APPLIED A.I. SOLUTIONS DEVELOPMENT PROGRAM

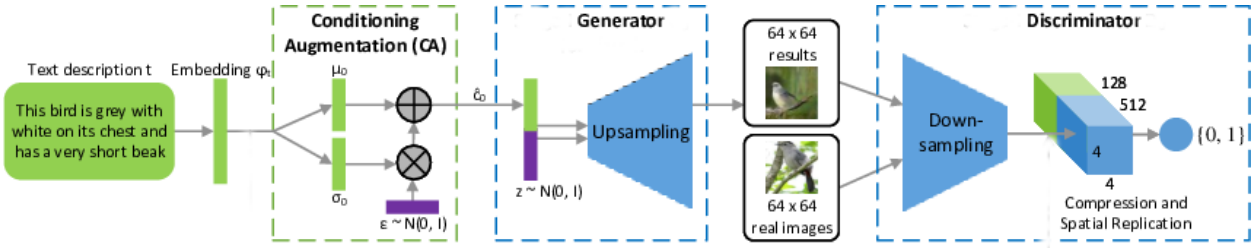
ADVANCED APPLIED MATHEMATICAL CONCEPTS FOR DEEP LEARNING

Final Project
Text to Image GAN Model from scratch

Group Members:
Qianfan Cheng: (101415827)
Himani: (101445554)

1 Objective

This project aims to design and implement a text to image generative adversarial neural network from scratch. We used sentence embedding from pretrain-model SentenceTransformer and the other layers were built using tensorflow framework. The architecture of the model is as shown following diagram:



2 Dataset

Dataset was download from Caltech-UCSD Birds: https://www.vision.caltech.edu/datasets/cub_200_2011/ which contains birds with captions in different classes. However, in this project we only need the images and captions. Images and captions were loaded into tensorflow using image_dataset_from_directory and text_dataset_from_directory respectively.

3 Model Summary

3.1 Generator Model

Generator is responsible for generating the image that could mislead discriminator.

Layer (type)	Output Shape	Param #	Connected to
EmbeddingInputLayer (InputLayer)	(None, 384)	0	–
ConditionalLayer (ConditioningAugme…	(None, 128)	98,560	EmbeddingInputLa…
NoiseInputLayer (InputLayer)	(None, 100)	0	–
ConcateLayer (Concatenate)	(None, 228)	0	ConditionalLayer… NoiseInputLayer[…
FCLayer (Dense)	(None, 16384)	3,735,552	ConcateLayer[0][…
ReshapeLayer (Reshape)	(None, 4, 4, 1024)	0	FCLayer[0][0]
DecConv512 (DecoderLayer)	(None, 8, 8, 512)	8,390,656	ReshapeLayer[0][…
DecConv256 (DecoderLayer)	(None, 16, 16, 256)	2,098,176	DecConv512[0][0]
DecConv128 (DecoderLayer)	(None, 32, 32, 128)	524,800	DecConv256[0][0]
DecConv64 (DecoderLayer)	(None, 64, 64, 64)	131,328	DecConv128[0][0]
ImageOutputLayer (Conv2D)	(None, 64, 64, 3)	1,728	DecConv64[0][0]

Total params: 14,980,800 (57.15 MB)

Trainable params: 14,978,880 (57.14 MB)

Non-trainable params: 1,920 (7.50 KB)

3.2 Discriminator Model

Discriminator is responsible for classifying real images and generated images.

Layer (type)	Output Shape	Param #	Connected to
ImageInputlayer (InputLayer)	(None, 64, 64, 3)	0	–
AugmentLayer (AugmentLayer)	(None, 64, 64, 3)	0	ImageInputlayer[…
EncConv64 (EncoderLayer)	(None, 32, 32, 64)	3,328	AugmentLayer[0][…
EncConv128 (EncoderLayer)	(None, 16, 16, 128)	131,584	EncConv64[0][0]
EmbeddingInputLayer (InputLayer)	(None, 384)	0	–
EncConv256 (EncoderLayer)	(None, 8, 8, 256)	525,312	EncConv128[0][0]
ConditionalLayer (ConditioningAugme…	(None, 128)	98,560	EmbeddingInputLa…
EncConv512 (EncoderLayer)	(None, 4, 4, 512)	2,099,200	EncConv256[0][0]
reshape_layer (ReshapeLayer)	(None, 4, 4, 128)	0	ConditionalLayer…
ConcateLayer (Concatenate)	(None, 4, 4, 640)	0	EncConv512[0][0], reshape_layer[0]…
conv2d_4 (Conv2D)	(None, 4, 4, 512)	327,680	ConcateLayer[0][…
batch_normalizatio… (BatchNormalizatio…	(None, 4, 4, 512)	2,048	conv2d_4[0][0]
leaky_re_lu_7 (LeakyReLU)	(None, 4, 4, 512)	0	batch_normalizat…
FlattenLayer (Flatten)	(None, 8192)	0	leaky_re_lu_7[0]…
DropoutLayer (Dropout)	(None, 8192)	0	FlattenLayer[0][…
ClsOutputLayer (Dense)	(None, 1)	8,193	DropoutLayer[0][…

Total params: 3,195,905 (12.19 MB)

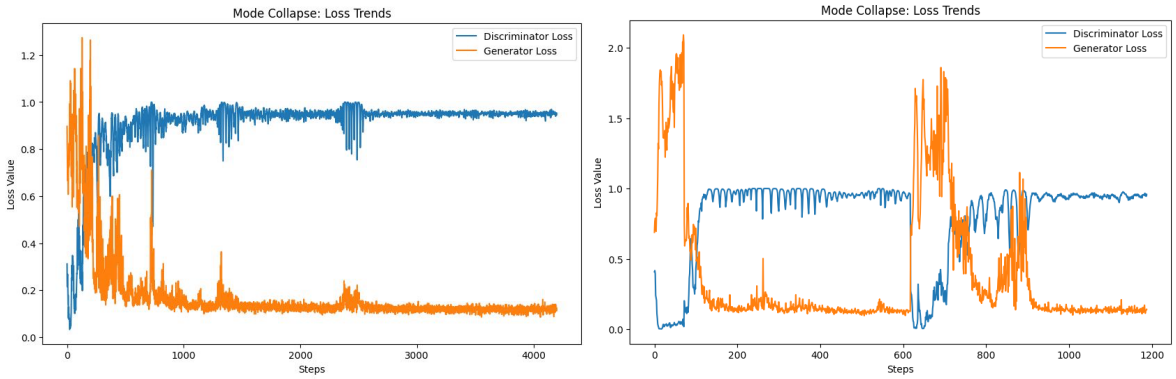
Trainable params: 3,192,961 (12.18 MB)

Non-trainable params: 2,944 (11.50 KB)

Discriminator and generator models are trained separately with different loss functions and metrics respectively. We deliberately set learning rate of Discriminator lower than generator, so that discriminator would not overwhelm the performance of generator quickly.

4 Problems

After completing implementation of the model and running the model, we were facing convergence failure in GAN. It's either generator overwhelmed discriminator, or discriminator overwhelmed generator. We tried to adjust learning rate by decreasing the overwhelming party, increasing dropout rate or add dropout layer in model to slow down learning of overwhelming party. We also tried to make the overwhelmed model deeper, however, as it's going to take much time to adjust all layers in both generator and discriminator model, this approach was not implemented in this project. We didn't make our model converge eventually due to the cost of GPU and time.



5 Conclusion

To build a text to image generative adversarial network model from scratch with tensorflow framework is full of fun and pain. It took us almost 4-5 days to build the whole network model architecture in tensorflow, especially the embedding part (tried to build it with tensorflow embedding layer and failed, because its gonna take too much time to reconstruct a sentence embedding), concatenate part (concatenate embeddings with noise and embedding with image) and calculations layer after layer.