MSc in Artificial Intelligence

Deep Learning

Text to Image Synthesis using GANs

Delikonstantis Dimitrios

MTN 2007

Bochalis Christodoulos

MTN 2019

July 5, 2021

Demo

text-to-image-synthesis

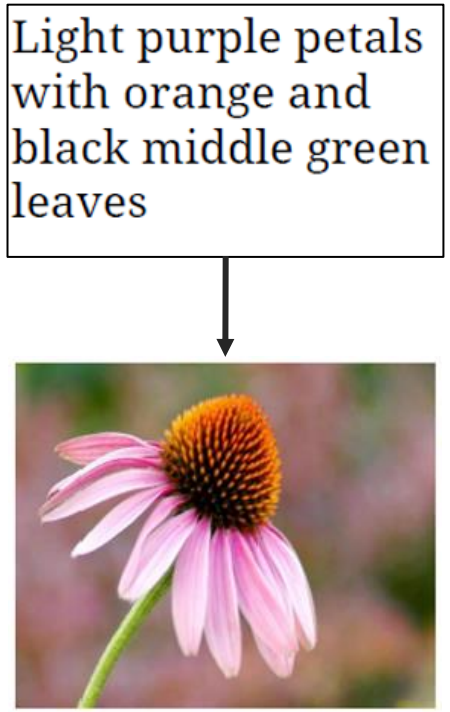
Problem Definition

Given

- Text descriptions of images written by a human

Generate

- Realistic and previous unseen images



Our approach is based on solving the following tasks:

- learn a **text feature representation** that captures the important **visual details.**
- **generate a realistic image** using the text representation

Fine-Grained Visual Descriptions

Multimodal Learning Method

Method presented by Scott Reed et al. in 2016 [1]

Given $S = \{(v_n, t_n, y_n), n = 1, \dots, N\}$

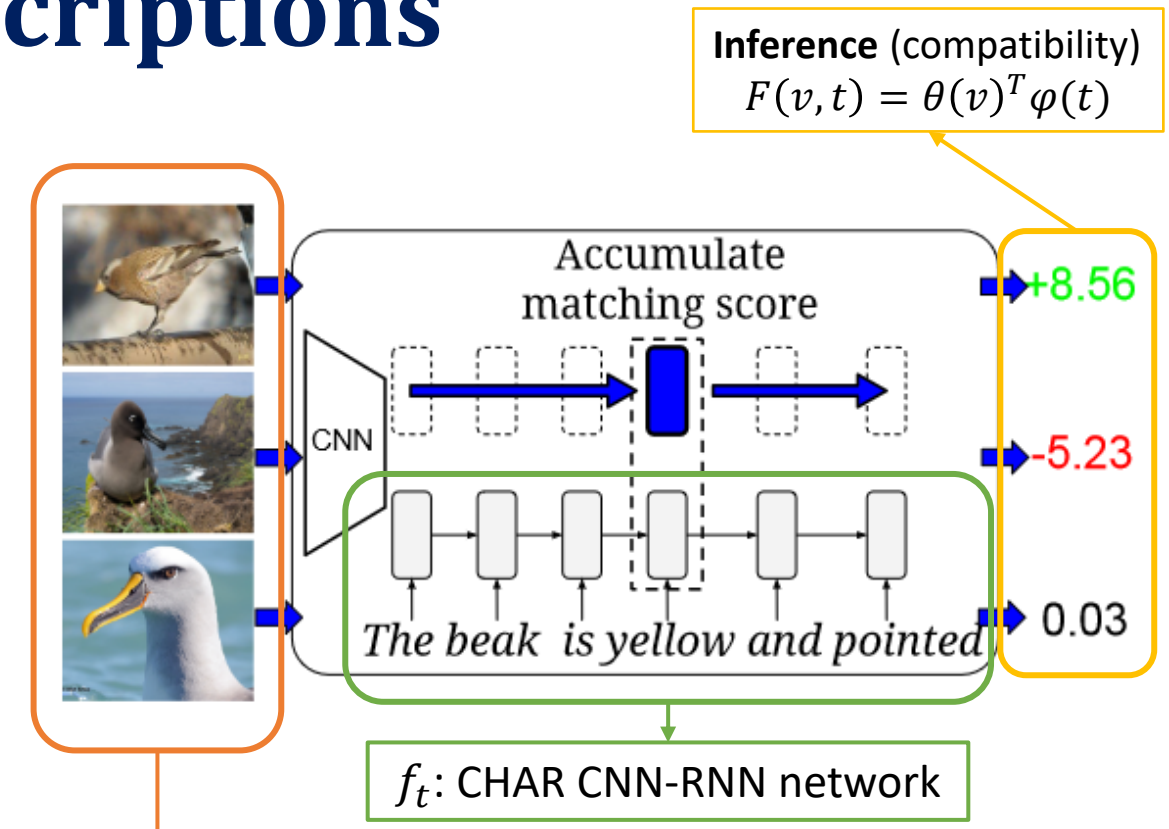
where:

- v_n visual information $v \in V$
- t_n textual information $t \in T$
- y_n class labels $y \in Y$

Learn functions $f_v: V \rightarrow Y$ and $f_t: T \rightarrow Y$

minimize $\frac{1}{N}\sum_{i=1}^{N} \Delta(y_n, f_v(v_n)) + \Delta(y_n, f_t(t_n))$

symmetric | predict by conditioning on both images and text



CHAR CNN-RNN

CHAR CNN-RNN architecture

- trains on (image, captions) combinations
- outputs an embedding vector that captures their underlying relationship

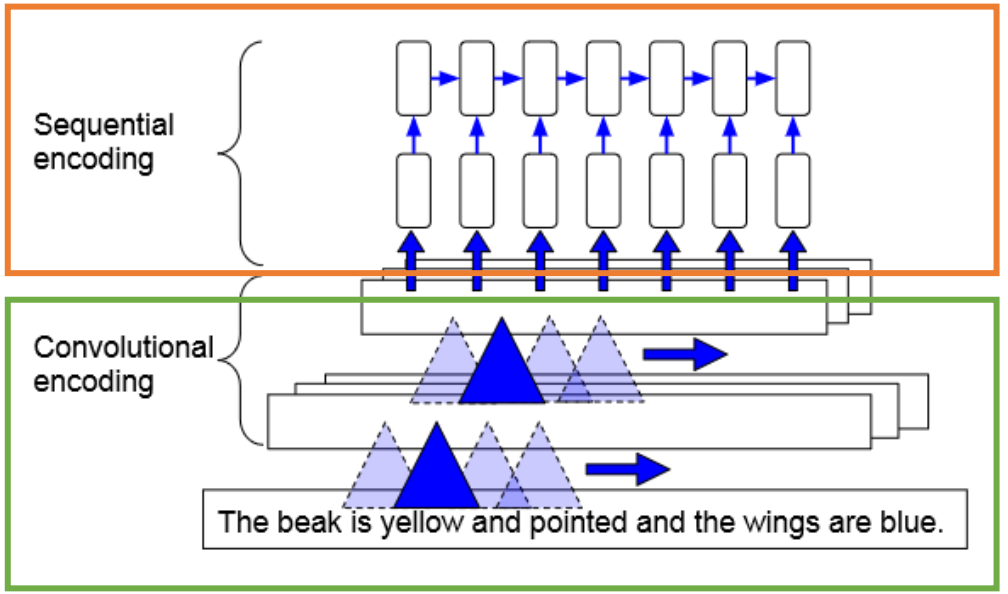
- The input of the network is an $A \times L$ one hot encoding feature vector where $\boldsymbol{A}$ the **alphabet size** and $\boldsymbol{L}$ the **max sentence length**

  In our case 70×200

```
# The alphabet used
alphabet = "abcdefghijklmnopqrstuvwxyz0123456789 \
            -,;.!?:'\"/\\|_@#$%^&*~`+-=<>()[]{} "
```

The model has 2.994.560 trainable parameters

RNN layer
adds temporal dependency
along the input text sentence

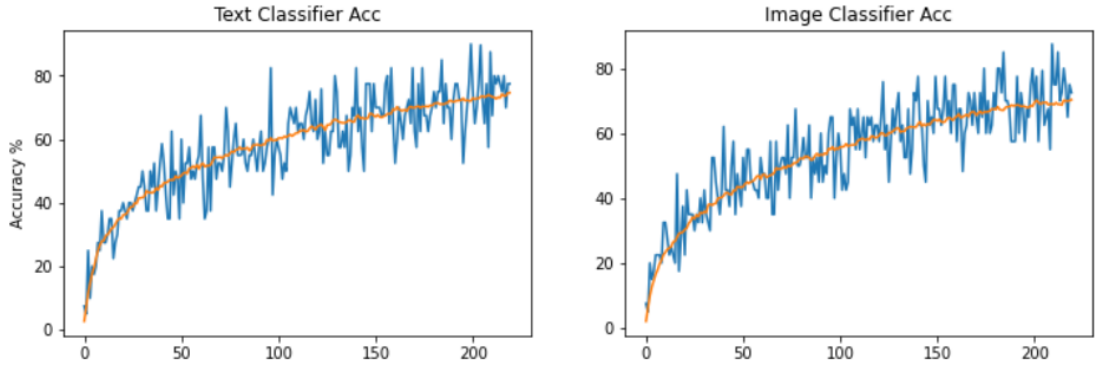


CNN hidden layer
output split along the time dimension
(ours: **RNN Steps = 8**)

CHAR CNN-RNN | Training & Evaluation

Preprocessed flowers dataset* with precomputed the image feature vectors

The original [Oxford Flowers-102 (Flowers) dataset.



Training

8.189 flower images

- 102 different categories
- up to 10 captions for each image
- 82 training + validation and 20 test classes

100 epochs, ~3 hours, used learning rate decay

Evaluation

Zero shot image classification and retrieval tasks

- Text encodings from test captions and averaged them per-class.
- ranking by compatibility function

```
----- RETRIEVAL -----
  mAP:
0.7500: mAP@1
0.6900: mAP@5
0.6850: mAP@10
0.5730: mAP@50
---------------------

--- CLASSIFICATION --
Average top-1 accuracy: 0.6372
---------------------
```

*authors github

Generative Adversarial Networks

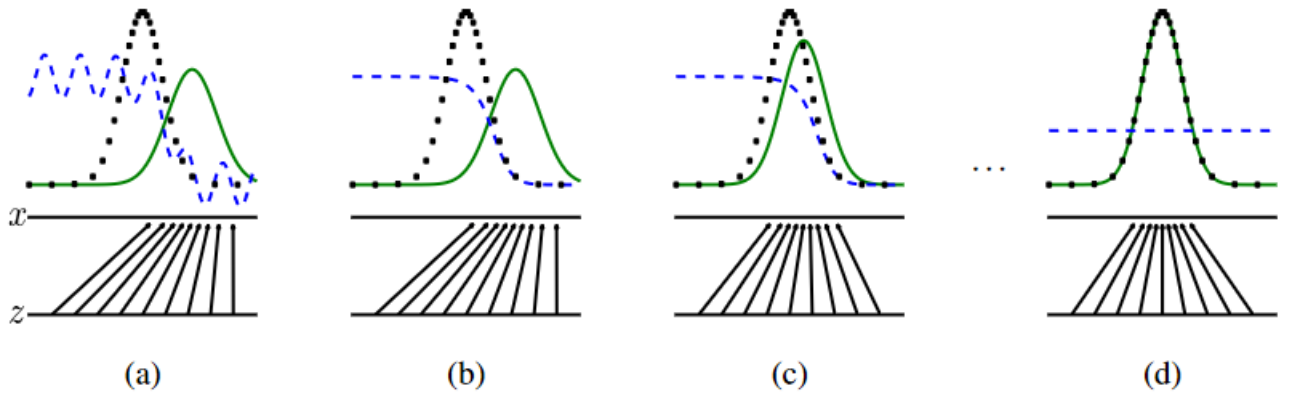
GANs and DCGAN framework

Capture the training data's distribution and generate new data from that same distribution

Loss function

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[logD(x)] + \mathbb{E}_{x\sim p_z(z)}[\log\left(1 - D\left(G(z)\right)\right)]$$

Global optimum is $\boldsymbol{p_g = p_{data}}$ and $\boldsymbol{D(x) = 50\%}$ (random guess)



Text to Image Synthesis

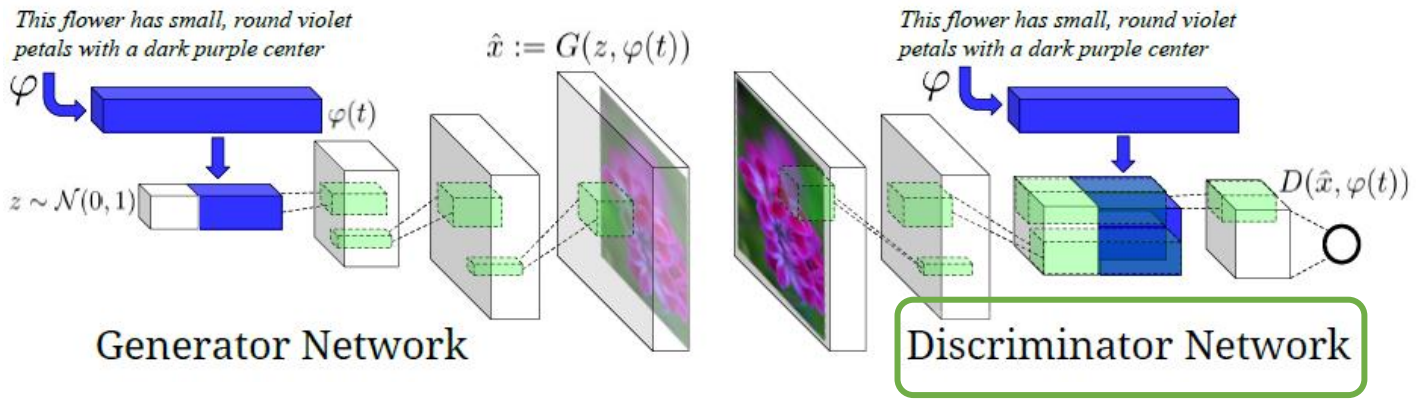
Network architecture consists of a DCGAN conditioned on text features encoded by a Char CNN-RNN network. Based on the method Scott Reed et al. in 2016 [2]

Discriminator

- strided convolution layers
- batch norm layers
- LeakyReLU activations

Generator

- convolutional-transpose layers
- batch norm layers
- ReLU activations



- real images with matching text
- generated images with arbitrary text
- real images with fake text

GAN-CLS

Discriminator does not recognize if real images match the text embeddings

The GAN-CLS adds a third type of input to the discriminator:

- real images with fake text

We observe that GAN generalizes slower with CLS mode

Configuration

- Initialize all model weights from a normal distribution:
  - mean=0, stdev=0.02, as proposed by DCGAN paper
- We train and test on **class-disjoint sets**, to test **generalization ability**
- **Fretchet Inception Distance** (FID) distance between the Inception-v3 activation distributions for real and generated images

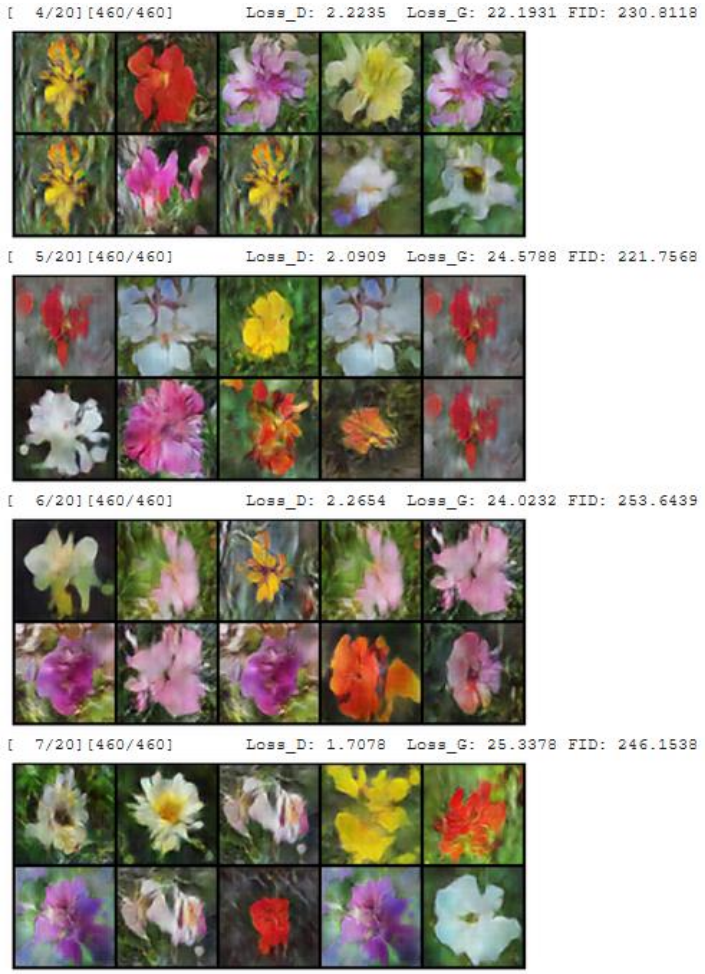
Managed to train for 40 epochs ~2 hours per 10 epochs

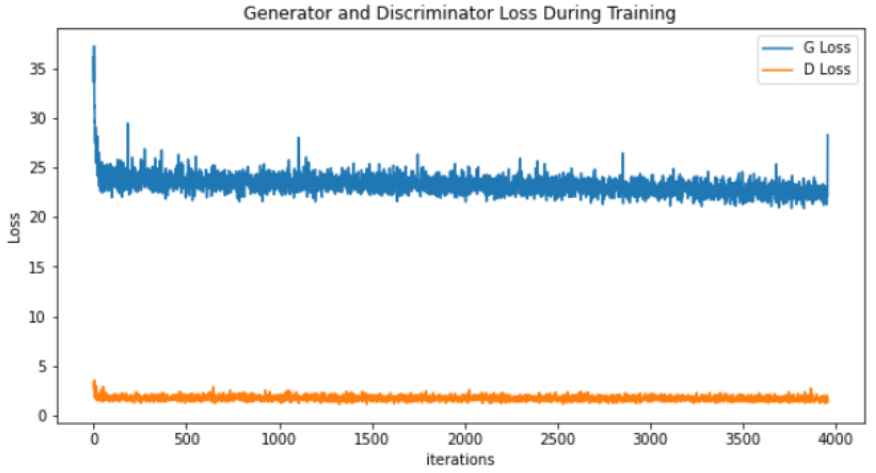
One sided label smoothing
- penalize discriminator prediction of real labels to avoid overconfidence

Generator Loss Function
- **Feature Matching Loss**
  Match also features, extracted from intermediate layer
- **L1 distance between generated and real images**
  Links the embedding feature vector to pixel values

```
[8]  # Size of z latent vector (i.e. size of generator input)
     noise_dim = 100

     # Batch size during training
     batch_size = 64

     # Number of workers for dataloader
     num_workers = 2

     # Learning rate for optimizers
     lr = 0.0002

     # Number of training epochs
     num_epochs = 20

     # Beta1 hyperparam for Adam optimizers
     beta1 = 0.5

     # L1 loss coefficient in the generator loss function
     l1_coef = 50

     # Feature matching coefficient in the generator loss function
     l2_coef = 100

     # True if GAN-CLS is used
     cls = False
```

Results (1/2) – Training and Evaluation





Results (2/2) – Real and generated images based on embedding

the flower has a white petals with many stamen around the green pollen tube

this pink flower has many pointed, oval petals clustered around the center.

this flower has pink pistil and pink petals as its main features


```
# captionText = 'Anything but a flower please'
captionText = 'Light purple petal swith orange and black middle green leaves'
```

Light purple petal swith orange and black middle green leaves

Networks Architectures

CHAR CNN-RNN

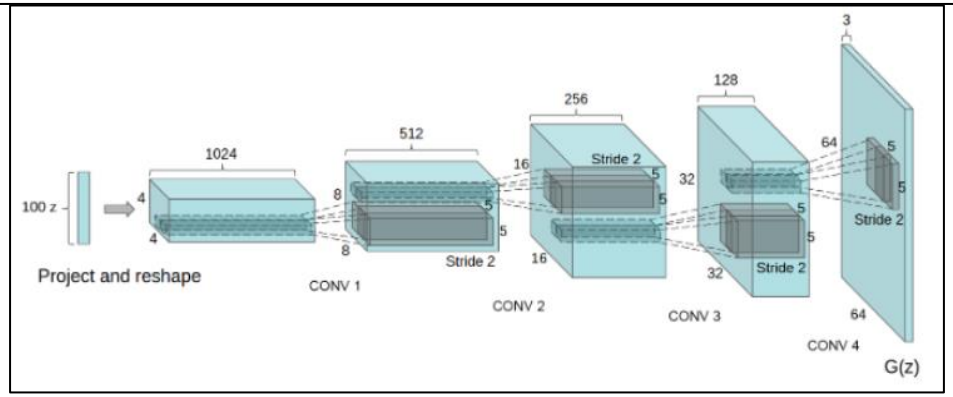
```
char_cnn_rnn(
  (conv1): Conv1d(70, 384, kernel_size=(4,), stride=(1,))
  (threshold1): Threshold(threshold=1e-06, value=0)
  (maxpool1): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv1d(384, 512, kernel_size=(4,), stride=(1,))
  (threshold2): Threshold(threshold=1e-06, value=0)
  (maxpool2): MaxPool1d(kernel_size=3, stride=3, padding=0, dilation=1, ceil_mode=False)
  (conv3): Conv1d(512, 512, kernel_size=(4,), stride=(1,))
  (threshold3): Threshold(threshold=1e-06, value=0)
  (maxpool3): MaxPool1d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  (rnn): fixed_rnn(
    (i2h): Linear(in_features=512, out_features=512, bias=True)
    (h2h): Linear(in_features=512, out_features=512, bias=True)
  )
  (emb_proj): Linear(in_features=512, out_features=1024, bias=True)
)
```

Discriminator - DCGAN

```
discriminator(
  (netD_1): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (projector): Concat_embed(
    (projection): Sequential(
      (0): Linear(in_features=1024, out_features=128, bias=True)
      (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): LeakyReLU(negative_slope=0.2, inplace=True)
    )
  )
  (netD_2): Sequential(
    (0): Conv2d(640, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
)
```

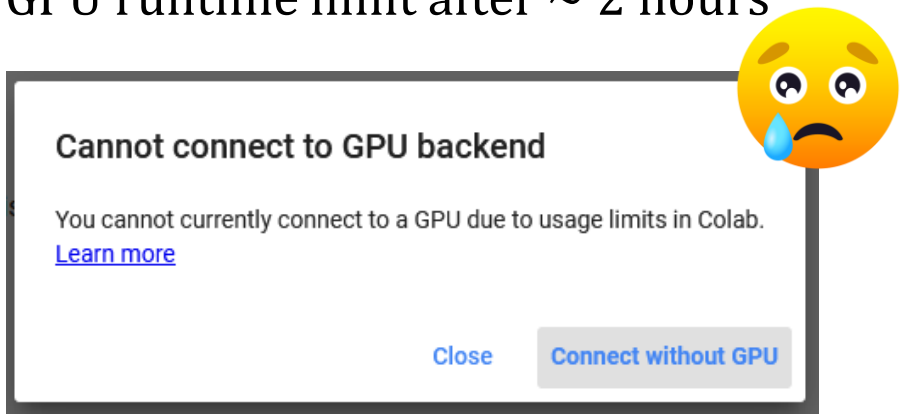
Generator - DCGAN

```
generator(
  (projection): Sequential(
    (0): Linear(in_features=1024, out_features=128, bias=True)
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): LeakyReLU(negative_slope=0.2, inplace=True)
  )
  (netG): Sequential(
    (0): ConvTranspose2d(228, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```



Issues Faced

- GPU runtime limit after ~ 2 hours



- COCO dataset too big to handle on Collab and extract text embeddings
  - ~90.000 images from 80 classes
- Birds dataset slower generalization
  - ~6000 images from 200 classes
- Moved datasets to collab leads to faster training

State of the Art

DALL·E[1] is a 12-billion parameter version of GPT-3 trained to generate images from text descriptions, using a dataset of text–image pairs. We've found that it has a diverse set of capabilities, including creating anthropomorphized versions of animals and objects, combining unrelated concepts in plausible ways, rendering text, and applying transformations to existing images.

OpenAI

TEXT PROMPT an armchair in the shape of an avocado. . . .

AI-GENERATED IMAGES

Edit prompt or view more images

TEXT & IMAGE PROMPT the exact same cat on the top as a sketch on the bottom

AI-GENERATED IMAGES

Edit prompt or view more images

TEXT PROMPT a store front that has the word 'openai' written on it. . . .

AI-GENERATED IMAGES

Edit prompt or view more images

TEXT PROMPT an illustration of a baby daikon radish in a tutu walking a dog

AI-GENERATED IMAGES

Edit prompt or view more images

Thank you

References

[1] Reed, Scott, et al. "Learning deep representations of fine-grained visual descriptions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. link

[2] Reed, Scott, et al. "Generative adversarial text to image synthesis." International Conference on Machine Learning. PMLR, 2016. link