Generative Adversary Nets

Adversarial Framework

- Discriminative model
  - "Police"
  - Learns to determine whether a sample is from the model distribution of the generative model or the data distribution
- Generative model
  - A team of "counterfeiters" trying to produce fake currency
  - Try to fool the discriminative model with its model distribution
- Until the counterfeits are indistinguishable from the genuine articles
  - Now the generative model generates a distribution indistinguishable from the data distribution

Objective of GAN

The objective is to train a parametrized function (the generator) which maps noise samples (e.g., uniform or Gaussian) to samples whose distribution is close to that of the data generating distribution

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

V(D,G)

- G(z): A generated sample from distribution z
- D(x) = Estimated (by D) prob. that x is a real data sample – D(x)=1: D regards x as a training sample w.p.1
  - D(x)=0: D regards x as a generative sample w.p.1
- The relationship with traditional minimax problem
- In a two-agent WuZi chess, V evaluates how bad the current position is to me.
  - Adversary's goal: maximize V
  - My Goal: minimize max V

Objective of GAN

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

V(D,G)

Algorithm

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

max V
D

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

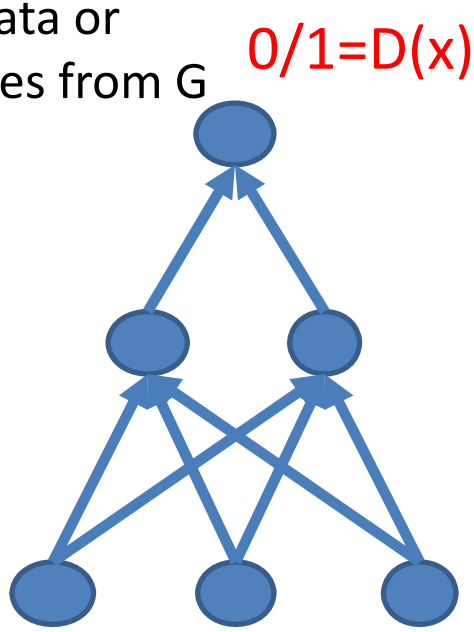
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

Approach: Objective

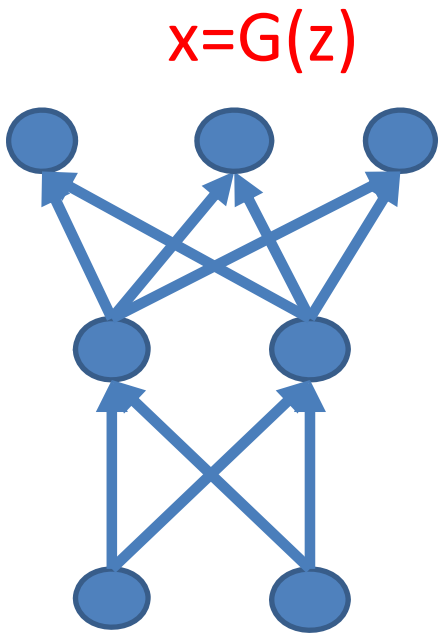
GAN training procedure: to train a discriminator which assigns higher probabilities to real data samples and lower probabilities to generated data samples, while simultaneously trying to move the generated samples towards the real data manifold using the gradient information provided by the discriminator

1: From data or
0: fake ones from G



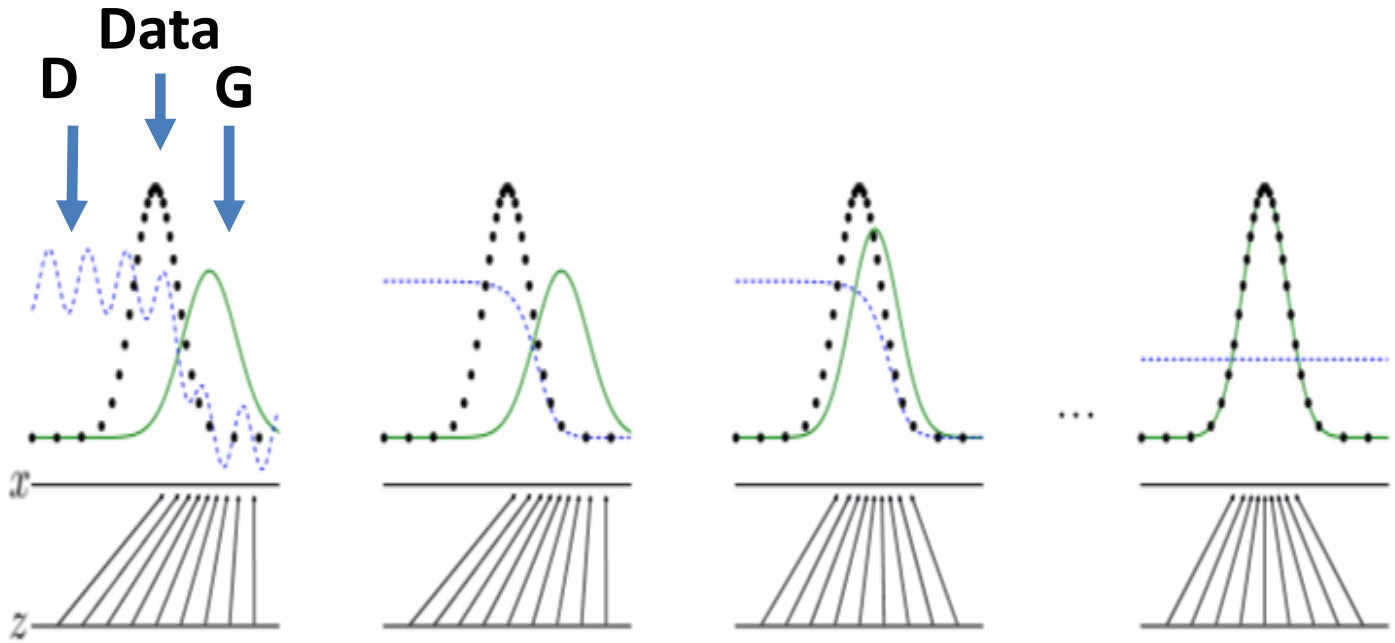
x

Fake x



z, aka random noise

Approach: Optimization



The GAN training procedure can be viewed as a non-cooperative two player game, in which the discriminator D tries to distinguish real and generated examples, while the generator G tries to fool the discriminator by pushing the generated samples towards the direction of higher discrimination values. Training the discriminator D can be viewed as training an evaluation metric on the sample space. Then the generator G has to take advantage of the local gradient $\nabla \log D(G)$ provided by the discriminator to improve itself, namely to move towards the data manifold.

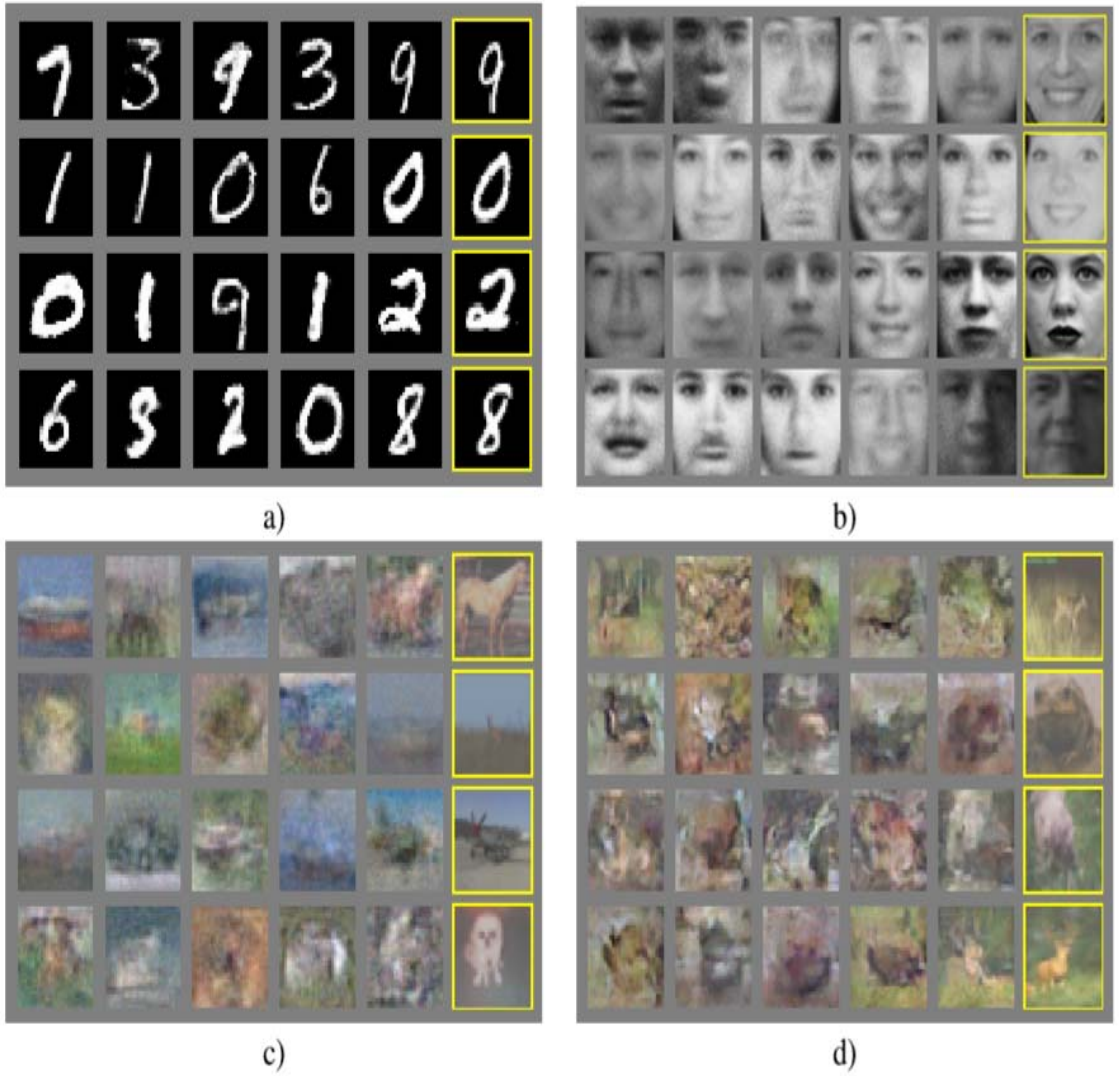


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and "deconvolutional" generator)

- the GAN is able to generate interesting local structure but globally incoherent images on various datasets

- 这种不需要预先建模的方式的缺点就是在于它太过自由了，对于较大的图片，较多的 pixel 的情形，基于简单 GAN 的方式就不太可控了。在 GAN[1] 中，每次学**参数的更新过程，被设为 D 更新 k 回，G 才更新 1 回，也是出于类似的考虑。
- 为了解决 GAN 太过自由的这个问题，一个很自然的思想便是给 GAN 加上一点点束缚，于是便有了 Conditional Generative Adversarial Nets（**CGAN**）
  - 在 D 和 G 的建模中分别加入 conditional 变量 y。
- Mirza & Osindero (2014) enlarges GAN's representation capacity by introducing an extra vector to allow the generator to produce samples conditioned on other beneficial information

Conditional Generative Adversarial Nets

本文是 GANs 的拓展，在产生 和 判别时，考虑到额外的条件 y，以进行更加"激烈"的对抗，从而达到更好的结果。

众所周知，GANs 是一个 minmax 的过程：

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

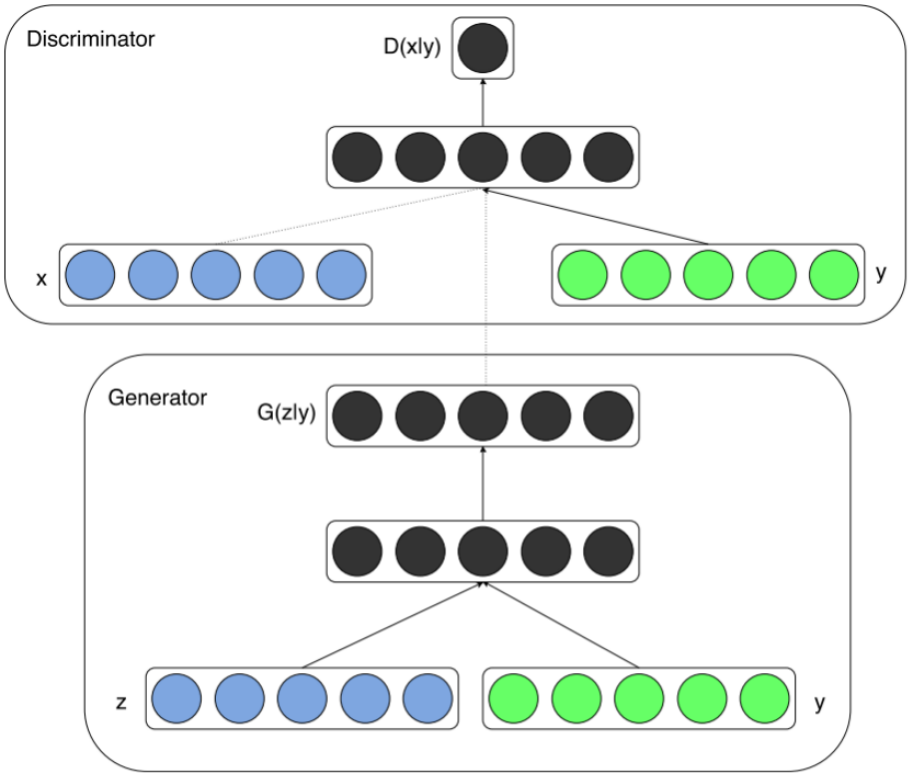
而本文通过引入 条件 y，从而将优化的目标函数变成了：

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]. \qquad (2)$$

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

GAN: The generator takes in an input noise vector from a distribution and outputs an image. The discriminator takes in this image (or a real image from the training data) and outputs a scalar describing how "real" the image is.

A conditional GAN (CGAN): both the discriminator and the generator receive another piece of information as an input. This information is likely in the form of some sort of class label or another image.



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

CGAN $$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

Laplacian Pyramid of Adversarial Networks

samples of what the generator outputted in Goodfellow's 2014 paper.

As you can see, the generator worked well with digits and faces, but it created very fuzzy and vague images when using the CIFAR-10 dataset.

- In order to fix this problem, Emily Denton, et al published the paper titled “Deep Generative Image Models using Lapalacian Pyramid of Adversarial Networks”.
- The main contribution is a type of network architecture that produces high-quality generated images that are mistaken for real images almost 40% of the time when assessed by human evaluators .
- It is hard to produce a large, complex, and natural image that is good enough to convince a trained discriminator. The way the authors combat this is by using multiple CNN models to sequentially generate images in increasing scales.

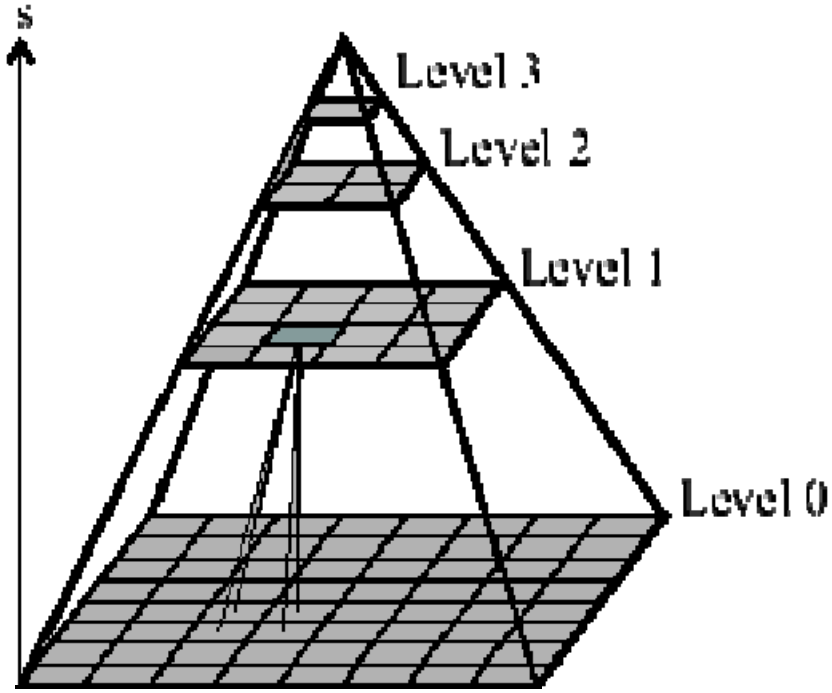
As Emily Denton said in her ***talk*** on LAPGANs

“It’s pretty easy to generate a low resolution image. It’s also not too hard to generate a slightly higher resolution image conditioned on a low resolution image”

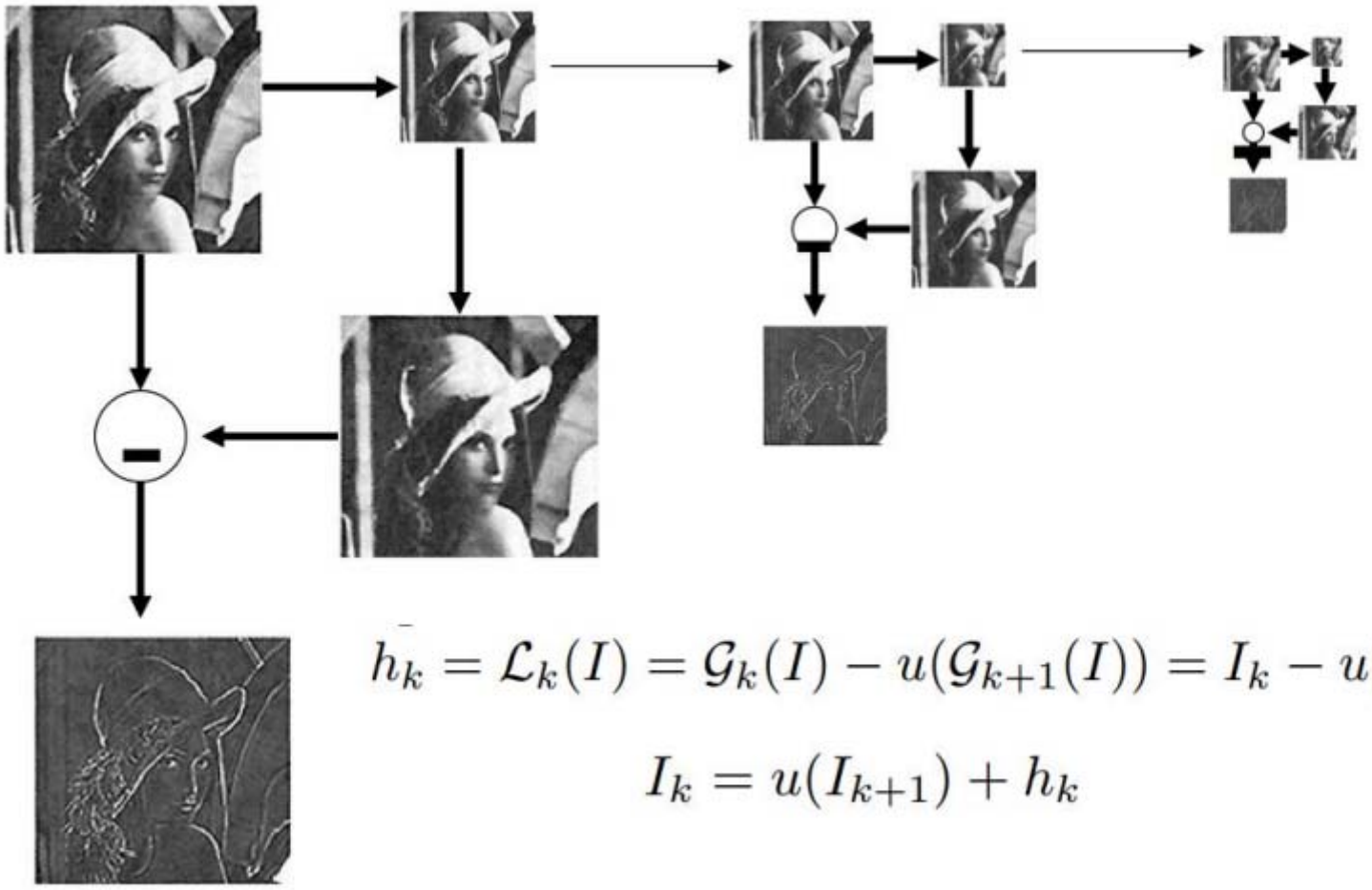
Laplacian Pyramid

- a Laplacian pyramid is basically an image representation that consists of a series of filtered images at successively sparser densities .
- The idea is that each level of this pyramid representation contains information about the image at a particular scale. It is a sort of decomposition of the original image.

http://www.cs.utah.edu/~arul/report/node12.html



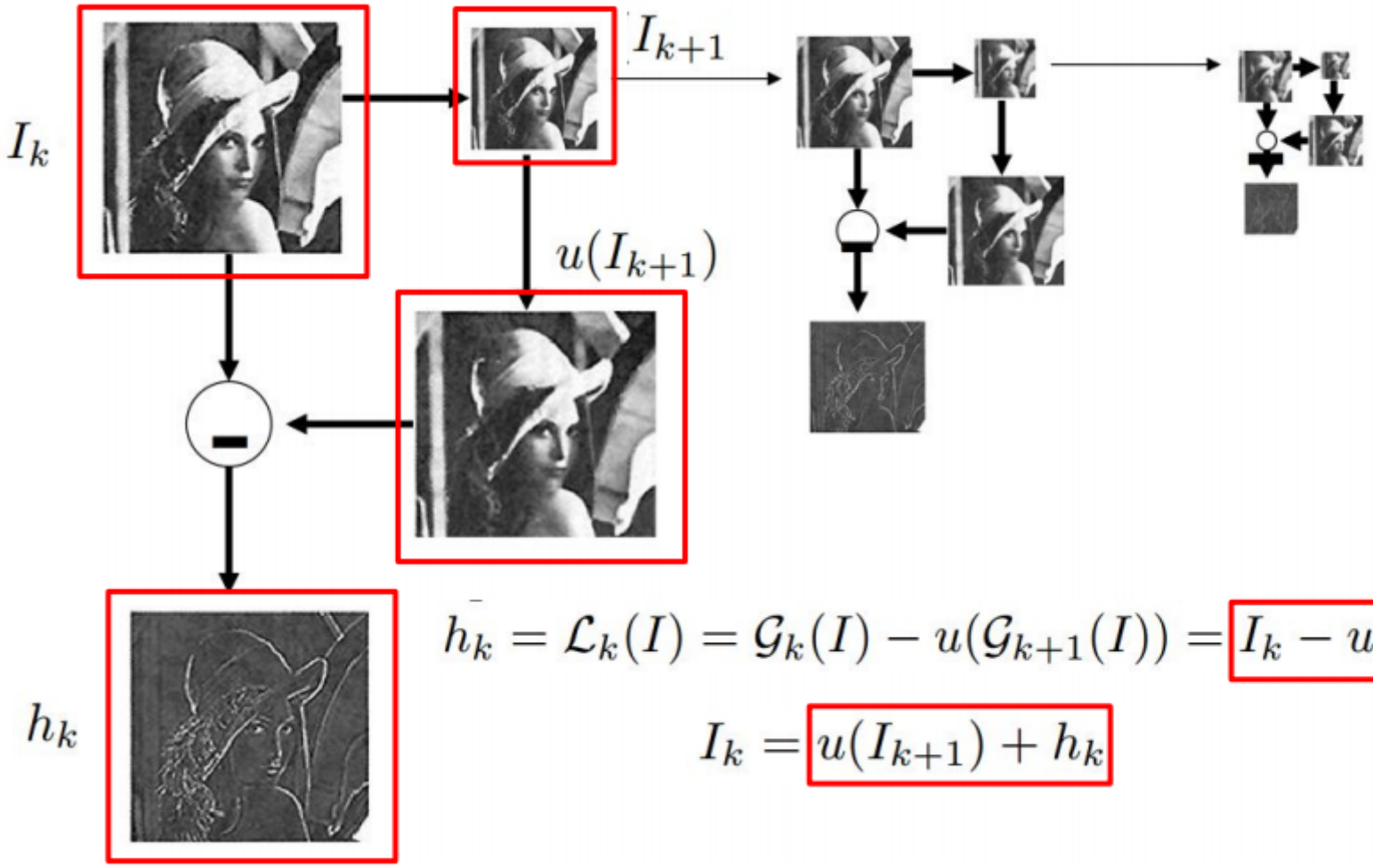
Laplacian pyramid



$$h_k = \mathcal{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = I_k - u(I_{k+1})$$

$$I_k = u(I_{k+1}) + h_k$$

Laplacian pyramid



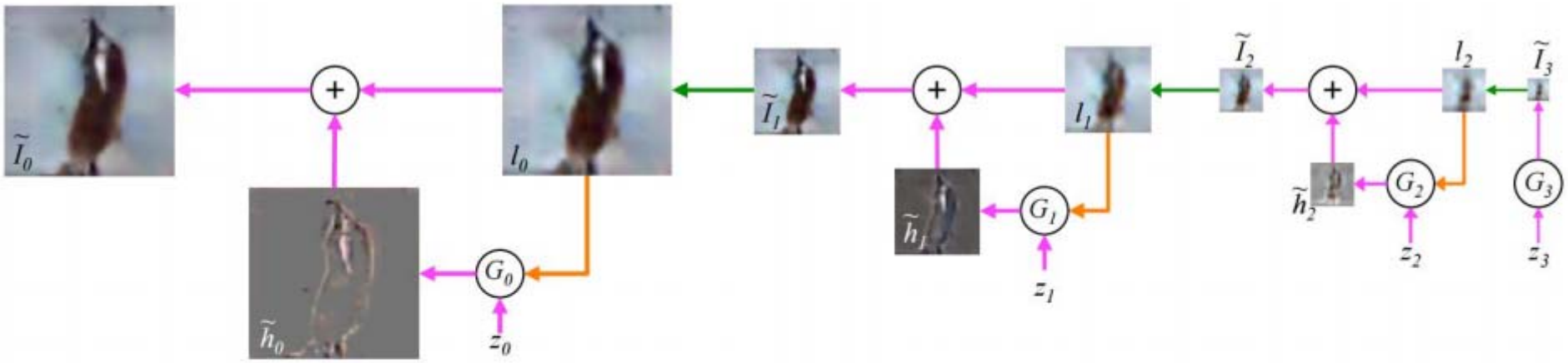
$$h_k = \mathcal{L}_k(I) = \mathcal{G}_k(I) - u(\mathcal{G}_{k+1}(I)) = \boxed{I_k - u(I_{k+1})}$$

$$I_k = \boxed{u(I_{k+1}) + h_k}$$

Network Architecture

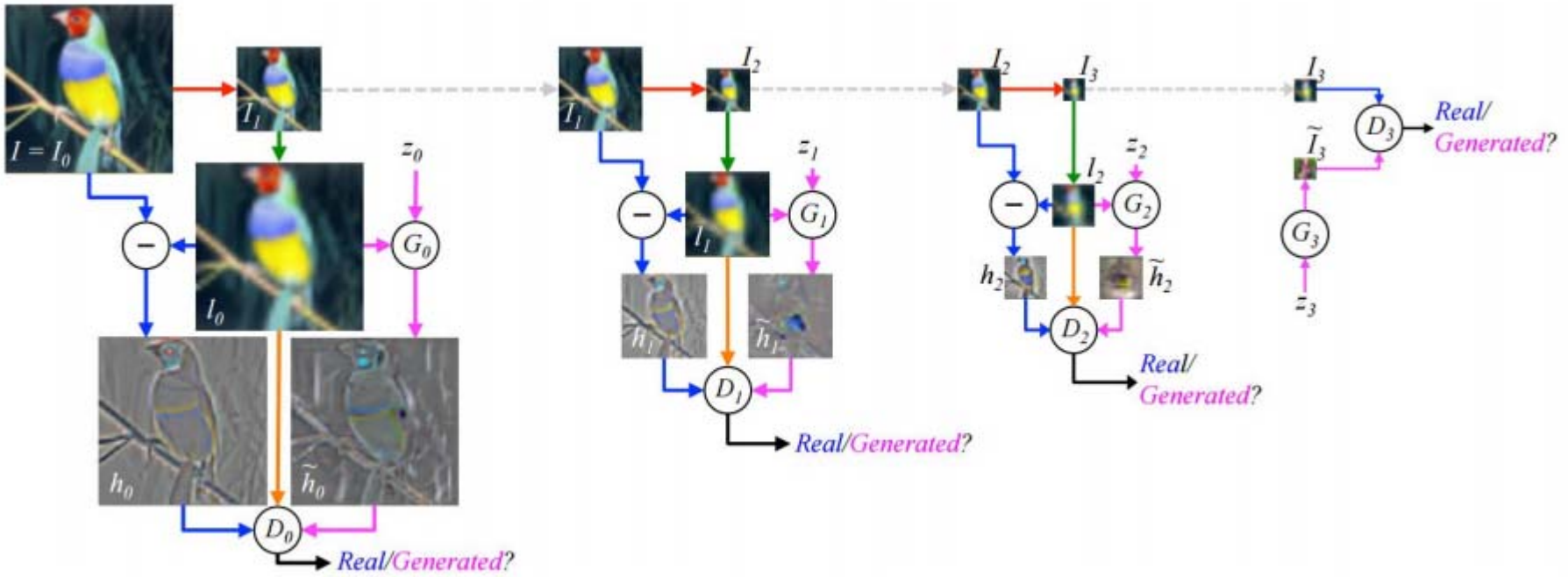
- The authors propose a set of convnet models and that each layer of the pyramid will have a convnet associated with it.
- The change is the traditional GAN structure is that instead of having just one generator CNN that creates the whole image, we have a series of CNNs that create the image **sequentially** by slowly increasing the resolution (aka going along the pyramid) and refining images in a coarse to fine fashion.
- Each level has its own CNN and is trained on two components. One is a low resolution image and the other is a noise vector (which was the only input in traditional GANs). This is where the idea of CGANs come into play as there are multiple inputs.
- The output will be a generated image that is then upsampled and used as input to the next level of the pyramid.

Image Generation

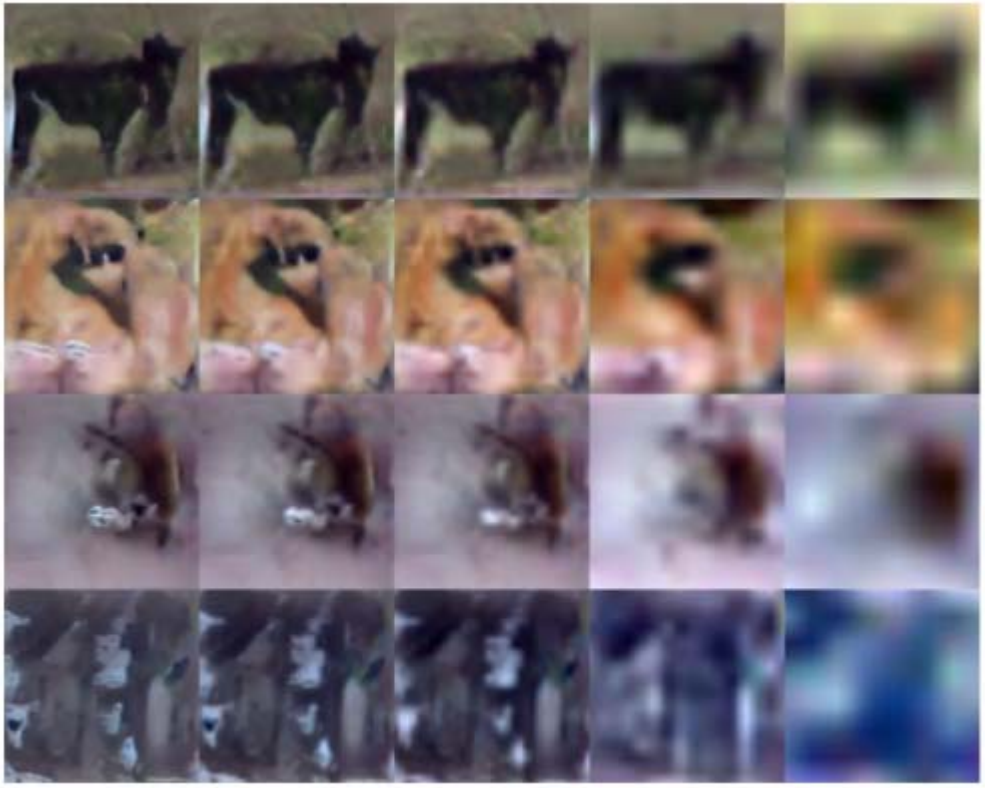


$$\tilde{I}_k = u(\tilde{I}_{k+1}) + \tilde{h}_k = u(\tilde{I}_{k+1}) + G_k(z_k, u(\tilde{I}_{k+1}))$$

Training

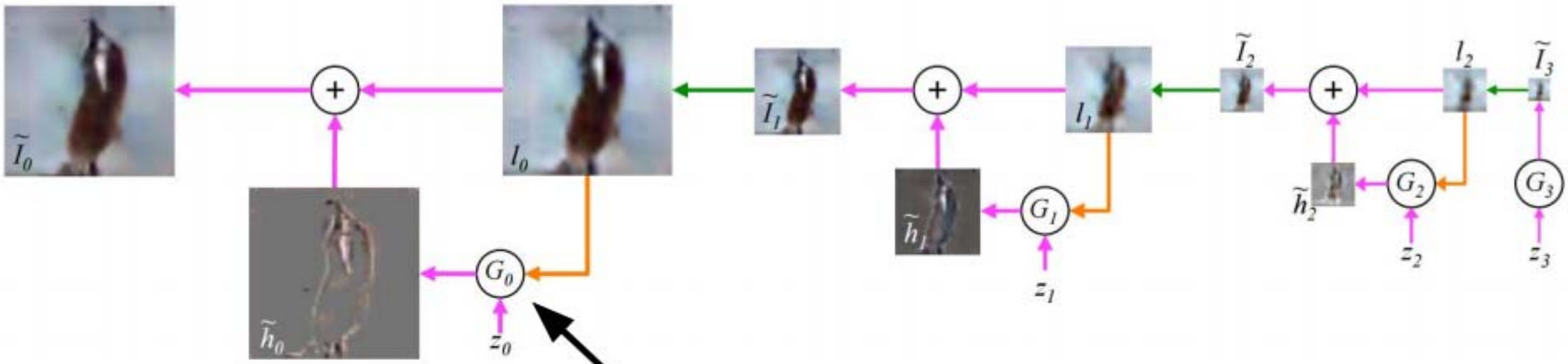

$I = I_0$
I_1
z_0
G_0
I_0
h_0
$\tilde{h}_0$
D_0
Real/Generated?
I_2
z_1
G_1
h_1
$\tilde{h}_1$
D_1
Real/Generated?
I_3
z_2
G_2
h_2
$\tilde{h}_2$
D_2
Real/
Generated?
$\tilde{I}_3$
G_3
z_3
D_3
Real/
Generated?

Generation: Coarse to fine



Some thoughts on the method

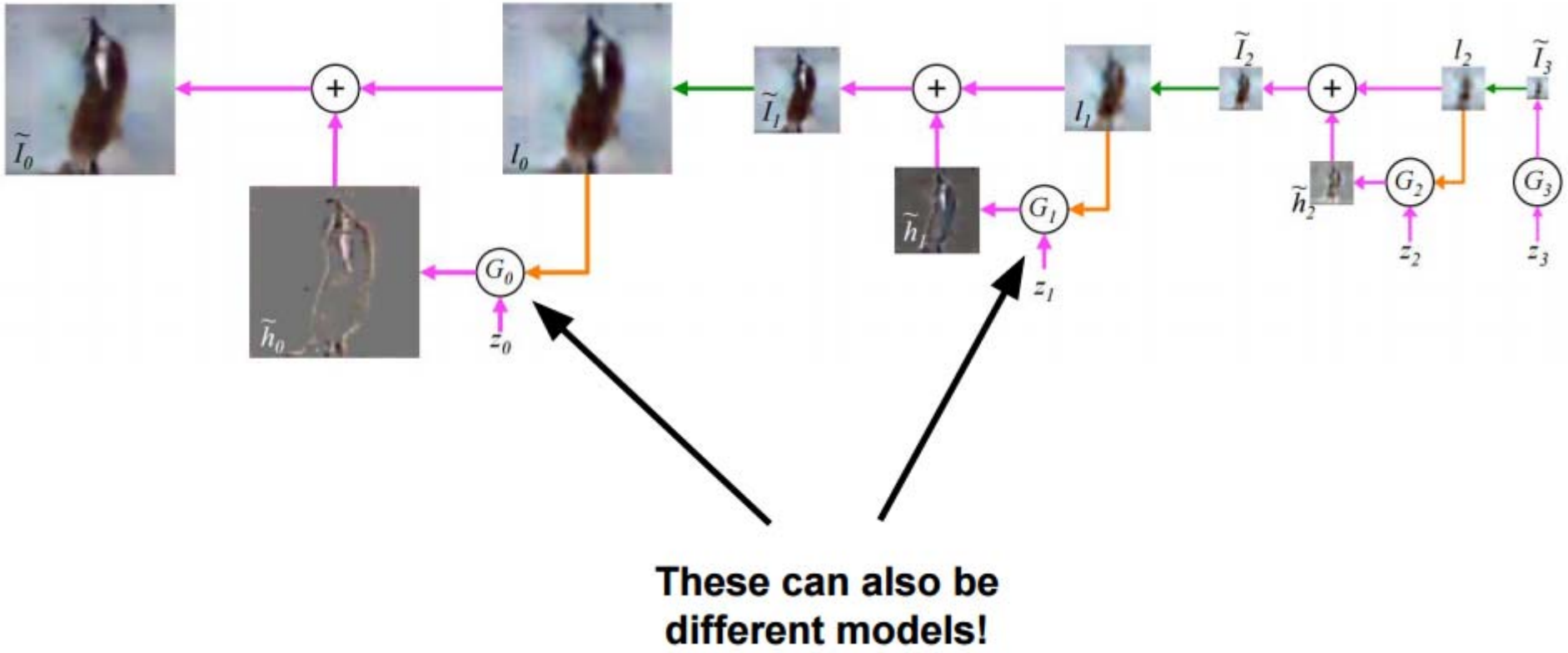
- The Laplacian Pyramid Framework is independent of the Generative Model



Possible to use a completely different model like Pixel RNN

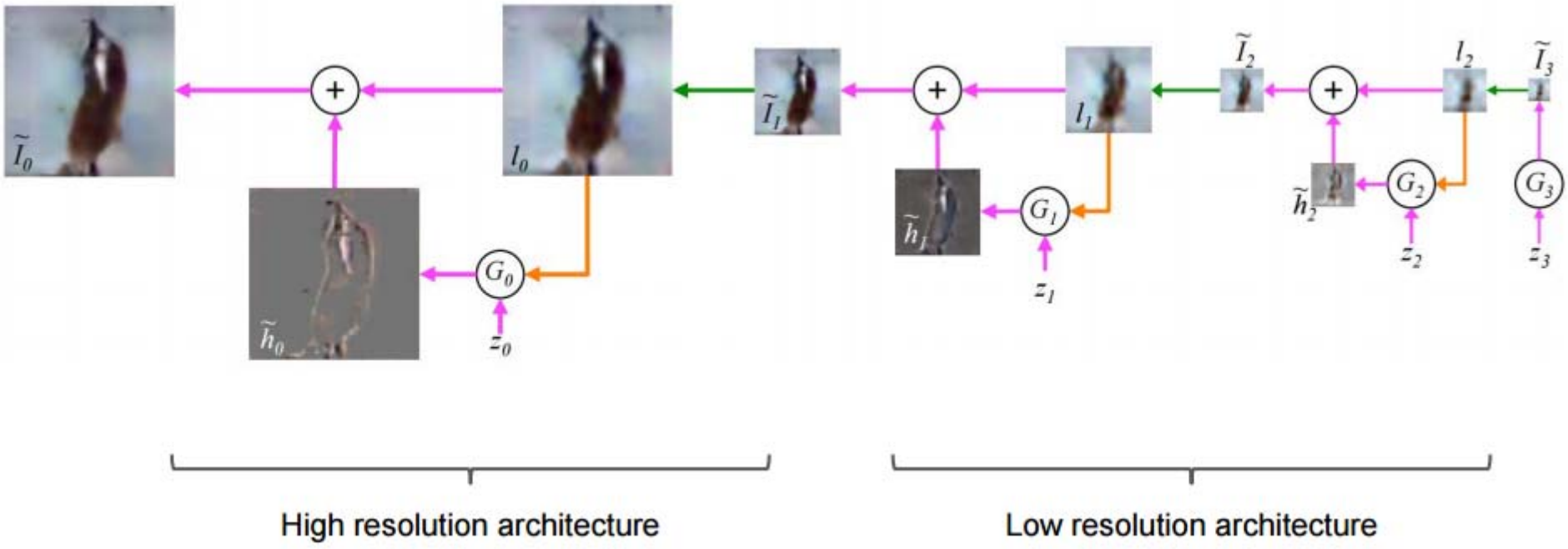
Some thoughts on the method

- The Generative Models at each step can be totally different!



Some thoughts on the method

- The Generative Models at each step can be totally different!



Generated images of ships

Generated images of horses

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

- stabilize Generative Adversarial networks with some architectural constraintsReplace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator
- Remove fully connected hidden layers for deeper architectures. Just use average pooling at the end.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

http://www.itdadao.com/articles/c15a445113p0.html

- use the discriminator as a pre-trained net for CIFAR-10 classification and show pretty decent results.
- generate really cool bedroom images that look super real

http://www.tuicool.com/articles/auYFvyb

Generative Adversarial Text to Image Synthesis

Motivation

Current deep learning models enable us to...

➢ Learn feature representations of images & text

➢ Generate realistic images & text pull out images based on captions generate descriptions based on images answer questions about image content

Author's code available at:
https://github.com/reedscot/icml2016

http://www.tuicool.com/articles/Qv2AJ3N

"Two pizzas sitting on top of a stove top oven"

Problem - Multimodal distribution

- Many plausible image can be associated with one single text description
- Previous attempt uses Variational Recurrent Autoencoders to generate image from text caption but the images were not realistic enough. (Mansimov et al. 2016)

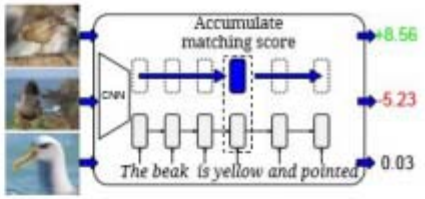
What GANs can do

- CGAN: Use side information (eg. classes) to guide the learning process
- Minimax game: Adaptive loss function

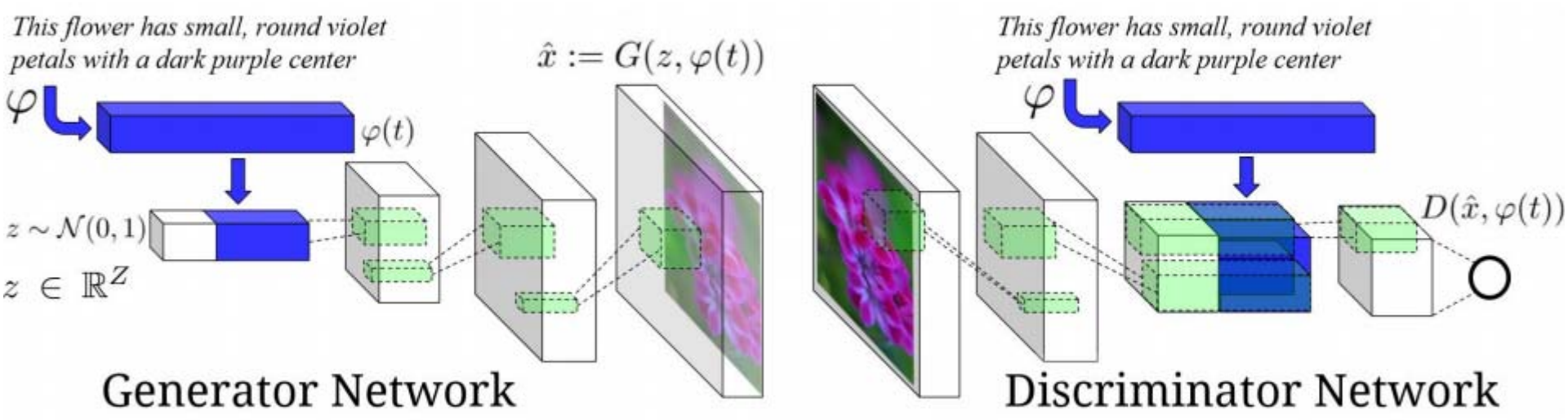
➢ Multi-modality is a very well suited property for GANs to learn.

The Model - Basic CGAN

Pre-trained char-CNN-RNN



Learns a compatibility function of images and text -> joint embedding



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

The Model - Variations

GAN-CLS

In order to distinguish different error sources: Present to the discriminator network 3 different types of input. (instead of 2)

Algorithm

Input: minibatch images x, matching text t, mismatching $\hat{t}$, number of training batch steps S
for $n = 1$ **to** S **do**
  $h \leftarrow \varphi(t)$ {Encode matching text description}
  $\hat{h} \leftarrow \varphi(\hat{t})$ {Encode mis-matching text description}
  $z \sim \mathcal{N}(0,1)^Z$ {Draw sample of random noise}
  $\hat{x} \leftarrow G(z, h)$ {Forward through generator}
  $s_r \leftarrow D(x, h)$ {real image, right text}
  $s_w \leftarrow D(x, \hat{h})$ {real image, wrong text}
  $s_f \leftarrow D(\hat{x}, h)$ {fake image, right text}
  $\mathcal{L}_D \leftarrow \log(s_r) + (\log(1 - s_w) + \log(1 - s_f))/2$
  $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$ {Update discriminator}
  $\mathcal{L}_G \leftarrow \log(s_f)$
  $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$ {Update generator}
end for

The Model - Variations cont.

GAN-INT

In order to generalize the output of G: Interpolate between training set embeddings to generate new text and hence fill the gaps on the image data manifold.

Updated Equation

$$\min_G \max_D V(D,G) =$$

$$= \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)]$$

$$+ \mathbb{E}_{x\sim p_z(z)}[\log(1 - D(G(z)))] +$$

$$\mathbb{E}_{t_1,t_2\sim p_{data}}[\log(1 - D(G(z, \beta t_1 + (1-\beta)t_2)))]$$

{fake image, fake text}

GAN-INT-CLS: Combination of both previous variations

Disentangling

- Style is background, position & orientation of the object, etc.
- Content is shape, size & colour of the object, etc.

- Introduce S(x), a style encoder with a squared loss function:

$$\mathcal{L}_{style} = \mathbb{E}_{t,z\sim\mathcal{N}(0,1)}||z - S(G(z,\varphi(t)))||_2^2$$

- Useful in generalization: encoding style and content separately allows for different new combinations

Intuitive Explanation

GAN 这种竞争的方式不再要求一个假设的数据分布，也就是说我们不用 formulate p(x)，而是直接进行 sampling，从而真正达到了理论上可以完全逼近真实数据。这是 GAN 最大的优势