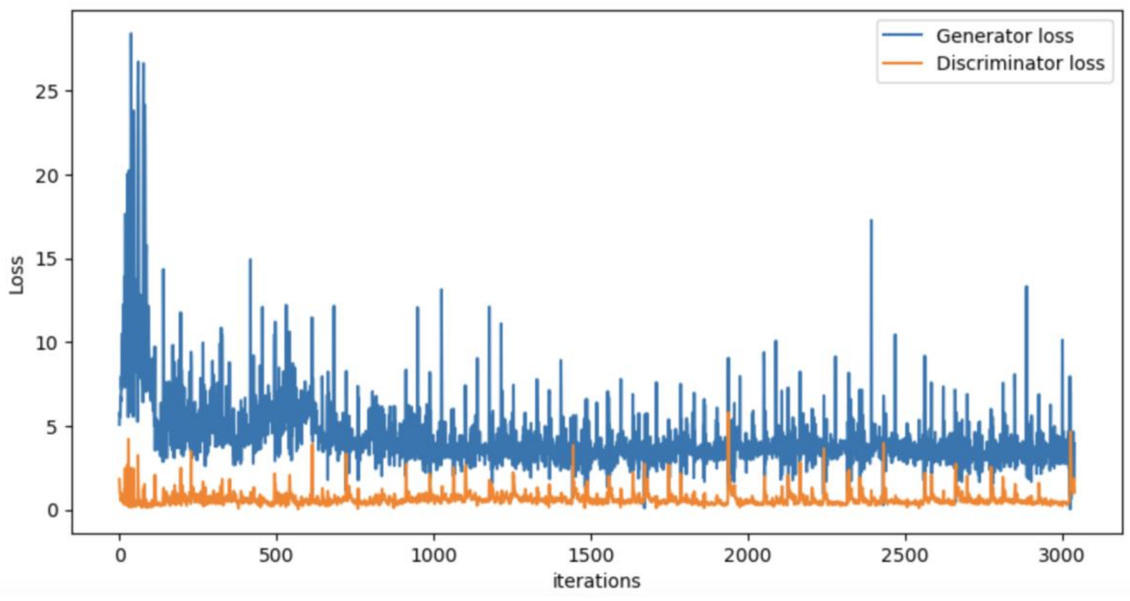
Puspak Priyabrata Sahu 21676 MTech CSA

A. **Implemented Basic DCGAN** as mentioned in the DCGAN paper using details of hyperparameters. Downloaded AFGQ-DOG dataset to generate the new dog photos.

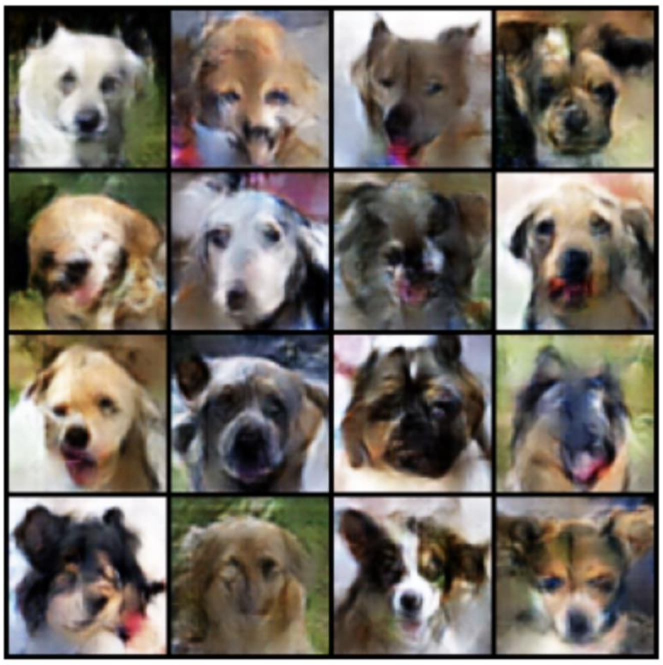
- These are the set of hyperparameters used for the implementation.

```
IMAGE_SIZE=64
TRAIN_DATA_ROOT="afhq/train"
TEST_DATA_ROOT="afhq/val"
BATCH_SIZE=128
NUM_CHANNELS=3
Z=100
NGPU=8
NUM_GEN_FM=64
NUM_DIS_FM=64
MAX_EPOCHS=80
LEARNING_RATE=0.0002
BETA_1=0.5
```

- The goal is to train the generator to generate **realistic images** that can fool the discriminator, while the discriminator is trained to distinguish between real and generated images.
- Both the generator and discriminator are updated in an **adversarial** manner. The discriminator is trained to minimize its loss, while the generator is trained to maximize the discriminator's loss.



- I have trained for 160 epochs in total.
- In the starting phase the Generator loss was much more and volatile but as the progress goes on it comes down closer to discriminator loss. As epoch increases, it becomes more stable and converges
- D(G(x)) and D(x) is converging to 0.5 after running sufficiently many epochs.
- Ideally, the training process converges when the discriminator is no longer able to distinguish between real and generated images, and the generator is able to generate images that are **indistinguishable from real images**.
- At this point, the discriminator's loss is at its minimum, and the generator's loss is at its maximum.
- Below are the randomly generated images after 160 epochs of training over AFHQ-Dog dataset.



At Epoch 160

B. PLAYING WITH THE HYPERPARAMETERS

❖ EPOCHS

- I have run the DCGAN for 80, 120 and 160 epochs and observed the quality of generated images.
- We can clearly see as the training progresses I.e., more epochs lead to **better convergence** of loss and hence **better-quality images.**
- Increasing the number of epochs in DCGAN training can lead to improved image quality, but it also comes with **longer training times,** a **higher risk of overfitting,** and more stable discriminator.

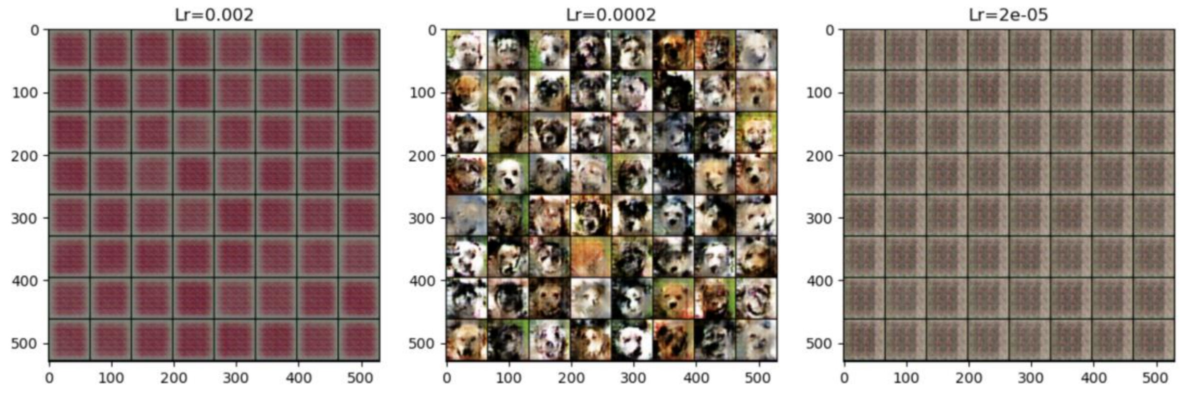
Epoch= 80

Epoch=120

Epoch=160

❖ LEARNING RATE

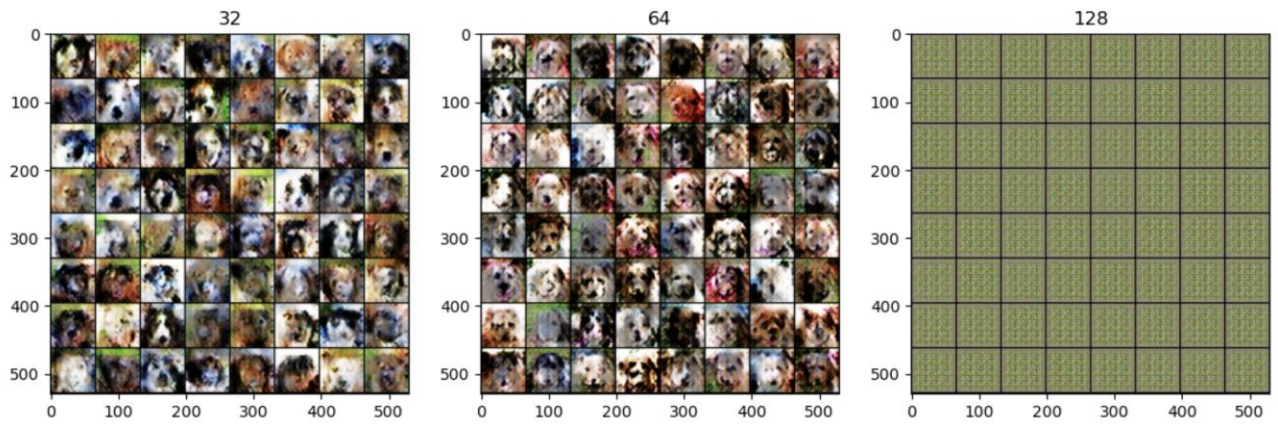
- Experimented with three learning rates (0.002, 0.0002, 0.00002) to see the results in the optimiser functioning and the quality of images.
- In the paper advised learning rate is 0.0002, and it is also mentioned that it is very **sensitive hyperparameter**.
- This should not be much played with.
- Our result confirms that learning rate should not be something to alter hoping for better results.



- As clearly seen, at different learning rates the results are significantly different.
- At 0.0002 it is perfect whereas with slight change in Learning rate the images are not forming this shows the sensitivity of this parameter.

❖ NUMBER OF GENERATOR AND DISCRIMINATOZR FEATURE MAPS

- I experimented with NGF with values 32,64 and 128.
- Basically, this hyper parameter controls the depth of each individual block in both discriminator and generator.
- So, increasing this increases the parameters significantly and similarly decreasing reduces the parameters. Ideally, we expect more parameters result in better results but not in this case.
- The paper DCGAN suggested 64 and our experiment convinces that 64 is optimal.
- The quality of images is better in 64 than in 32.

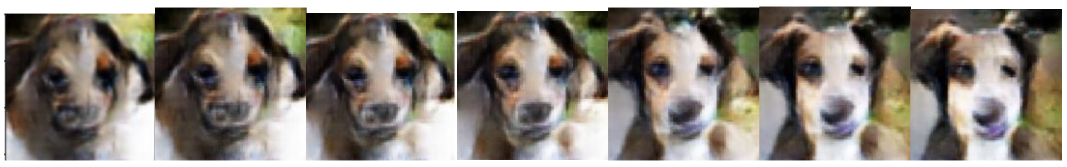


C. INTERPOLATION IN Z_SPACE

- Sampled two points in the latent space and generated their corresponding images using the generator network

```
for alpha in alpha_l:
    z_i=z1*alpha+z2*(1-alpha)
```

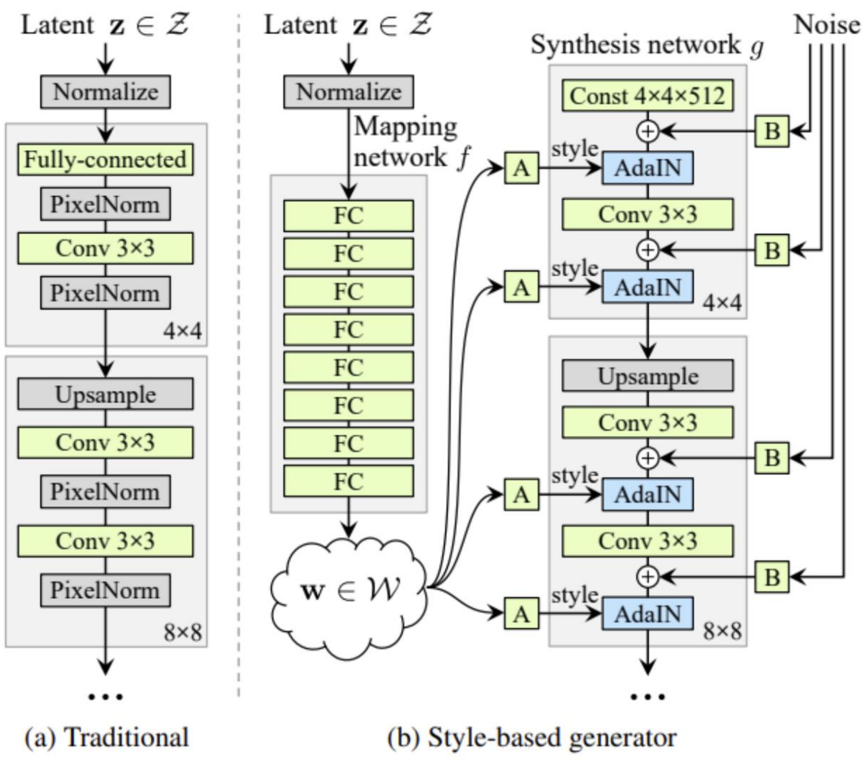
- By varying the value of α between 0 and 1, we can control the degree of interpolation and generate a range of intermediate images between the two endpoints.
- Interpolation in the z-space of DCGAN is a powerful technique for exploring the latent space of the generator network and generating novel images



- The results prove that the latent space is traversed properly as the middle image is brown when first and last images are white and black dog respectively.

D. ADAIN AND MAPPING LAYER

- Basic changes include MLP layers between Z and W space
- Adding of Noise and ADAIN Normalisation.


Latent $\mathbf{z} \in \mathcal{Z}$
Normalize
Fully-connected
PixelNorm
Conv 3×3
PixelNorm
4×4
Upsample
Conv 3×3
PixelNorm
Conv 3×3
PixelNorm
8×8
(a) Traditional
Latent $\mathbf{z} \in \mathcal{Z}$
Normalize
Mapping network f
FC
$\mathbf{w} \in \mathcal{W}$
Synthesis network g
Noise
Const 4×4×512
A
B
style
AdaIN
Conv 3×3
Upsample
4×4
8×8
(b) Style-based generator