Deep Learning Competition 03: Reverse Image Caption

DataLab

Department of Computer Science,
National Tsing Hua University, Taiwan

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Reverse Image Caption

- In this work, we are interested in translating **text** in the form of single-sentence human-written descriptions directly into **image** pixels

this flower has petals that are yellow and has a ruffled stamen

this pink and yellow flower has a beautiful yellow center with many stamens

Reverse Image Caption

- Here we use Oxford-102 flower dataset and its paired texts as our training dataset

- Training

  - Input: 7370 images as training set, where each images is annotated with at most 10 texts

  - Output: image with size 64x64x3 conditioned on given text

- Testing

  - Input: 819 texts

  - Output: image with size 64x64x3 conditioned on given text

Reverse Image Caption

- Given a text, in order to generate the image which can illustrate it, what kind of model do we need? Or, more specifically, how many models are required?

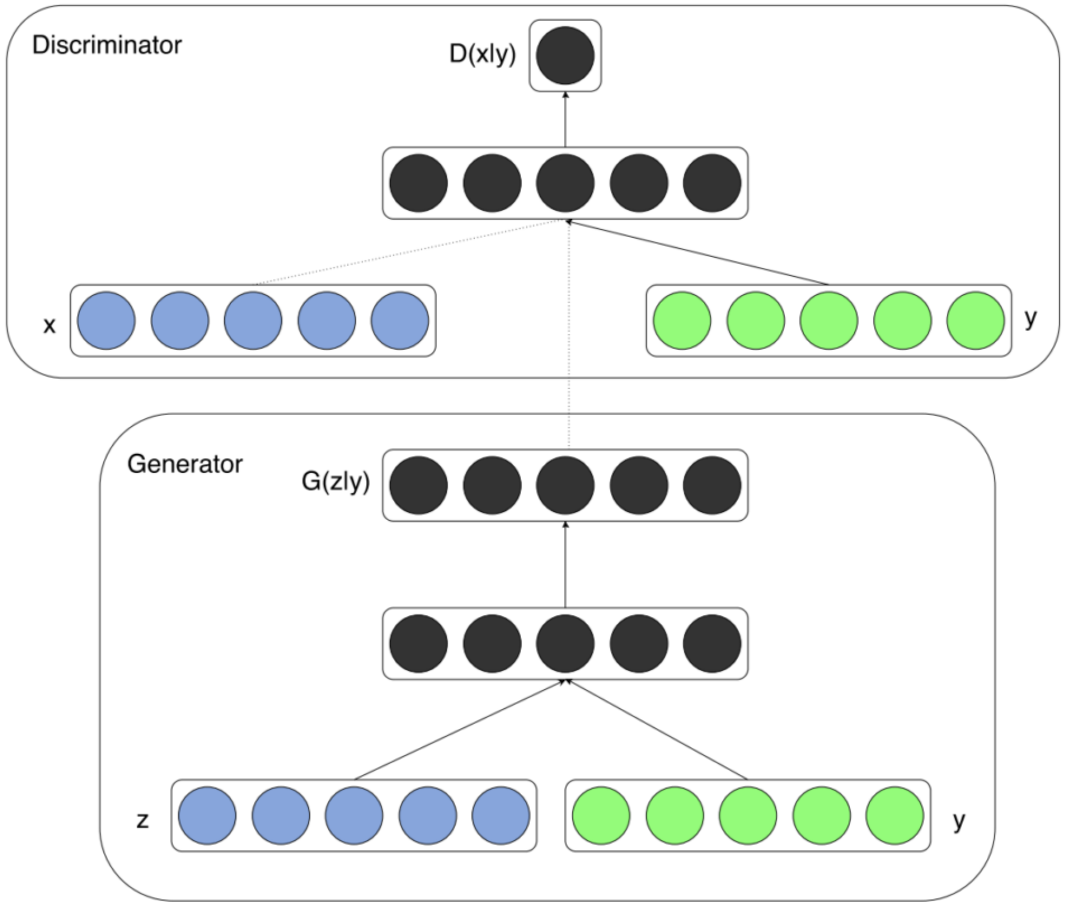
Reverse Image Caption

- Our model should have ability to understand and extract the meaning of given texts
  - Use RNN or other language model, such as BERT, ELMo or XLNet, or Word2Vec, to capture the meaning of text
- Our model should be able to generate image
  - Use GAN to generate high quality image
- GAN-generated image should illustrate the text
  - Use conditional-GAN to generate image conditioned on given text

Conditional GAN

- GANs can be extended to a conditional model if both the generator and discriminator are conditioned on some extra information y

- We can perform the conditioning by feeding y into both the discriminator and generator as additional input layer

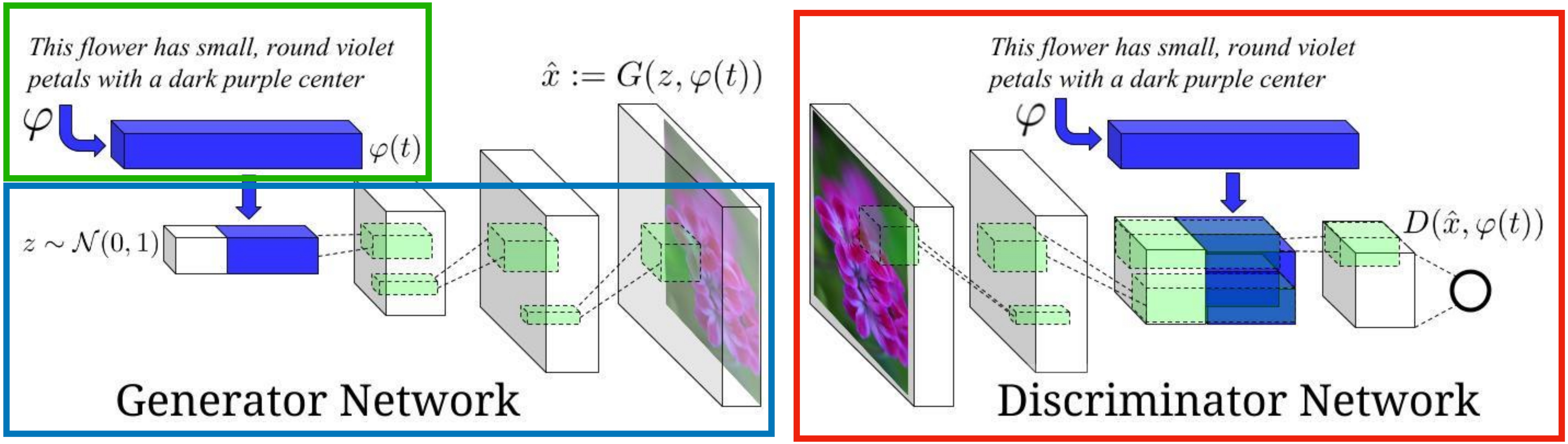


Conditional GAN

- There are two motivations for using some extra information in a GAN model
  - Improve GAN
  - Generate targeted image
- Additional information that is correlated with the input images, such as class labels, can be used to improve the GAN
  - This improvement may come in the form of more **stable training**, **faster training**, and/or **generated images that have better quality**

Conditional GAN

- Therefore, we need three models in this task, they are
  - Text encoder
  - Generator
  - Discriminator



Text Encoder

- A RNN encoder that captures the meaning of input text
  - Input: text, which is a list of ids
  - Output: embedding, or hidden representation of input text

Generator

- A image generator which generates the target image illustrating the input text
  - Input: hidden representation of input text and random noise z with random seed
  - Output: target image, which is conditioned on the given text, in size 64x64x3

Discriminator

- A binary classifier which can discriminate the real and fake image
- Real Image
  - Input: real image and the paired text
  - Output: a floating number representing the result, which is expected to be 1
- Fake Image
  - Input: generated image and paired text
  - Output: a floating number representing the result, which is expected to be 0

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Evaluation

- In this competition, we use both inception score and cosine similarity as our final score to evaluate quality and diversity of generated images. The final score is based on:
  - Similarity of images and the given contents. How similar are the generated images and the given texts?
  - KL divergence of generated images. Are the generated images very diverse?
- Score range:
  - Lowest: 0
  - Highest: 1.5

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Inception Score

- In this competition, we are going to use the Inception Score (IS) for judging the image outputs of Generative Adversarial Networks (GANs)

What is Inception Score?

- Inception Score measures how realistic a GAN’s output is
  - In the words of its authors, “we find [the IS] to correlate well with human evaluation of [image quality]”. It is an automatic alternative to having humans grade the quality of images.
- The score measures two things simultaneously
  - Images have variety (e.g. each image is a different breed of flower)
  - Each image distinctly looks like something (e.g. one image is clearly a Lavender, the next a great example of a Jasmine)
- If both things are true, the score will be high. If either or both are false, the score will be low

What is Inception Score?

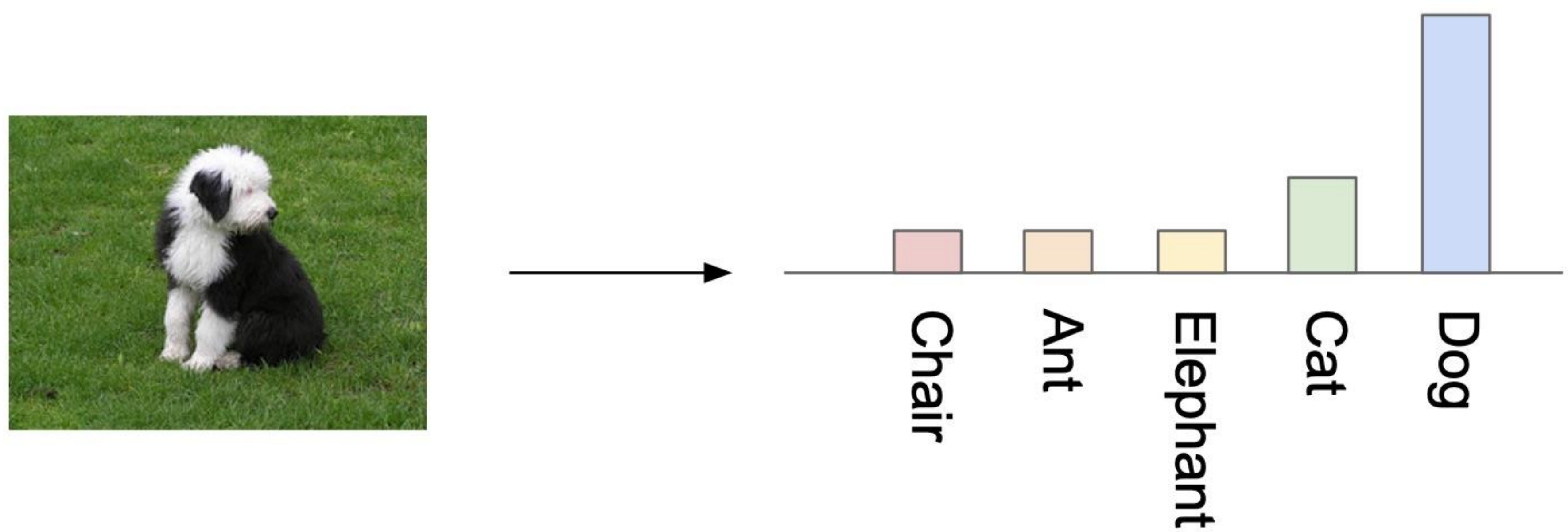
- A higher score is better. It means your GAN can generate many different distinct images
- Score range:
  - Lowest: 0
  - Highest: ∞

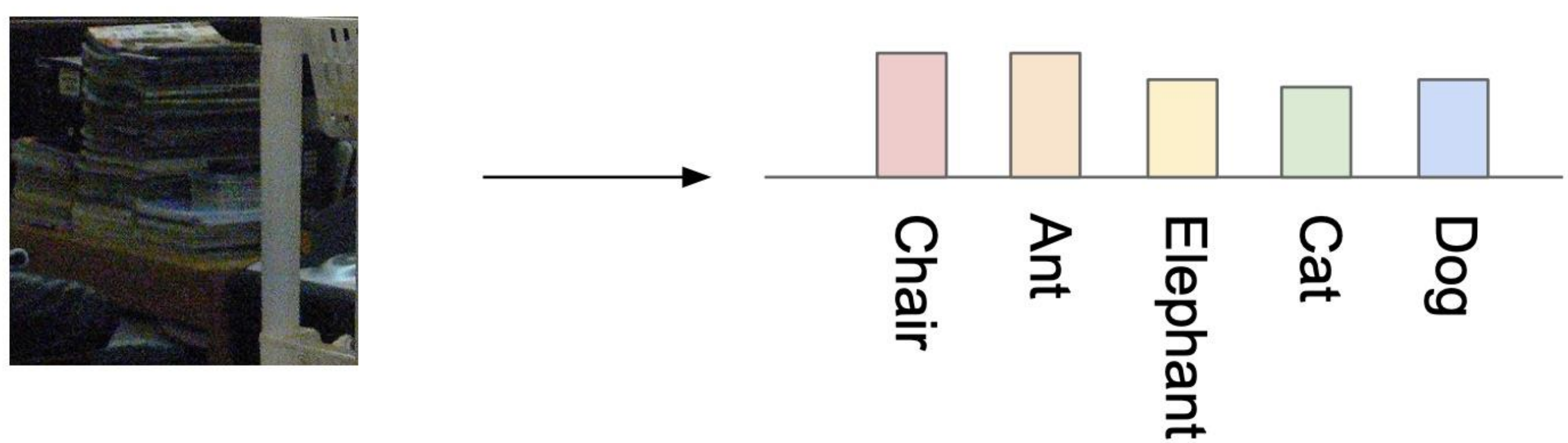
How does Inception Score work?

- The Inception score was first introduced in this paper in 2016, and has since become very popular
  - The IS takes its name from the Inception classifier, an image classification network from Google, which takes images as input, and returns probability distribution of labels for the images

How does Inception Score work?

- There are a couple useful things we can use this classifier for. We can detect if the image contains one distinct object (above), or not (below):





How does Inception Score work?

- Based on the previous slide, we obverse that
  - if the image contains just one well-formed thing, then the output of the classifier is a **narrow distribution**
  - if the image is a jumble, or contains multiple things, it is closer to the **uniform distribution** of many similar height bars

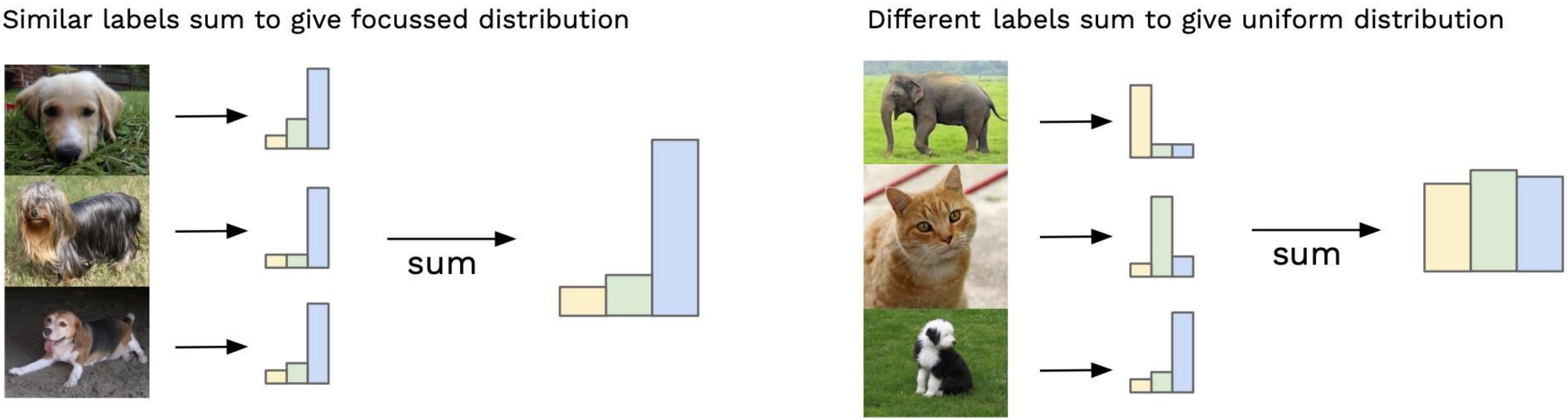
How does Inception Score work?

- The next trick we can do is combine the label probability distributions for many of our generated images

- By summing the label distributions of our images, we create a new label distribution, the **marginal distribution**

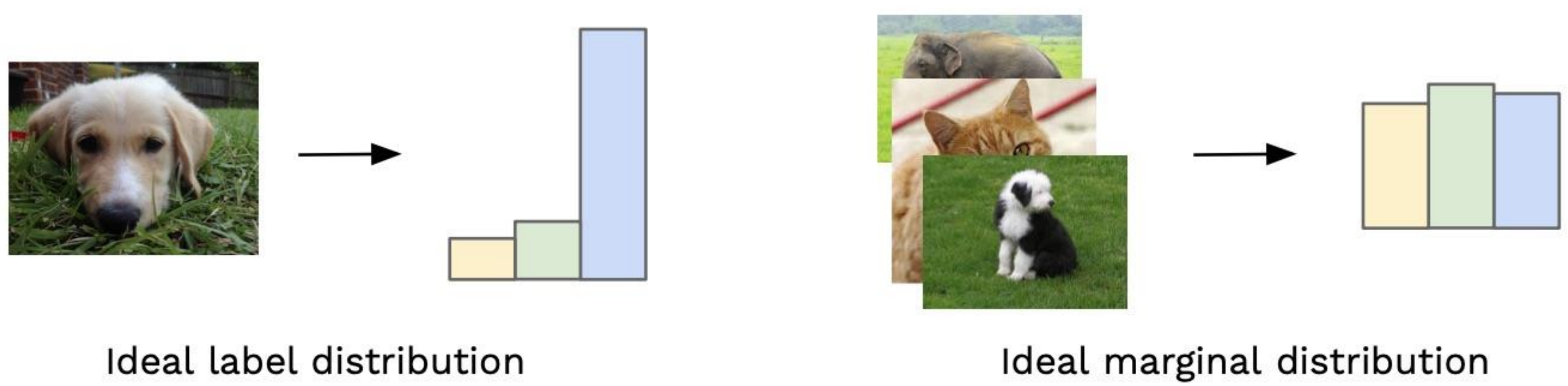
How does Inception Score work?

- The marginal distribution tells us how much variety there is in our generator’ s output:



How does Inception Score work?

- We want each image to each be distinct and to collectively have variety
- Ideal distributions are opposite shapes, the label distribution is narrow, the marginal distribution is uniform



How does Inception Score work?

- Therefore, by comparing each image’s label distribution with the marginal label distribution for the whole set of images, we can give a score of how much those two distributions differ. The more they differ, the higher a score we want to give, and this is our Inception score

KL Divergence

- To produce this score, we use a statistics formula called the Kullback-Leibler (KL) divergence. The KL divergence is a measure of how similar/different two probability distributions are. KL divergence is high when distributions are dissimilar

$$\textbf{KL Divergence} = p(y|x) * log\left(\frac{p(y|x)}{p(y)}\right)$$

- To get the final score, we take the exponential of the KL divergence and finally take the average of this for all of our images. The result is the Inception score!

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Cosine Similarity

- Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

- Instead of cosine similarity, we use **cosine distance** in this task, which is **1 - cosine similarity**

- Score range:
  - Lowest: 0
  - Highest: 1

Evaluation

- After generating images with given testing texts, you have to run evaluation script to generate `score.csv` file, and then upload it to Kaggle to get the final score

Evaluation

1. Open terminal and move to the folder containing `inception_score.py`. Otherwise you have to modify the path used in the file

2. Run python ./inception_score.py [argv1] [argv2] [argv3]

    1. argv1: directory of generated image (inference)

    2. argv2: directory of output file and its name

    3. argv3: batch size. Please set batch size to 1, 2, 3, 7, 9, 21, 39 to avoid remainder

    4. For example, run following commend `python inception_score.py ../inference/demo ../score_demo.csv 39`

3. It is better for you to know that evaluation needs to run on GPUs, please make sure the GPU resource is available

Outline

- Reverse Image Caption
- Evaluation
  - Inception Score
  - Cosine Similarity
- Precautions

Precautions

- Timeline
  - 2021/12/16(Thur) competition announced
  - 2022/01/04(Tue) 23:59(UTC) competition deadline
  - 2022/01/09(Sun) 23:59(TW) report deadline
  - 2022/01/06(Tue) winner team share (tentative)
- Scoring
  - Ranking of **private** leaderboard of competition **(50%)**
  - Inference images **(30%)**
  - Report **(20%)**

Precautions

- The final report should contain following points:
  - Pick **5 descriptions** from testing data and generate **5 images** with different noise z for each image respectively (25 images in total)
  - Models you tried during competition. Briefly describe the main idea of the model and the reason you chose that model
  - List the experiment you did. For example, data augmentation, hyper-parameters tuning, architecture tuning, optimizer tuning, and so on
  - Anything worth mentioning. For example, how to pre-train the model

Precautions

- Submit the link of Google Drive containing **report**, **model** and **819 inference images**
  - Name the report as `DL_comp3_{Your Team number}_report.ipynb`
  - Name code of trainable model as `DL_comp3_{Your Team number}_model.ipynb`
  - Place inference images under the folder called `inference`, compress that folder with the other two notebook, and then upload to Google Drive. The compressed file should be named as `DL_comp3_{Your Team number}.zip`.

```
└── DL_comp3_{Your Team Number}.zip
    ├── DL_comp3_{Your Team Number}_report.ipynb
    ├── DL_comp3_{Your Team Number}_model.ipynb
    └── inference
        ├── inference_0023.jpg
        ├── inference_0041.jpg
        ├── inference_0057.jpg
        ├── ...
        ├── ...
```

Hints

- You can find details about text to image in Generative Adversarial Text to Image Synthesis. This competition is based on this paper
- **Data augmentation** might improve performance a lot
- Use more complicated **loss function** to increase training stability and efficiency, i.e. creating more kind of training pair
- **Pretrained RNN** might have better hidden representation for input text. Additionally, it might accelerate the training process, and also make training more stable
  - Learning Deep Representations of Fine-Grained Visual Descriptions model proposes a better RNN architecture and corresponding loss function for text to image task. This architecture can encode text into image-like hidden representation

Hints

- **Different architecture of conditional GAN** might generate the image with better quality or higher resolution
  - You can try with simple GAN, DCGAN or WGAN first
  - Generative Adversarial Text to Image Synthesis proposes another RNN architecture for text to image task. The author also propose another architecture of conditional GAN to generate the images with better quality
  - StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks proposes two-stage architecture to generate more impressive images
  - Improved Training of Wasserstein GANs improves WGAN loss on conditional GAN can improve training stability
  - You can find other architecture of GAN in The GAN Zoo
- Check GAN training tips to obtain some GAN training tricks, including how to generate diverse images and to prevent mode collapsing. It is worth knowing that **GAN is not easy to train**, and those tricks are quite helpful

What you can do

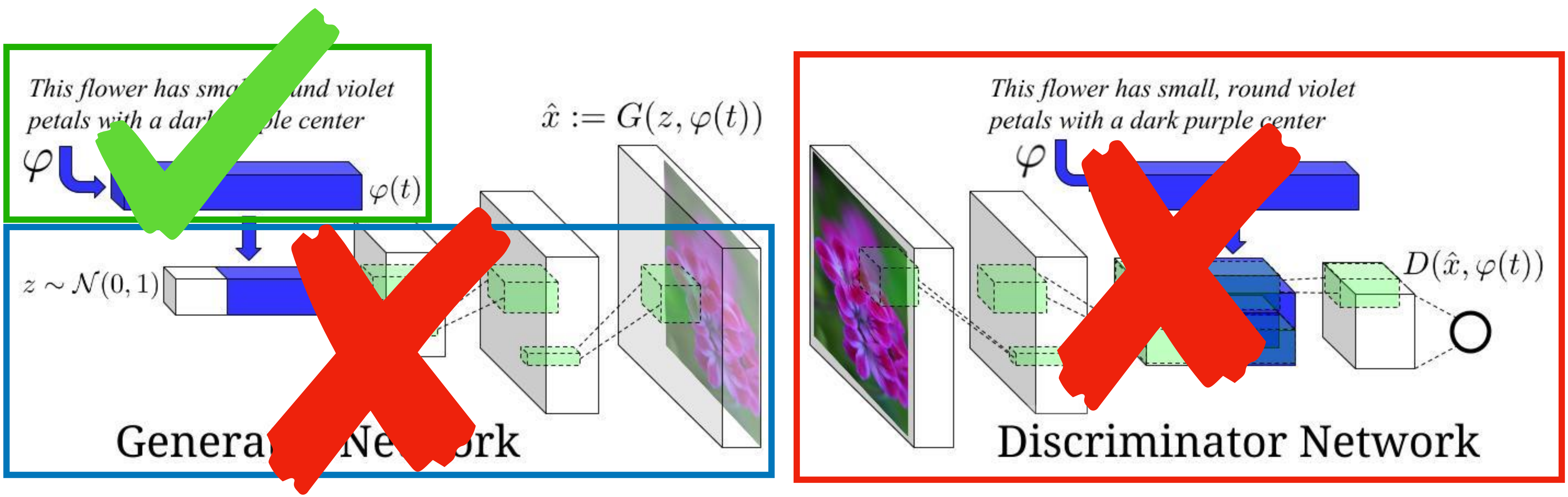
- Pre-train text encoder on other dataset
- Use pre-trained text encoder, like general purpose word embedding, pre-trained RNN or other language model, such as BERT, ELMo and XLNet. But you are **not allowed** to use any text encoder pre-trained on 102 flowers dataset
- Reuse the data and model from previous competitions and labs
- Use any package under tensorflow, but you **cannot** implement your model by `tensorlayer` API

What you should NOT do

- Use categorical labels from flower dataset in any part of model
- Use official Oxford-102 flower dataset and other image dataset to train your GAN. Pre-trained GAN and transfer learning are prohibited as well
- Clone others' project or use pre-trained model from other resources(you can only use general purpose word embedding or pretrained RNN, not pretrained GAN)
- Use text encoder pre-trained on 102 flowers dataset
- Access data or backpropagation signals from testing model in `inception_score.py` and `eval_metrics.pkl`
- Plagiarism other teams' work

What part is allowed to use pre-trained model?

- We need three models in this task, they are
  - Text encoder
  - Generator
  - Discriminator



Demo (TA80)

Demo (TA80)

1. This flower is white and yellow in color with petals that are rounded at the edges

2. The flower has a several pieces of yellow colored petals that looks similar to its leaves

3. This flower has several light pink petals and yellow anthers

4. This flower has petals that are yellow with orange lines

1 2 3

4

Demo (TA80)

Reference

- Visualization of inception score is based on [What is the Inception score?] by David Mike

- The code of inception score is based on [How to Implement the Inception Score (IS) for Evaluating GANs] by Jason Brownlee