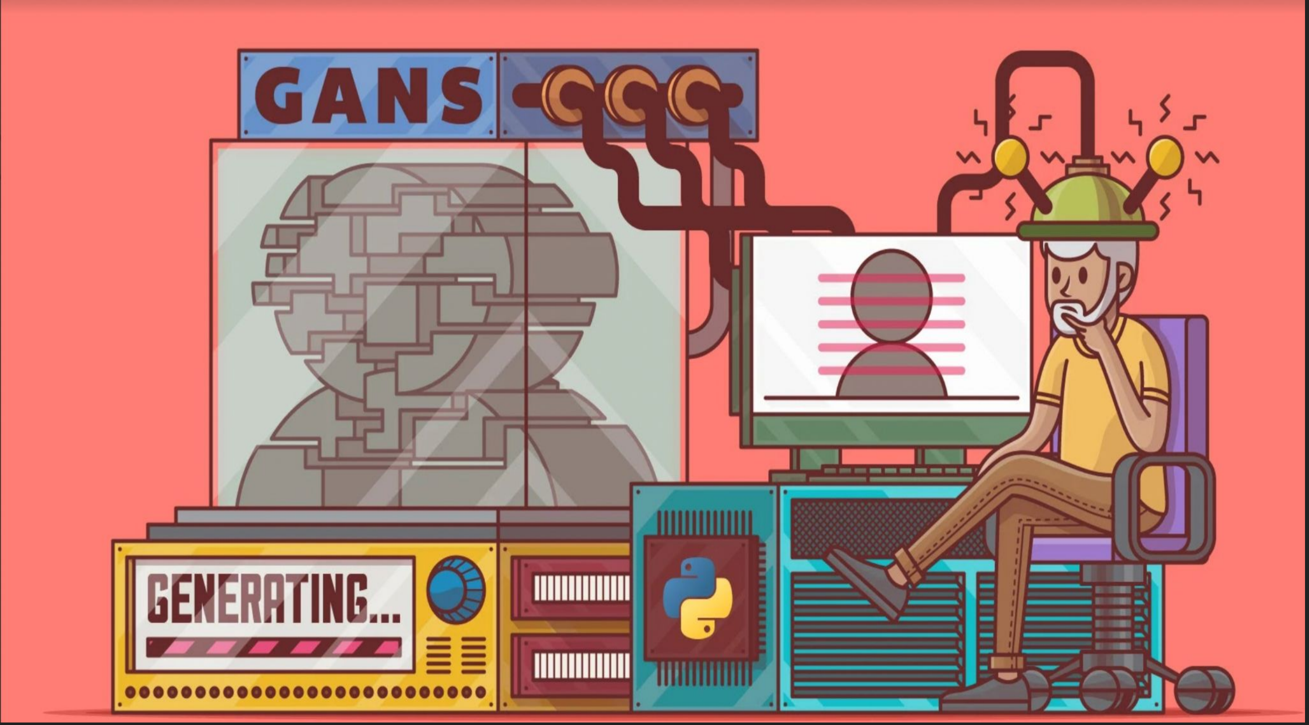


WEB IMPLEMENTATION OF GENERATIVE ADVERSARIAL NETWORKS

Internal Guide
Sreedhar Potla

Co-Ordinator
Dr. Sunil Bhutada

Done by
Dheeraj Perumandla (17311A1232)
Likhitha Alle (17311A1228)
Nikitha Katakam (17311A1215)

Abstract:

This Generative Adversarial Network is designed for attention-driven, multi-stage refinement for fine-grained text-to-image (Bird Image) generation based on the captions given by the user.

Automatically generating images according to natural language descriptions is a fundamental problem in many applications, such as art generation and computer-aided design. It also drives research progress in multimodal learning and inference across vision and language, which is one of the most active research areas in recent years.

Introduction:

- Automatically generating images according to natural language descriptions is a fundamental problem in many applications, such as art generation and computer-aided design. It also drives research progress in multimodal learning and inference across vision and language, which is one of the most active research areas in recent years.

- In many fields, like computer vision and natural language processing, the need of data is very crucial, but every time the data might not be in large size.

- In this scenario, GANs come into play. They generate data.

Specifications:

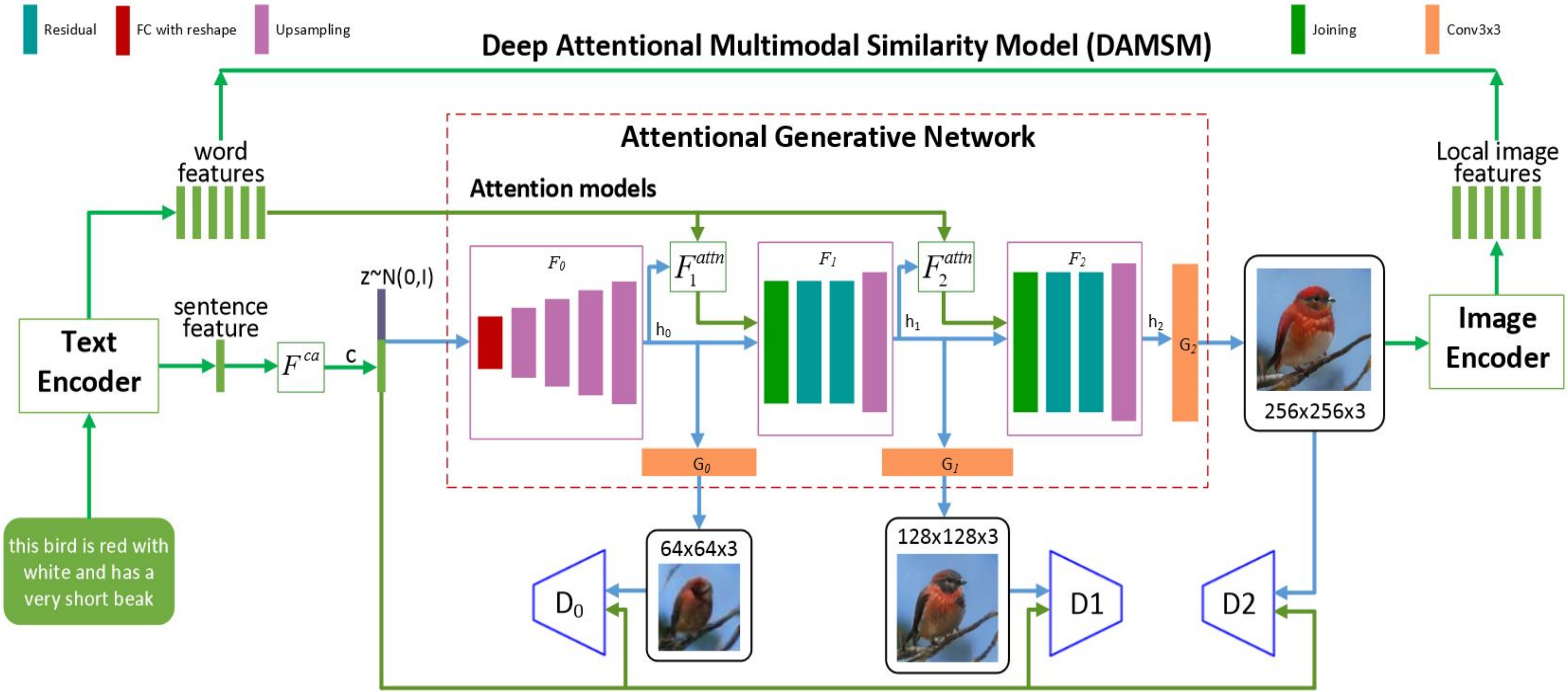
Software: 1. React JS

2. Python 3.x

Hardware: 1. Nvidia GPU (>=4GB) (Compulsory as the program needs CUDA)

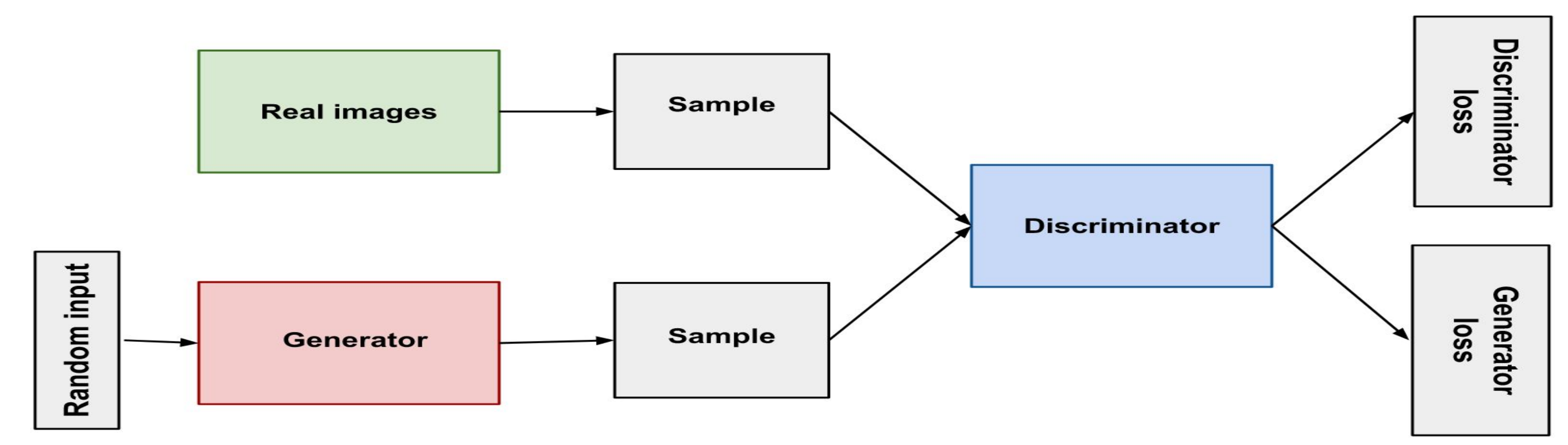
2. RAM >= 8GB

Architectural design:



GENERATOR:

- The generator part of a GAN learns to create fake data by incorporating feedback from the discriminator. It learns to make the discriminator classify its output as real.



DISCRIMINATOR

- The discriminator in a GAN is simply a classifier. It tries to distinguish real data from the data created by the generator. It could use any network architecture appropriate to the type of data it's classifying.

- The discriminator’s training data comes from two sources:
  - Real data instances, such as real pictures of people. The discriminator uses these instances as positive examples during training.
  - Fake data instances created by the generator. The discriminator uses these instances as negative examples during training.

TEXT EMBEDDER(SKIP-THOUGHTS):

- The Skip-Thoughts model is a sentence encoder. It learns to encode input sentences into a fixed-dimensional vector representation.

- A trained Skip-Thoughts model will encode similar sentences nearby each other in the embedding vector space.

React JS:

- React is an open-source JavaScript library for building user interfaces or UI components.

- React is only concerned with rendering data to the DOM, and so creating React applications usually requires the use of additional libraries for state management and routing.

Modules:

Front-End Modules:

1. Welcome Page
2. Input Page
3. Output Page

Datasets: Caltech-UCSD Birds-200-2011, Oxford-102

Back-End Modules:

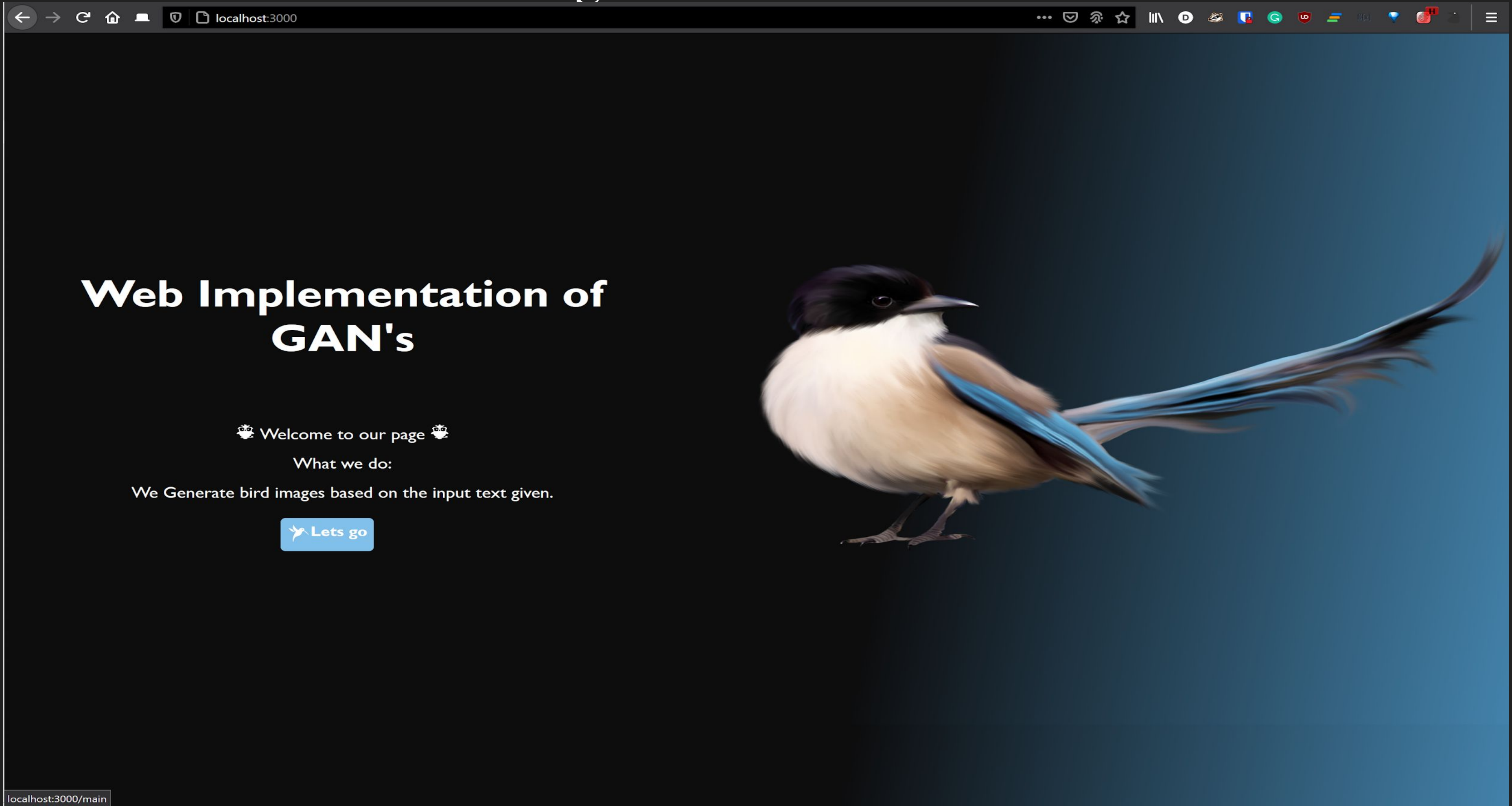
1. Main.py (for backend running)
2. Datasets.py (for accessing dataset and preprocessing)
3. Training.py (for training the model)
4. Model.py (Accessing the model)
5. Pretrain.py (to used already saved models)

Implementation:

1. Use command “npm start” to start the front end server

2. Run app.py to start the backend server.

Front-end look: 1. Home Page



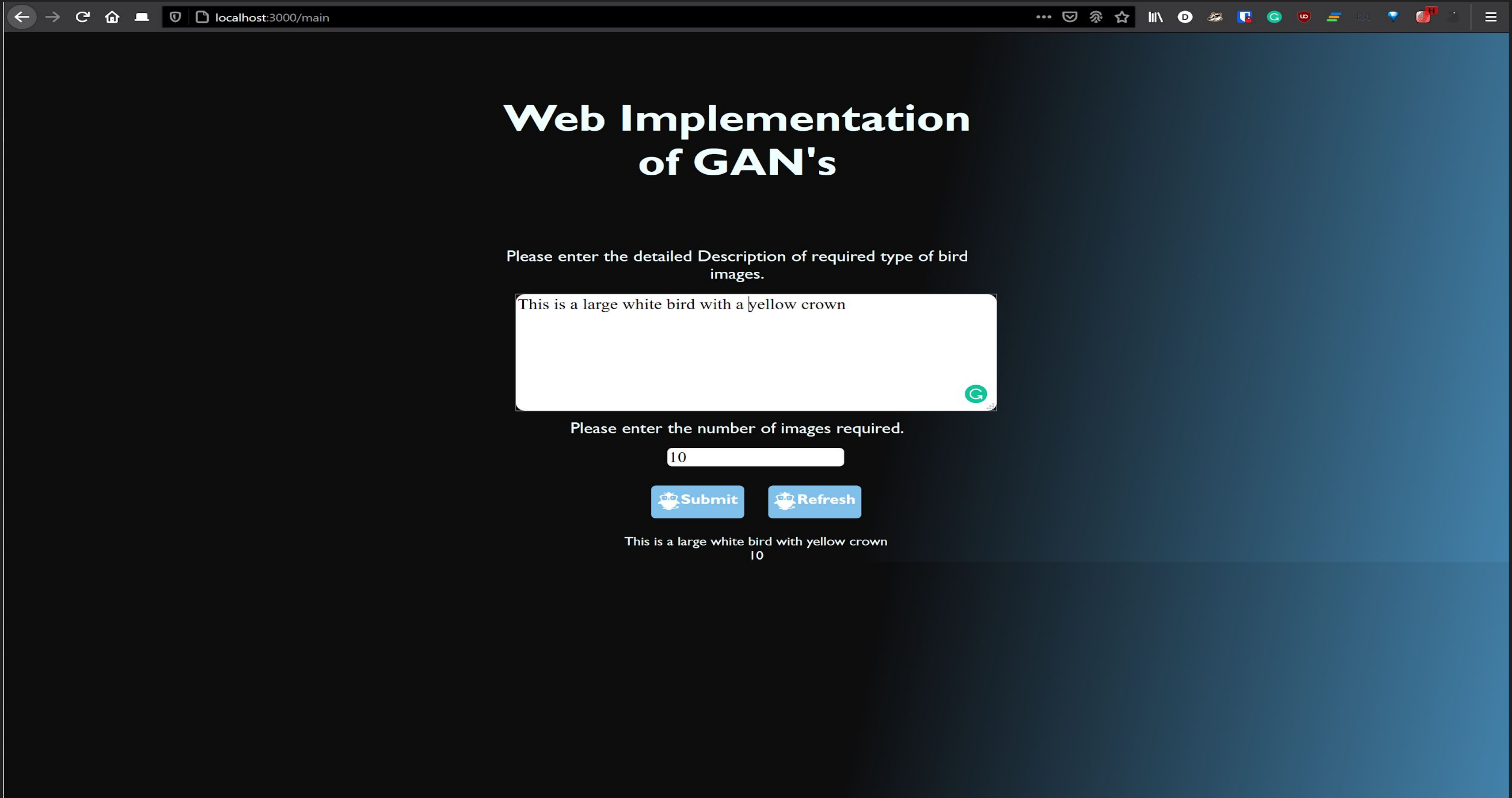
Web Implementation of GAN's

Welcome to our page

What we do:

We Generate bird images based on the input text given.

2. Input Page:



Web Implementation of GAN's

Please enter the detailed Description of required type of bird images.

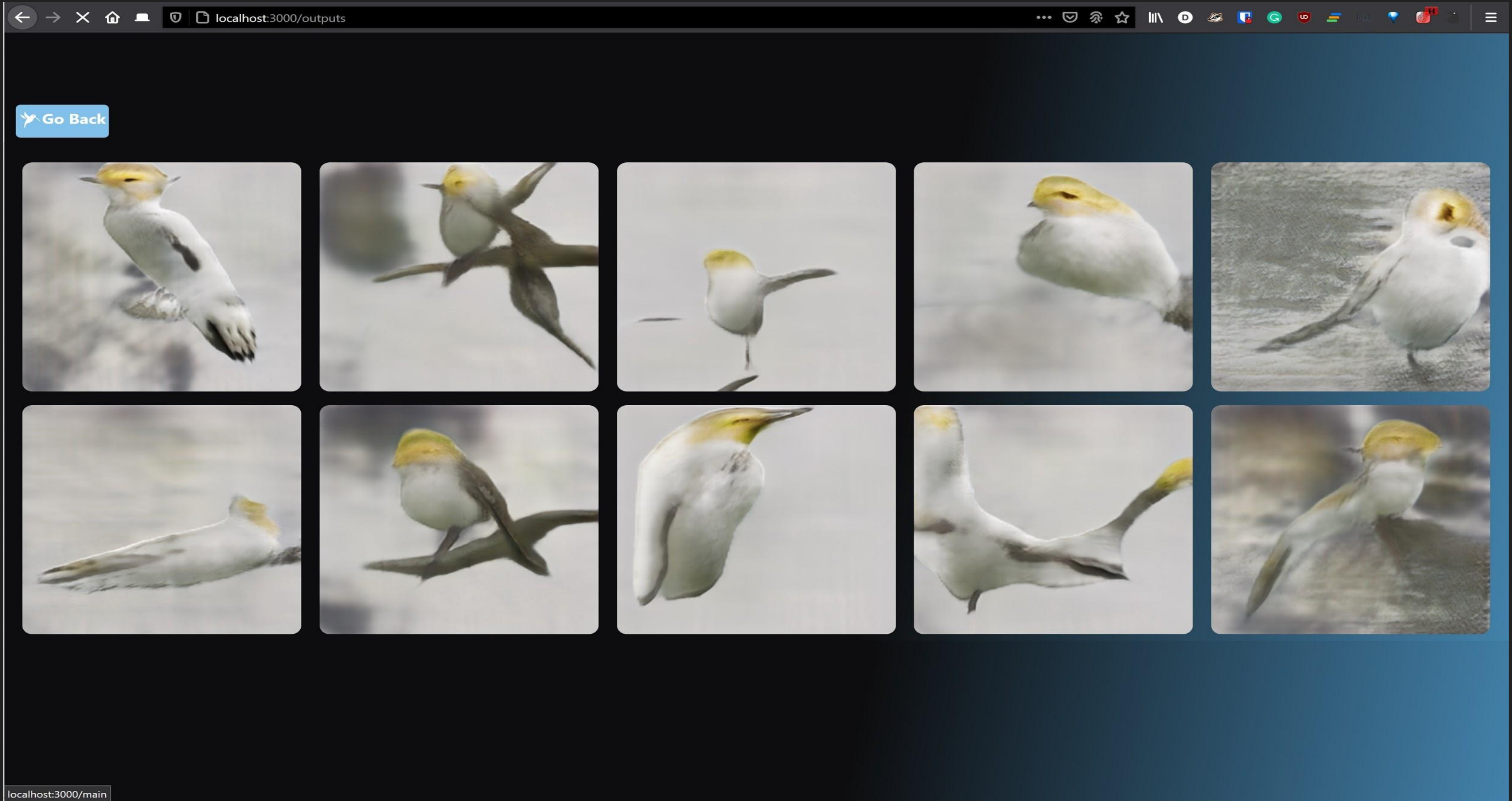
This is a large white bird with a yellow crown

Please enter the number of images required.

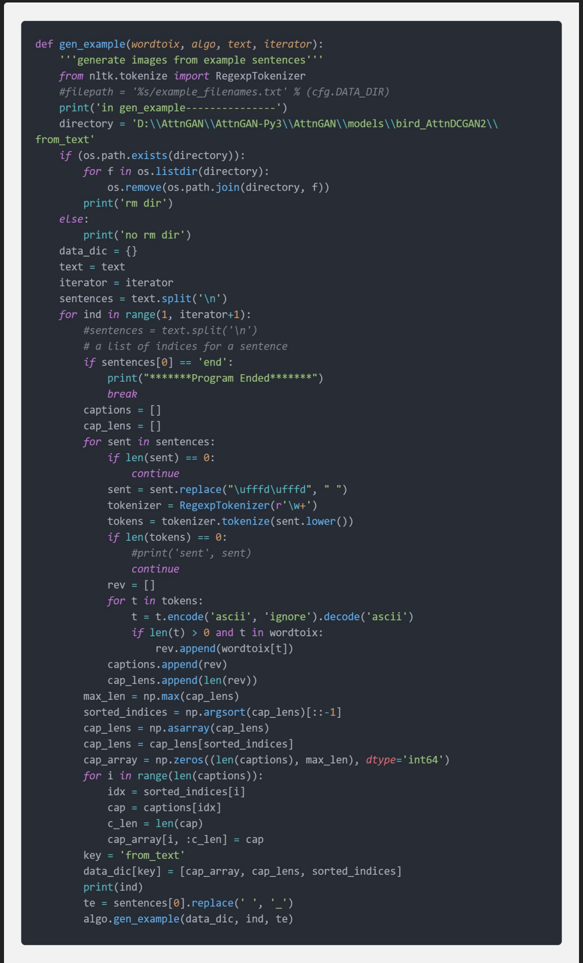
10

This is a large white bird with yellow crown
10

3. Output Page:

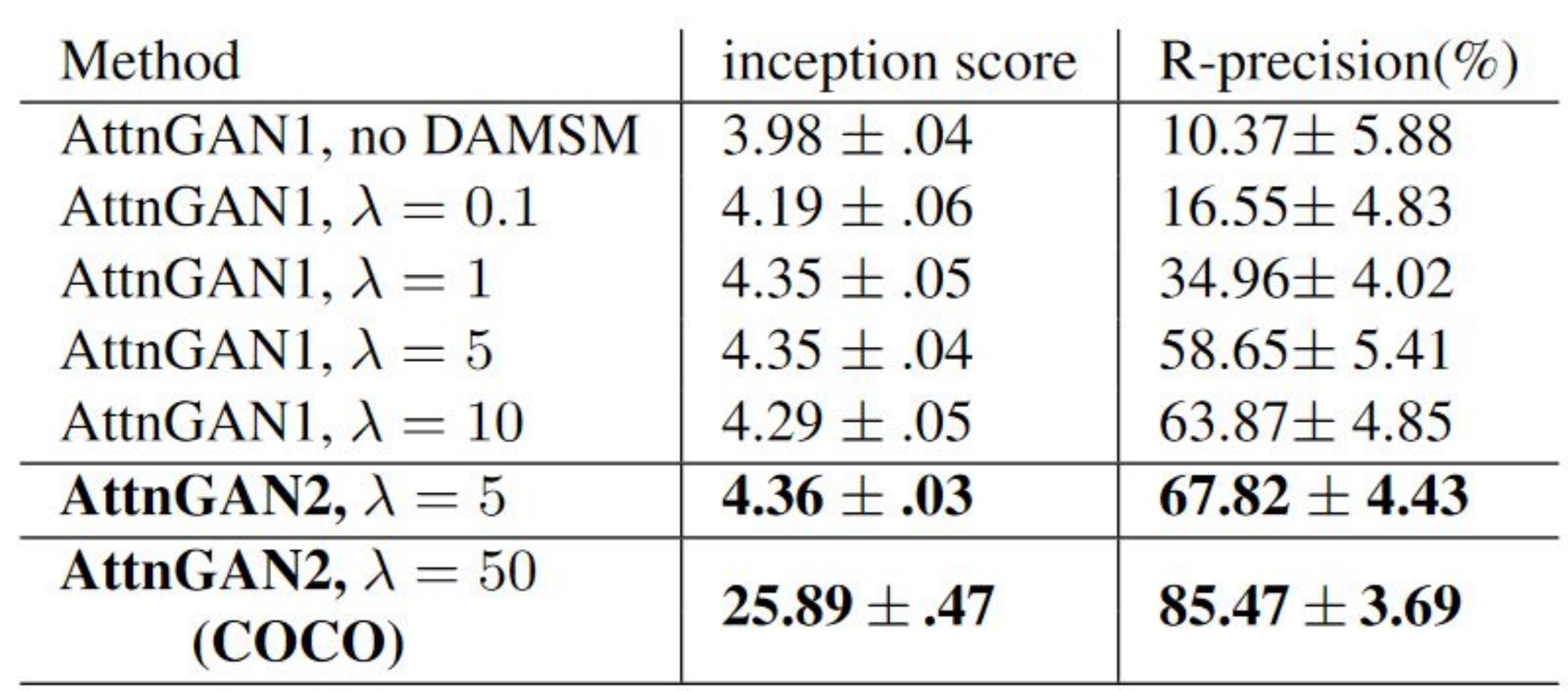


Sample code:

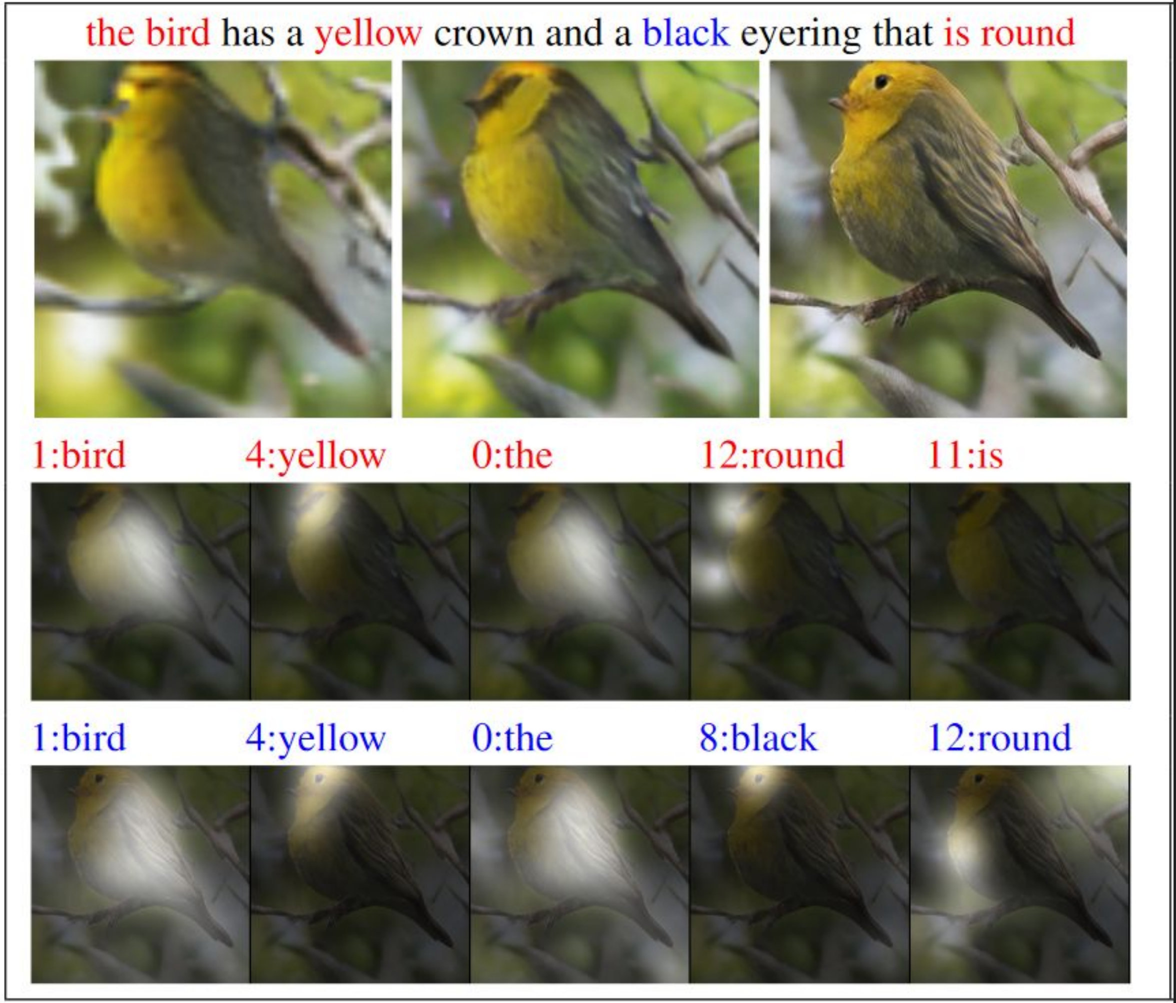


```
def gen_example(wordtoix, algo, text, iterator):
    '''generate images from example sentences'''
    from nltk.tokenize import RegexpTokenizer
    #filepath = '%s/example_filenames.txt' % (cfg.DATA_DIR)
    print('in gen_example---------------')
    directory = 'D:\\AttnGAN\\AttnGAN-Py3\\AttnGAN\\models\\bird_AttnDCGAN2\\
from_text'
    if (os.path.exists(directory)):
        for f in os.listdir(directory):
            os.remove(os.path.join(directory, f))
        print('rm dir')
    else:
        print('no rm dir')
    data_dic = {}
    text = text
    iterator = iterator
    sentences = text.split('\n')
    for ind in range(1, iterator+1):
        #sentences = text.split('\n')
        # a list of indices for a sentence
        if sentences[0] == 'end':
            print("*******Program Ended*******")
            break
        captions = []
        cap_lens = []
        for sent in sentences:
            if len(sent) == 0:
                continue
            sent = sent.replace("\ufffd\ufffd", " ")
            tokenizer = RegexpTokenizer(r'\w+')
            tokens = tokenizer.tokenize(sent.lower())
            if len(tokens) == 0:
                #print('sent', sent)
                continue
            rev = []
            for t in tokens:
                t = t.encode('ascii', 'ignore').decode('ascii')
                if len(t) > 0 and t in wordtoix:
                    rev.append(wordtoix[t])
            captions.append(rev)
            cap_lens.append(len(rev))
        max_len = np.max(cap_lens)
        sorted_indices = np.argsort(cap_lens)[::-1]
        cap_lens = np.asarray(cap_lens)
        cap_lens = cap_lens[sorted_indices]
        cap_array = np.zeros((len(captions), max_len), dtype='int64')
        for i in range(len(captions)):
            idx = sorted_indices[i]
            cap = captions[idx]
            c_len = len(cap)
            cap_array[i, :c_len] = cap
        key = 'from_text'
        data_dic[key] = [cap_array, cap_lens, sorted_indices]
        print(ind)
        te = sentences[0].replace(' ', '_')
        algo.gen_example(data_dic, ind, te)
```

Results:



Method	inception score	R-precision(%)
AttnGAN1, no DAMSM	$3.98 \pm .04$	10.37 ± 5.88
AttnGAN1, $\lambda = 0.1$	$4.19 \pm .06$	16.55 ± 4.83
AttnGAN1, $\lambda = 1$	$4.35 \pm .05$	34.96 ± 4.02
AttnGAN1, $\lambda = 5$	$4.35 \pm .04$	58.65 ± 5.41
AttnGAN1, $\lambda = 10$	$4.29 \pm .05$	63.87 ± 4.85
AttnGAN2, $\lambda = 5$	**4.36** $\pm$ **.03**	**67.82** $\pm$ **4.43**
AttnGAN2, $\lambda = 50$ **(COCO)**	**25.89** $\pm$ **.47**	**85.47** $\pm$ **3.69**


the bird has a yellow crown and a black eyering that is round
1:bird
4:yellow
0:the
12:round
11:is
1:bird
4:yellow
0:the
8:black
12:round

Conclusion and Future Scope:

1. Our model generates images based on input captions with a inception score of 4.36 ±.03 on CUB dataset and 25.89±.47 on COCO dataset.

2. Future Scope: The inception score can be further improved in the future

References:

1. [1612.03242] StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks

2. The Generator | Generative Adversarial Networks

3. Generative Models