CS3485

Deep Learning for Computer Vision

Lec 19: Advanced GANs

Announcements

- Final project:
  - **Groups of 1-4 students.**
  - Three options for the **theme**:
    1. Do a literature review on the SOTA of some Computer Vision task (like Image Classification for example).
    2. Try to solve any problem of your choice using Deep Learning (it does not need to be in Computer Vision, it can be involving audio, text, etc.)
    3. Implement a software that uses DL (does not need to be related to CV).
  - The teams should send a **proposal** the Dec 4th with a problem statement, motivation, the main tasks and how each student will contribute to it.
  - The **presentation** will be in person on the day/time for our final exam, and it should last for at least 8 min, such that each student member presents for at least 3 min. It should also present some sort of **demonstration**. If some student can't be present, they should join via zoom (or ask me for an exception).
  - **More information on it on the Syllabus and on the website.**

Announcements

- **KLAVIERFEST!**

SCHEDULE

Friday, November 8

10:30 a.m.–11:30 a.m.	**Torin Hopkins,** Postdoctoral Research Fellow, University of Singapore "Minds, Machines, and Music: Interfacing with the Digital World to Advance Musicianship and Musical Practice"
11:30 a.m.–12:00 p.m.	Panel Discussion and Q&A
12:00 p.m.–1:30 p.m.	Lunch
2:00 p.m.–3:00 p.m.	**Cynthia Lee Wong,** Digital Artist and Composer "Orchestrating the Future: Amplifying Imagination with Technology"
3:30 p.m.–4:00 p.m.	Panel Discussion and Q&A
4:00 p.m.–5:00 p.m.	**Hans Tutschku,** Fanny P. Mason Professor of Music, Harvard University "The Piano in My Life: From Preparation to Live Electronics and AI"
5:00 p.m.–5:30 p.m.	Panel Discussion and Q&A
5:30 p.m.–7:30 p.m.	Dinner
7:30 p.m.–9:00 p.m.	Gala Concert: Piano Duo **DUO MUNDI GEORGE & GULI**

Saturday, November 9

9:00 a.m.–10:00 a.m.	**João Paolo Casarotti,** Professor of Piano, Glendale Community College "The Use of Technology and AI Applications in Piano Instruction"
10:00 a.m.–10:30 a.m.	Panel Discussion and Q&A
10:30 a.m.–10:45 a.m.	Break
10:45 a.m.–noon	**Konrad Swierczek,** McMaster Institute for the Music and the Mind "Does AI Hear What We Hear? Testing Music Technology's Human Touch"
Noon–1:30 p.m.	Lunch
1:30 p.m.–2:30 p.m.	**Brian Liu,** Bowdoin College Class of 2025 "Embracing Eclecticism: How Music and AI Empower the Modern Student"
2:30 p.m.–3:00 p.m.	Final Panel Discussion and Closing Remarks

Announcements

- Project Proposal:
  - Due on Dec 4th, and there is a submission link on canvas,
  - Remember it counts as part of the grade!
- Info about late submissions on the website (more for next year, actually).
- Interesting application of dense pose estimation:

(Tentative) Lecture Roadmap

Basics of Deep Learning

- Intro to Computer Vision
- Linear Classifiers and Perceptron
- Multilayer Perceptron
- Optimization in Deep Learning
- Pytorch I – MLPs
- Pytorch II – Images and Regularization
- Convolutional Neural Networks
- Data Augmentation and Deep CNNs
- Transfer Learning and Residual Nets
- Inception Net and what CNNs learn
- Adversarial Examples and Self-supervision

Deep Learning and Computer Vision in Practice

- Intro to MLOps
- Intro to Object Detection
- Fast Object Detection
- Intro to Image Segmentation
- Applications of Detection and Segmentation
- Autoencoders
- Image Generation with GANs
- Advanced GANs
- The Attention Mechanism
- Transformers and ChatGPT
- Image Generation by Prompt
- Misc. Topics

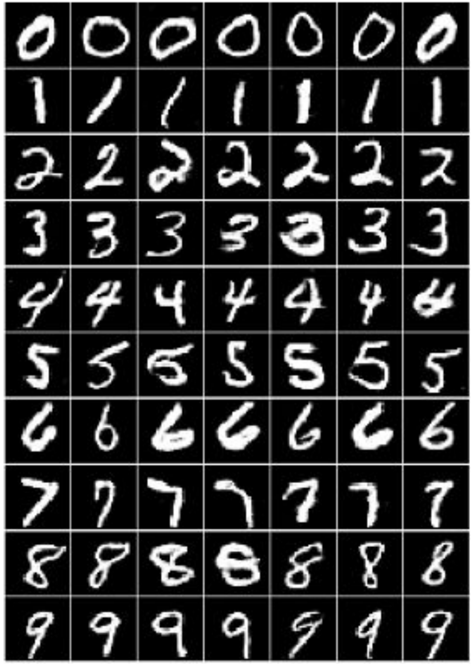
More interesting GANs

- Last time we saw how GANs can generate new digits from the MNIST dataset and new faces.
- Although interesting, these results **were not realist enough** compared to more modern GAN architectures.
- Today, we'll see how modern GANs (such as StyleGAN) are able to generate **visually stunning high-resolution face images**!
- Before that, we'll also see how to **conditionally generate new images** using GANs which will provide us with tools to solve many other problems in image generation.

New faces generated by StyleGAN

Conditional GANs

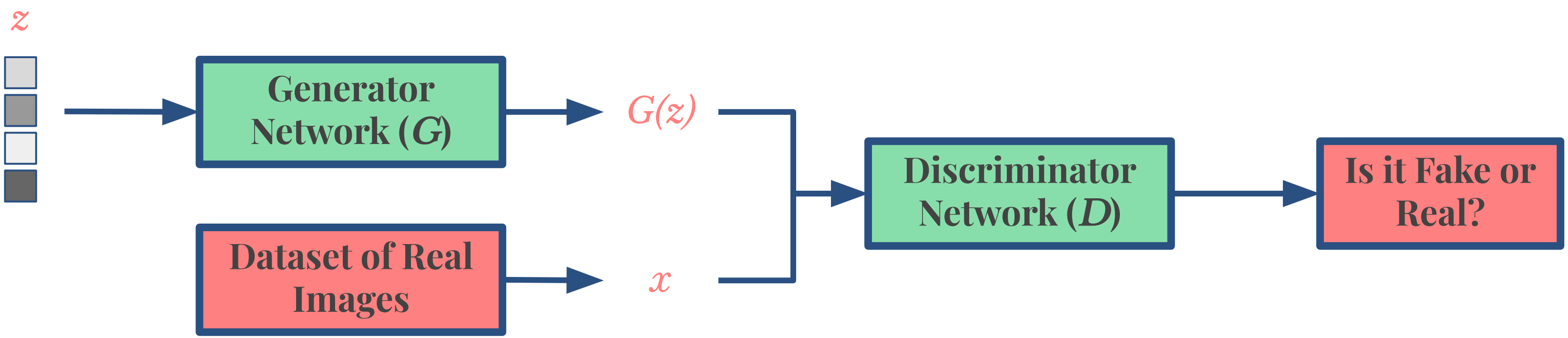
- All GAN models we have seen so far model a probability density in high dimension and provide means to sample according to it, which is useful **for image synthesis only**.
- However, most of the practical applications require the ability to sample a **conditional distribution**, i.e., sample new data conditioned on some information we have at our disposal.
- For example, we may want to sample a datapoint conditioned on its class (I may want to sample only new 7's instead of any random digit).
- **Conditional GAN**, published in 2014, was conceived to adapt our previous, simple GAN architecture (called **Vanilla GAN**) to this setting.



New MNIST digits generated according to their classes.

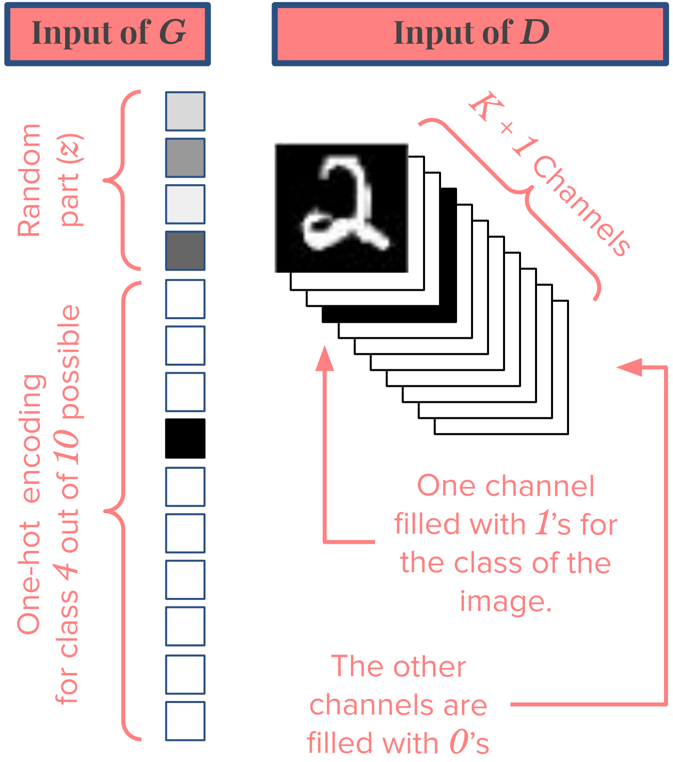
Conditional GANs

- Let's first review our previous GAN approach:
  - We have a **Generator Network** G that takes in a random vector z and produces a new, generated image $G(z)$.
  - We also have a **Discriminator Network** D that takes in an image as its input and classifies it in fake (i.e., generated by G) or real (i.e., coming from an image dataset).
  - The goal is twofold: **(1)** train a very good discriminator network and **(2)** train a generator that beats this discriminator.



Conditional GANs

- In Conditional GAN, the same training approach is taken, but now both generator and discriminator inputs will carry **class information**.
- To do that, we just need to "add more data" to both inputs. Say we have K classes ($K = 10$ for MNIST):
  - For the **Generator input**, append to z a vector of K dimensional one-hot encoding of the class you want the generated image to be from.
  - For the **Discriminator input**, append K more channels to the input image such that they work as an one-encoding of the that image's class (from either the image dataset or the generator's input)*.



* Note that, even if the image is realistic, but the class that image is attached to is not the correct one the discriminator here should output "fake".

Image-to-Image Translation

- We can use the same principle of conditioning the generation to a class to create interesting conditions.
- For example, we may want to generate a realistic image conditioned in a certain edge map, i.e., a new image that has its edges given by the user.

Edge Map | Edge Source | New Image

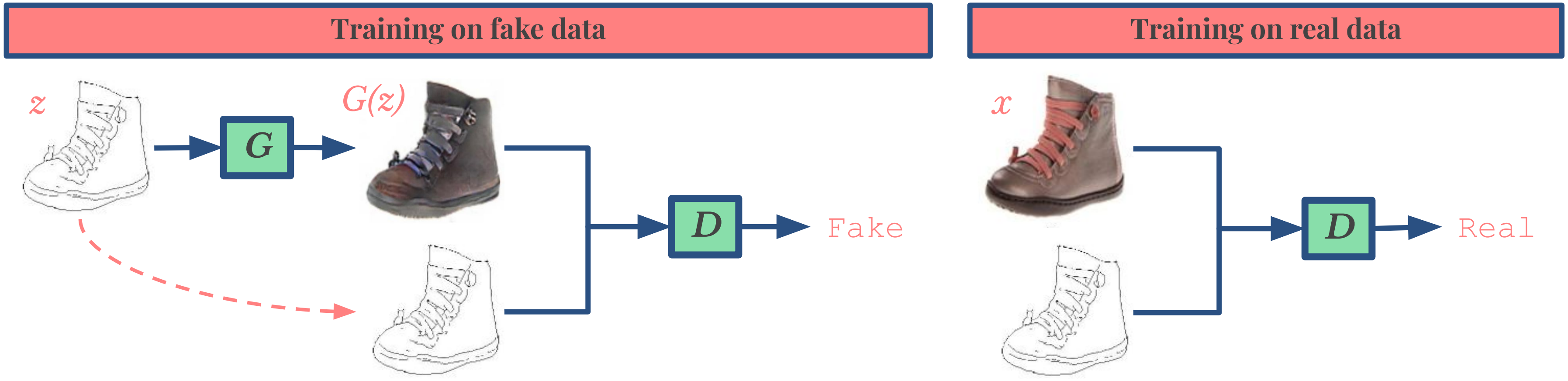
- This approach will be very useful for the task of **Image-to-Image Translation**:

Image-to-image translation is the task of taking images from one domain and transforming them so they have the characteristics of images from another domain.

- In our example above, we converted an image in the domain of edges to the domain of realistic RBG images.

Pix2Pix

- Published in 2016, the Pix2Pix strategy to solve image to image translation involved a GAN network that used the concepts from Conditional GANs.
- Here, the difference is that the generator receives an image input z in one domain (edge map, for example) and outputs the corresponding image on the other domain.
- The discriminator is then tasked to check if the pairings edge map/image are realistic.



Pix2Pix

Edges to Photos

Pix2Pix

- Note that the edge maps don’t need to be realistic. These are the results from when you input a line drawing to generator trained on edge maps.

Drawings to Photo

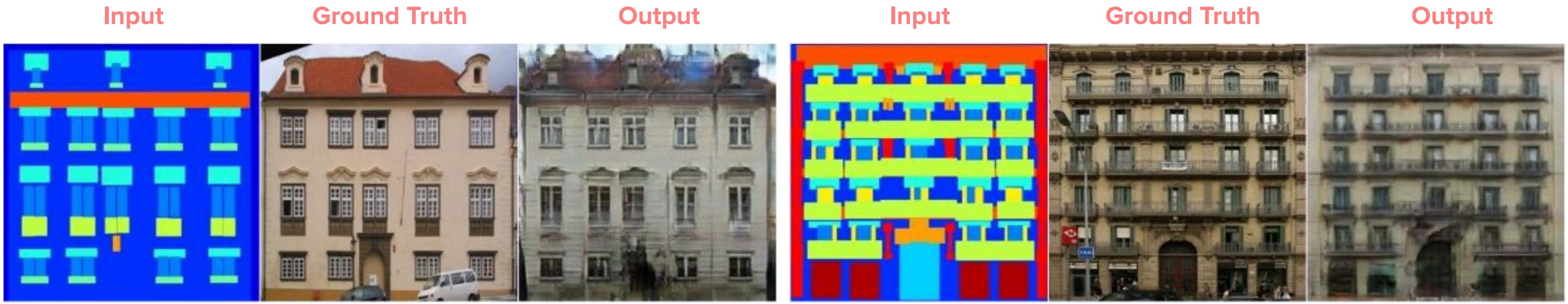
Pix2Pix Applications

- Pix2Pix has been applied in translation domains beyond that of edges to RGB images (but always following the same training strategy).
- Here, you can have the generator generate aerial photos from a map or maps from aerial photos.

Pix2Pix Applications

- In a similar way, Pix2Pix was used to generate new building facades according to a image of facade labels, i.e., positions of windows, doors, roofs, etc.

Facade labels to Photo



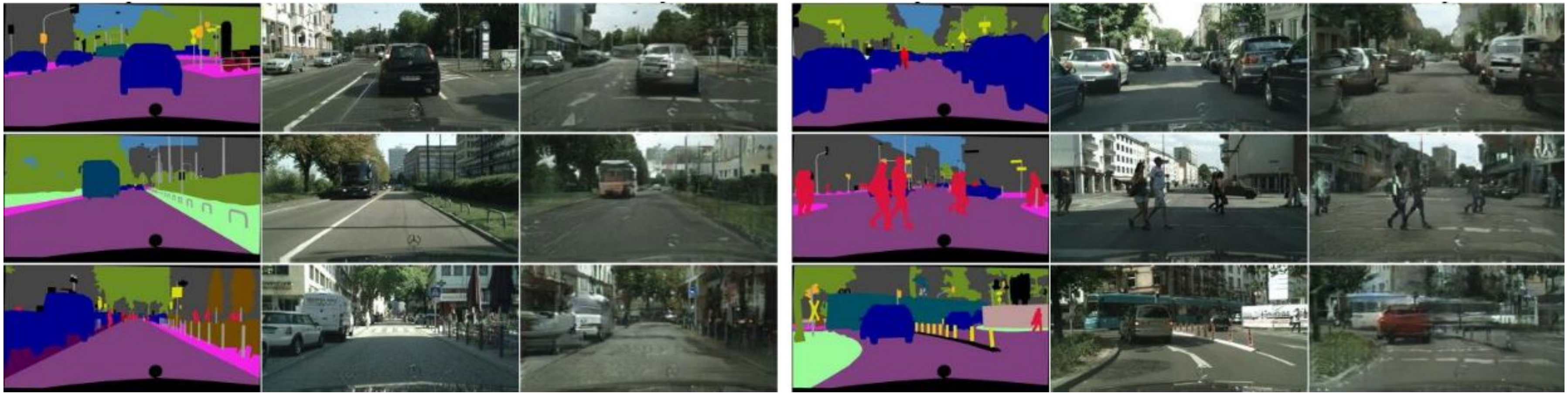
- You can actually try out some of these algorithms yourself! In this link, you'll find the edge to image and the facade labels to image applications.

Pix2Pix Applications

- Pix2Pix can be applied to image generation conditioned on a given semantic segmentation.

Segmentation to Photo

Input	Ground Truth	Output	Input	Ground Truth	Output



Pix2Pix Applications

- The principles of Pix2Pix have also been applied to many artistic endeavors.

Learning to see work piece

GauGAN art generator

Pix2Pix Applications

Day Image to Night Image

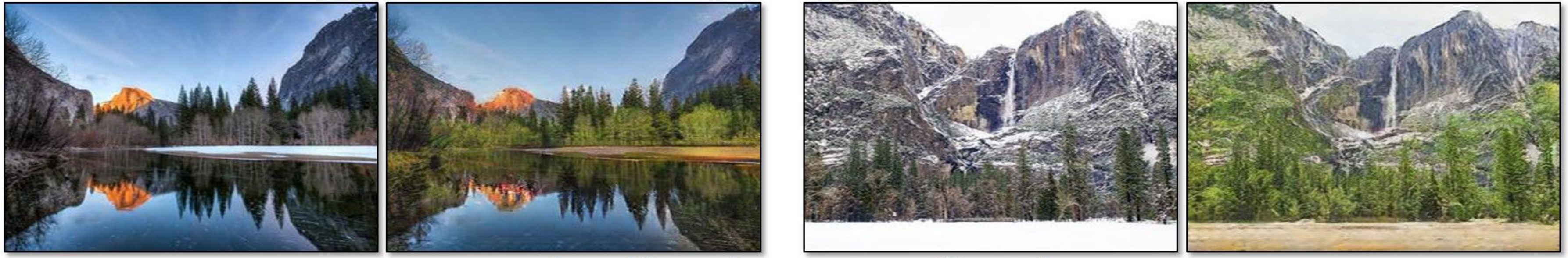
Input	Ground Truth	Output	Input	Ground Truth	Output

Thermal Image to Photo

Input	Ground Truth	Output	Input	Ground Truth	Output

Pix2Pix Applications

Season changer



winter Yosemite → summer Yosemite

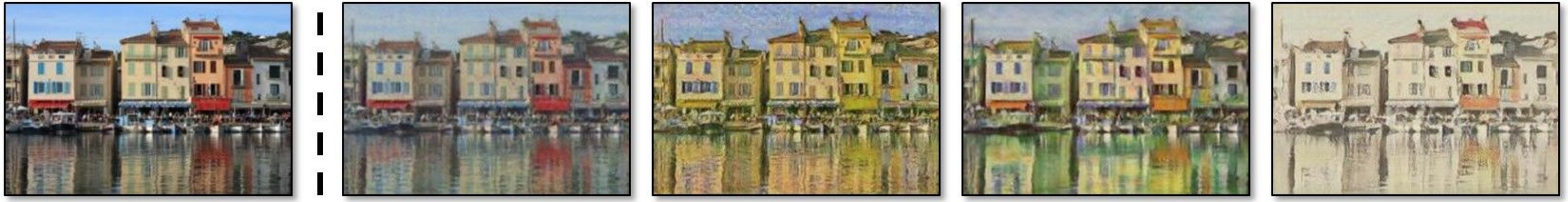
summer Yosemite → winter Yosemite

Pix2Pix Applications

Photo Enhancement (post-hoc focusing) and Painting Style Transfer

Input Output Input Output Input Output

Input Monet Van Gogh Cezanne Ukyio-e



Exercise (*in pairs*)

- Play with Pix2Pix! You can go to this link and try out some of their algorithms. What do your notice when you play with it?

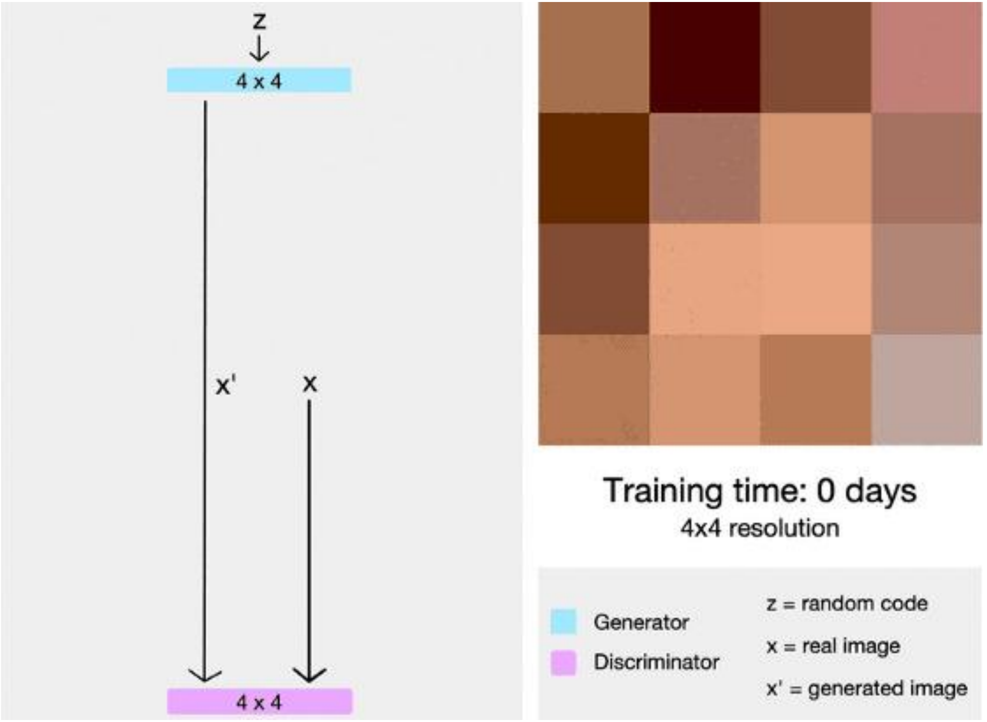
Getting high resolution images

- We saw that using GANs we can generate small images in various settings, but how can one generate **high resolution images** (images that contain fine level visual features)?
- Standard GANs could work here, but they would not be practical to generate high quality images (1024×1024 size) because of their architecture limitations.
- The first attempt to solve this issue was proposed in 2017 and was called **ProGAN (Progressive Generative Adversarial Networks)**.

Generated face using ProGAN.

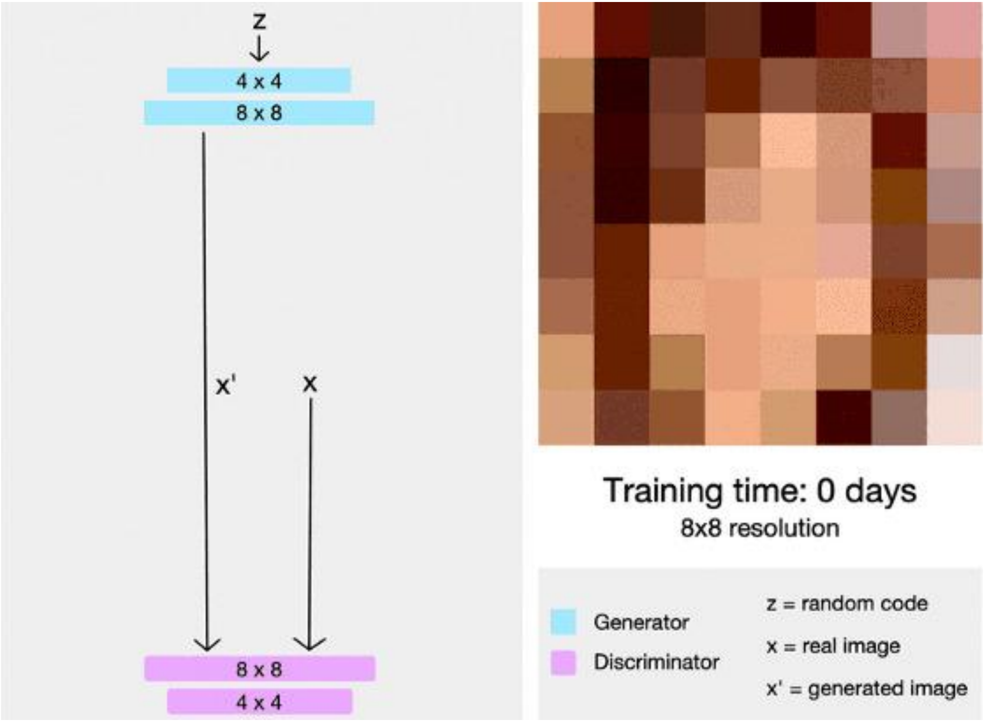
ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



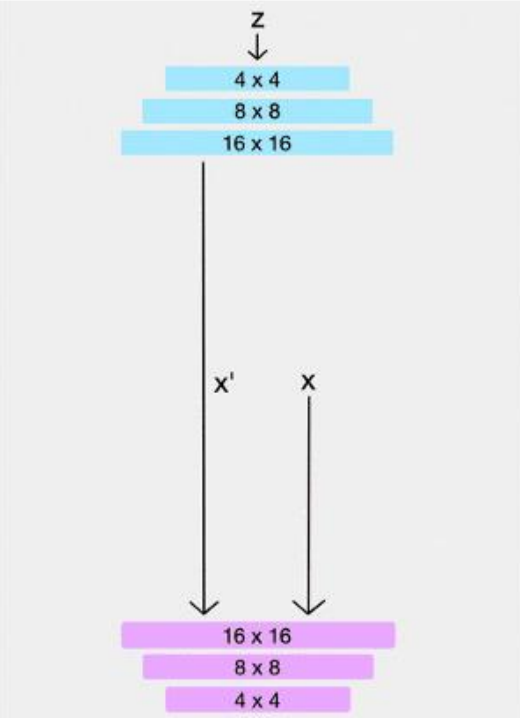
ProGAN

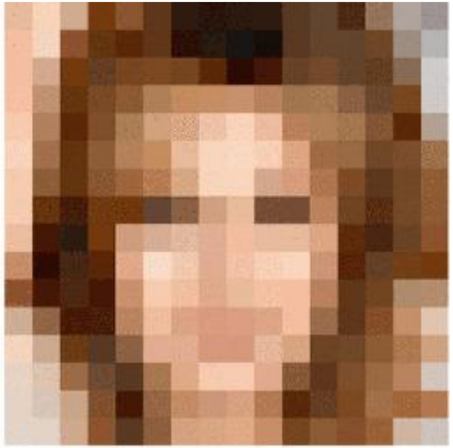
- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.

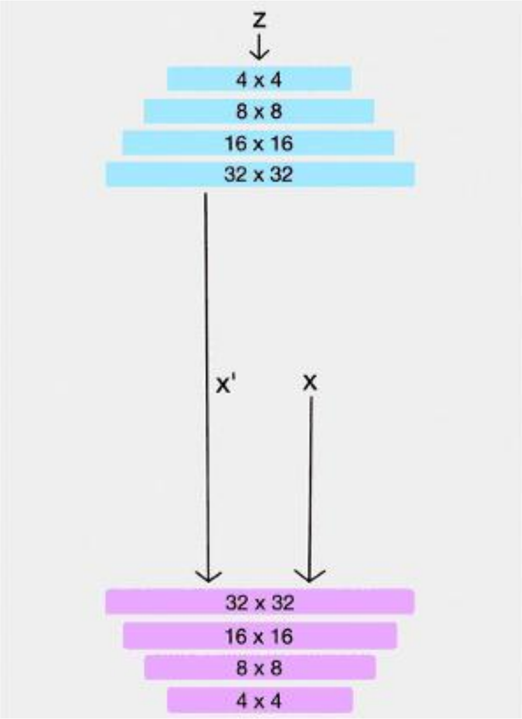


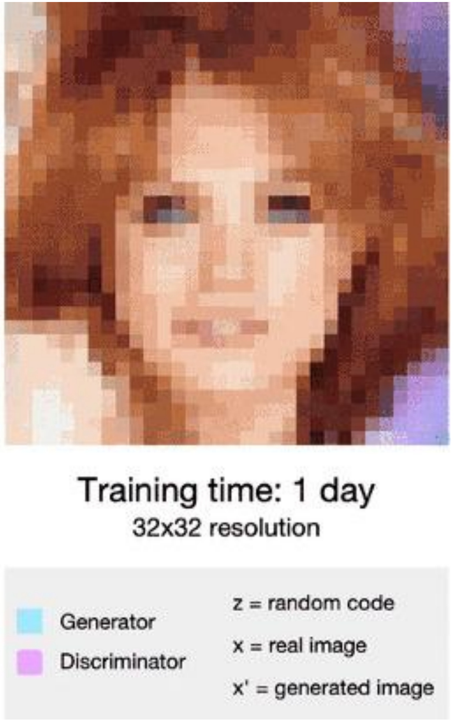


Training time: 0 days
16x16 resolution

ProGAN

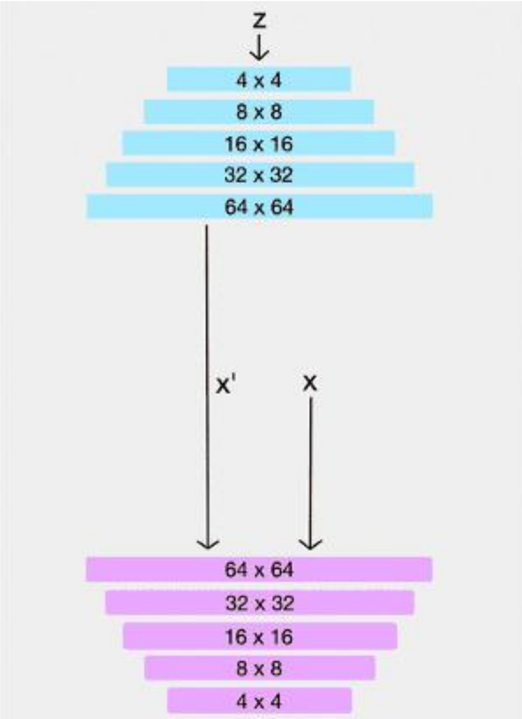
- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.





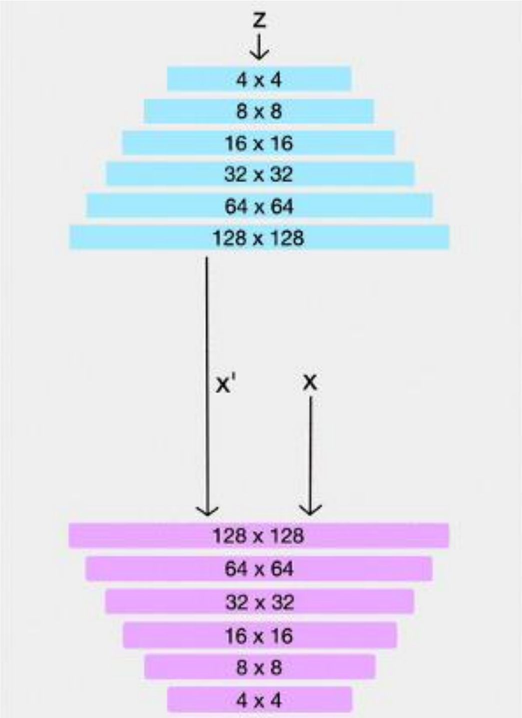
ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



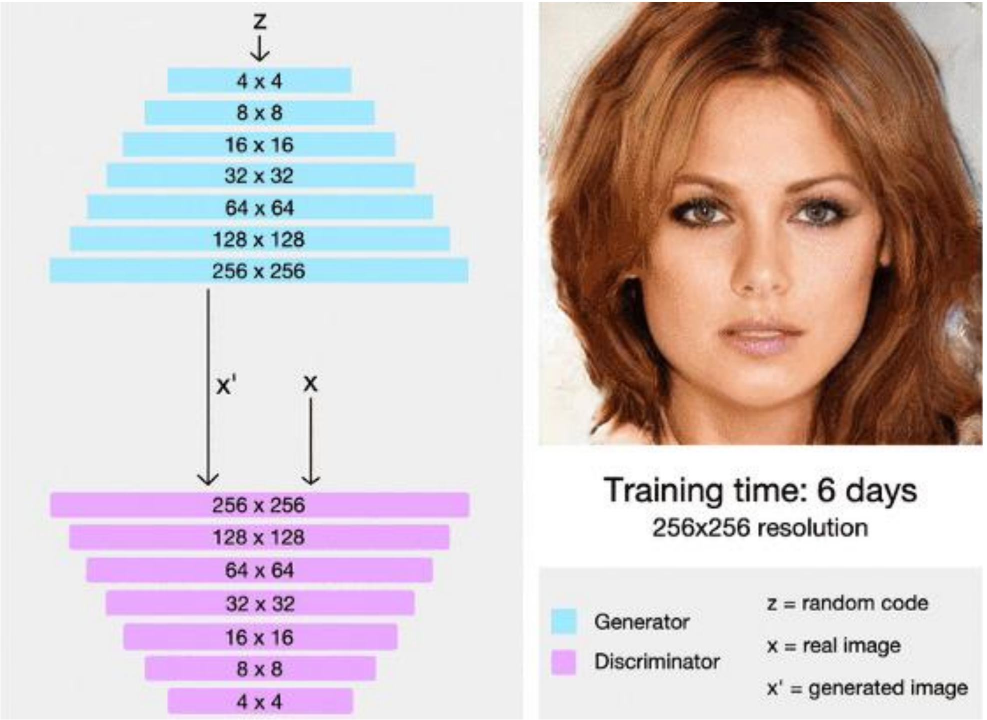
ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



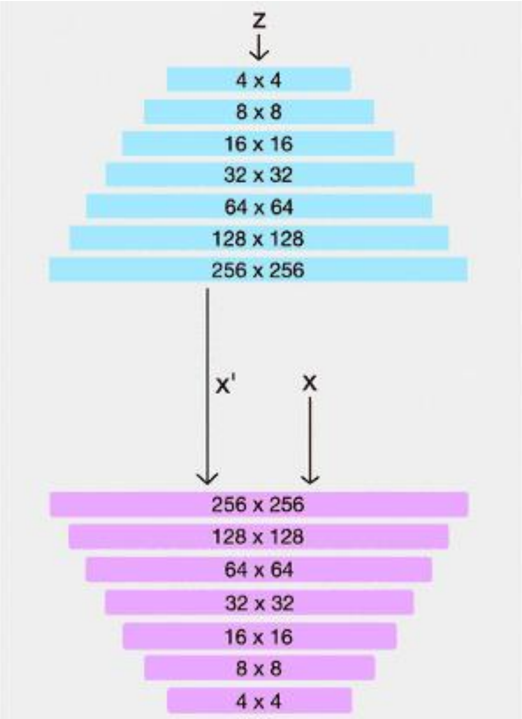
ProGAN

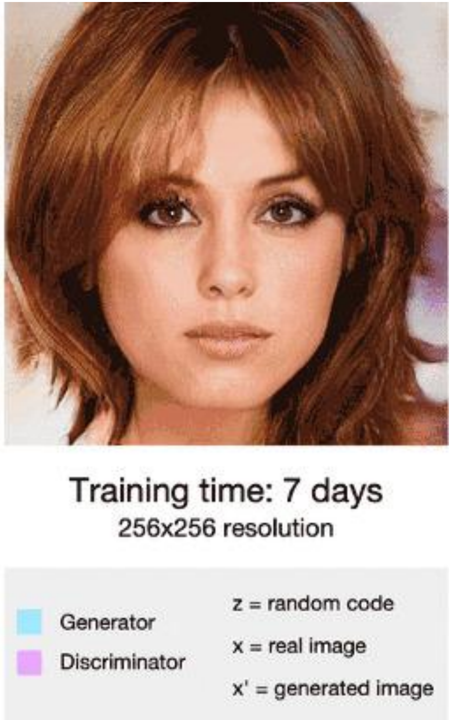
- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



ProGAN

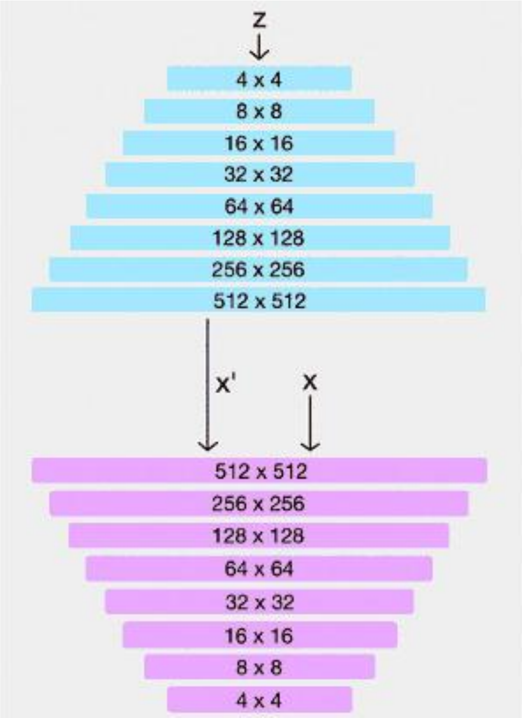
- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.





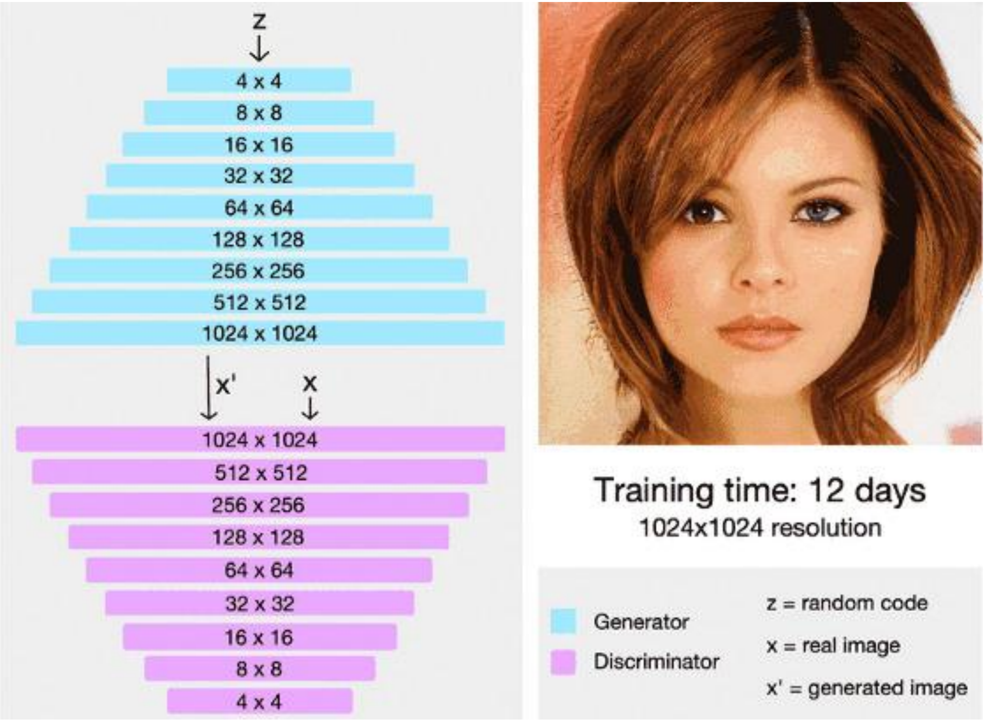
ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.



ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.

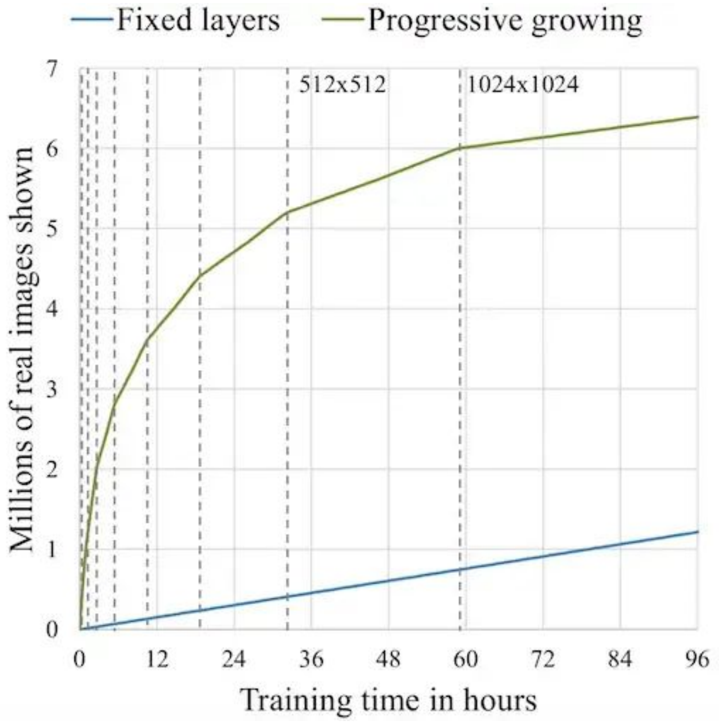


ProGAN

- ProGAN is based on an efficient way (in terms of training time) to train a GAN for High Res images.
- Instead of attempting to train all layers of the generator and discriminator at once, ProGAN trains them one layer at a time, to learn progressively higher resolution versions of the images.
- When the images generated in given resolution are good enough, we proceed to the next resolution.

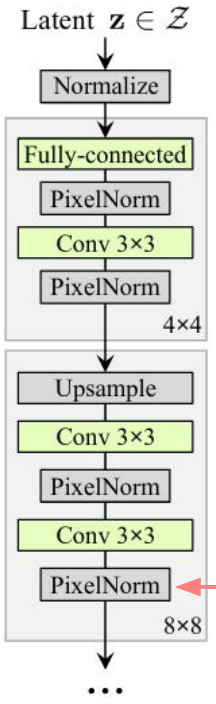
ProGAN

- The progressive growth in ProGAN time allowed the training on much bigger datasets of very large images in a much quicker time compared to when the layers were fixed.
- Although ProGAN expanded vanilla GANs ability to generate high-res images, still lacked the control over the **styling** of the output.
- This means that we couldn't change specific features such pose, face shape and hairstyle in a generated image from ProGAN.
- Considering this issue, the same ProGAN authors proposed **StyleGAN** in 2018.



StyleGAN

- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.

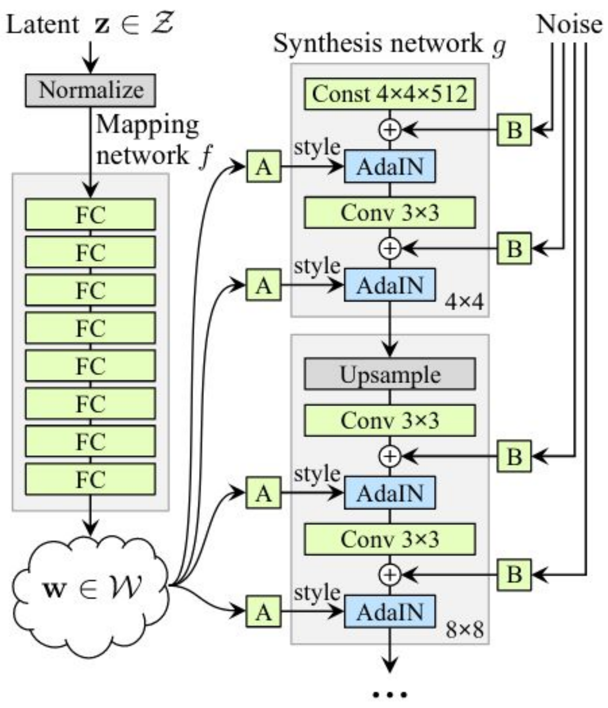


These "PixelNorm" layer is similar to Batch Normalization: It normalizes the feature vector in each pixel to unit length, and is applied after the convolutional layers in the generator. This is done to prevent signal magnitudes from spiraling out of control during training.

Generator in ProGAN

StyleGAN

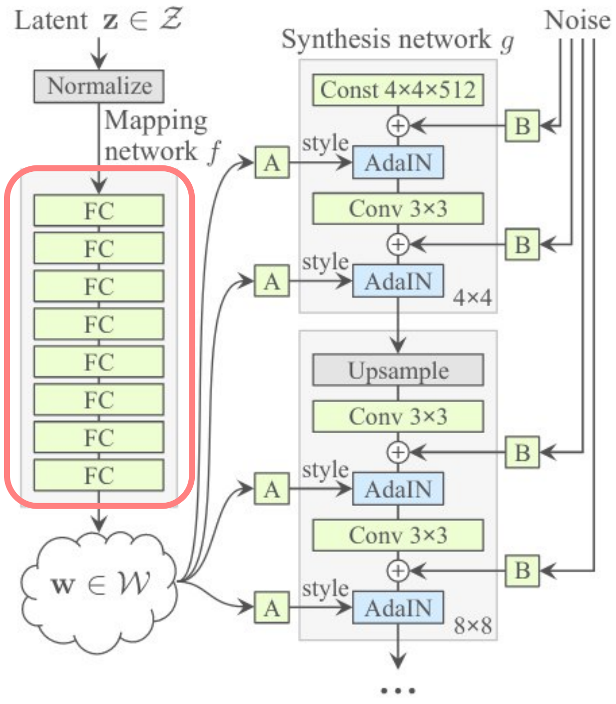
- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.



Generator in StyleGAN.

StyleGAN

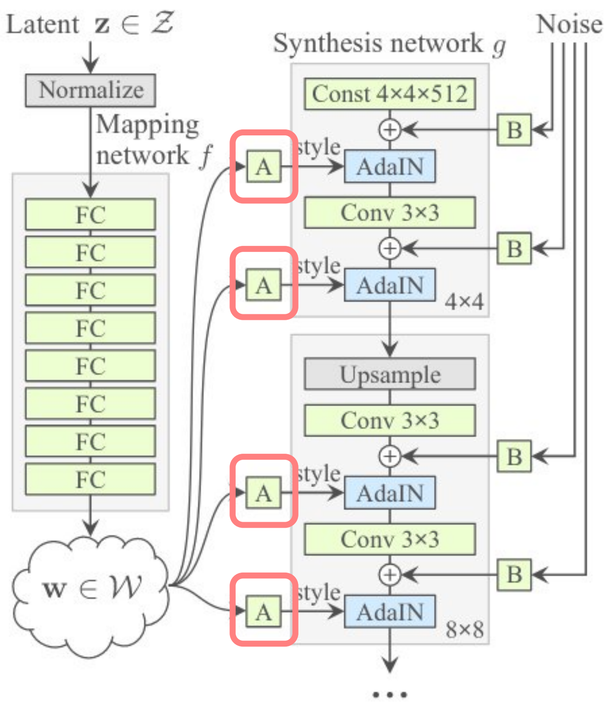
- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.
- The new generator has the following novelties:
  - The latent vector z is first transformed into what is called a vector $w = f(z)$ via a mapping network f.



Generator in StyleGAN.

StyleGAN

- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.
- The new generator has the following novelties:
  - The latent vector z is first transformed into what is called a vector $w = f(z)$ via a mapping network f.
  - w is sent to MLPs (named "A" on the right, two per resolution level) that output the **style** $y = (y_s, y_b)$.

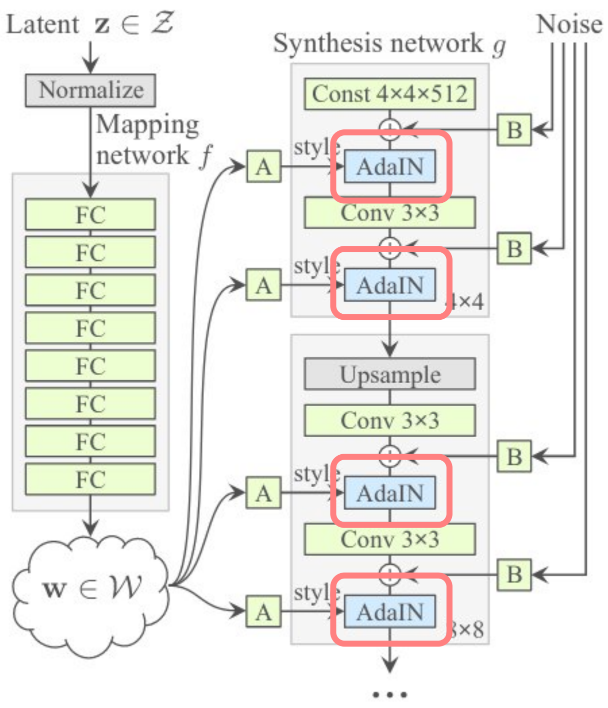


Generator in StyleGAN.

StyleGAN

- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.
- The new generator has the following novelties:
  - The latent vector z is first transformed into what is called a vector $w = f(z)$ via a mapping network f.
  - w is sent to MLPs (named "A" on the right, two per resolution level) that output the **style** $y = (y_s, y_b)$.
  - On the **synthesis network** g, a learned constant tensor gets sequentially mixed with each level's style y via an AdaIN operation in order to generate a full size image.

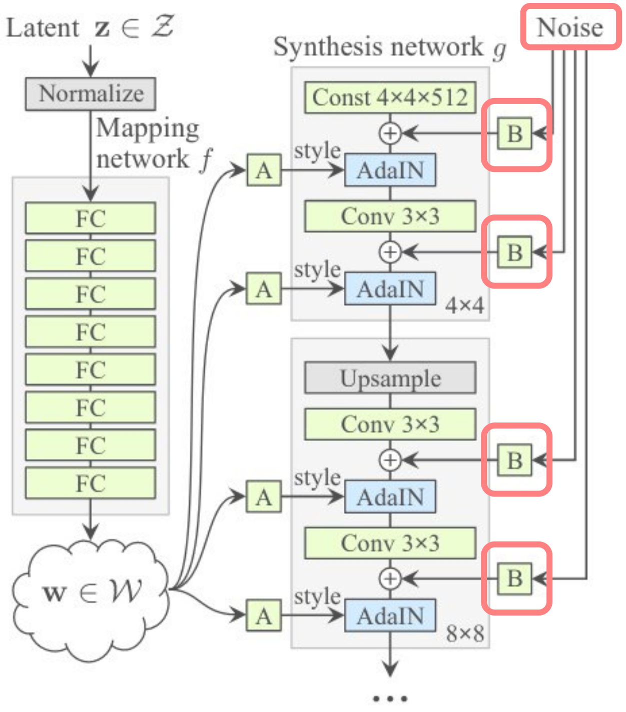
$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i},$$



Generator in StyleGAN.

StyleGAN

- StyleGAN mainly improves upon the existing architecture of Generator network to achieve the desired results and keeps Discriminator network and everything else **untouched**.
- The new generator has the following novelties:
  - The latent vector z is first transformed into what is called a vector $w = f(z)$ via a mapping network f.
  - w is sent to MLPs (named "A" on the right, two per resolution level) that output the **style** $y = (y_s, y_b)$.
  - On the **synthesis network** g, a learned constant tensor gets sequentially mixed with each level's style y via an AdaIN operation in order to generate a full size image.
  - Finally, **noise** is inserted into g via some MLPs (the "B"s) to introduce style variation at a given level of detail.



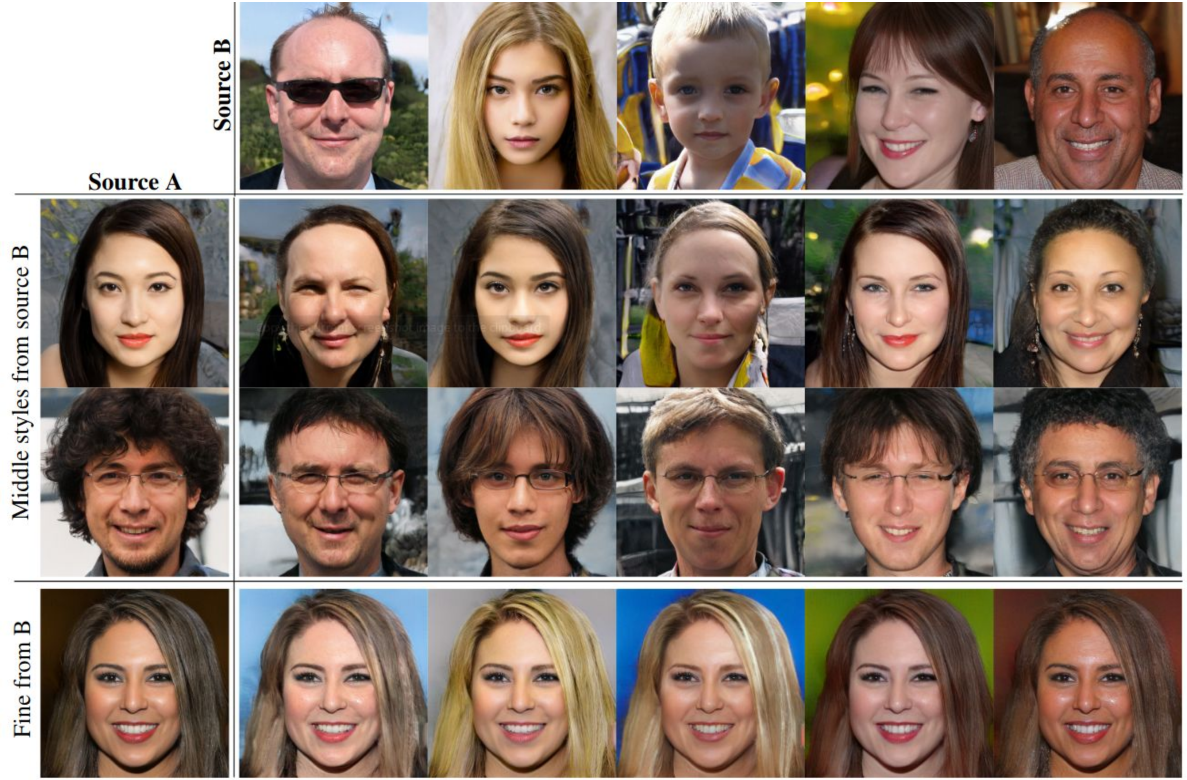
Generator in StyleGAN.

Mixing styles in StyleGAN

- With StyleGAN we can mix the styles of different generated images!
- Here, two sets of images were generated from their respective latent codes (sources A and B).
- The other images were generated by copying a subset of styles y from B and the rest from A.
- Coarse y's are those from 4^2 and 8^2 resolutions.

Mixing styles in StyleGAN

- Middle and fine y's are from resolutions 16^2 - 32^2 and 64^2 - 1024^2, resp.
- Here we note that:
  - Coarse y's correspond to high-level aspects such as general hair style pose, face shape...
  - Middle y's relate to small facial aspects, hair style, eyes open/closed.
  - Middle y's brings mainly the color scheme and microstructure.



Training of StyleGAN and other versions

- In the StyleGAN paper, the authors also introduce a new dataset of human faces called Flickr-Faces-HQ Dataset (FFHQ) consisting of 70,000 high-quality face images with which they trained their networks.

- StyleGAN was improved in a few ways in StyleGAN2 (published in 2019) and StyleGAN3 (published in 2021). Their main contributions are related to removing weird unexpected generated artifacts and make styles be learned in a more natural hierarchical manner.
- A nice thing about StyleGANs: their codes are available online (here, here and here) and many people trained them in other datasets and released the models (here and here)!

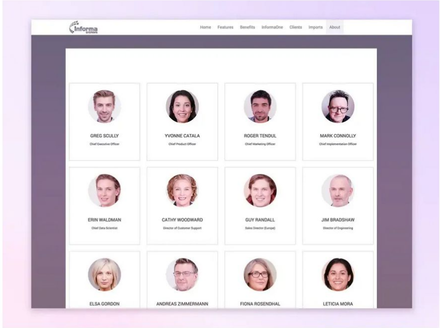
Applications of StyleGAN

- The ability to generate some many high fidelity controllable face generation has sparkled many applications (for the good and for the bad). Some of them are:

Face Interpolation

Image Editing

Well, Generate Faces (duh!)



INSIDER

That company's 'About Us' page may be full of fake pictures of 'people' who don't actually exist

Ryan Hogg and Evan Ratliff

Informa Systems is one of the companies named in an Insider report as using AI-generated images of fake "employees" on its website. informa/Insider

Exercise (*in pairs*)

- The same concept of StyleGAN has been applied to many of image domains other than faces (cats, horses, memes...). Here is a website of a collection of artificially generated images from various domains (unfortunately, some of the links are broken, here is a link for face generation). Play around with them!

Video: *AI paintings*