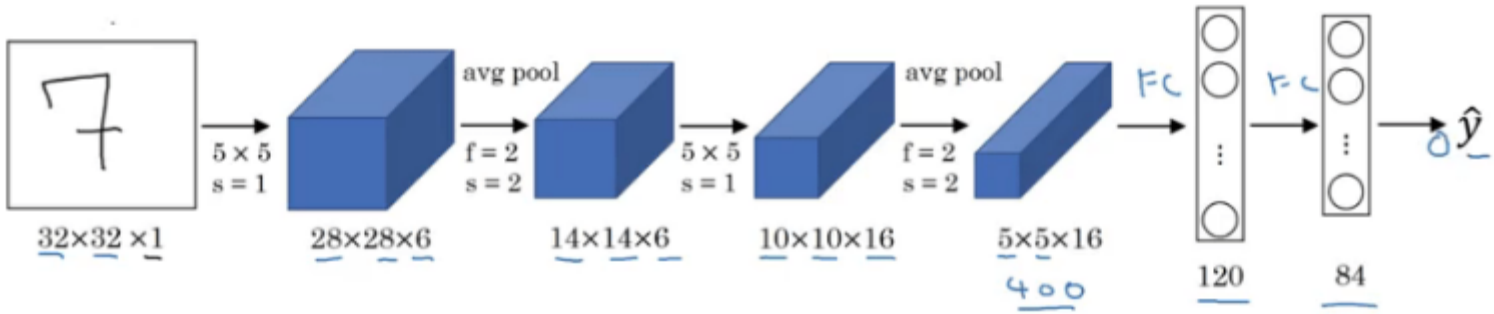
LeNet-5

Let's start with LeNet-5:

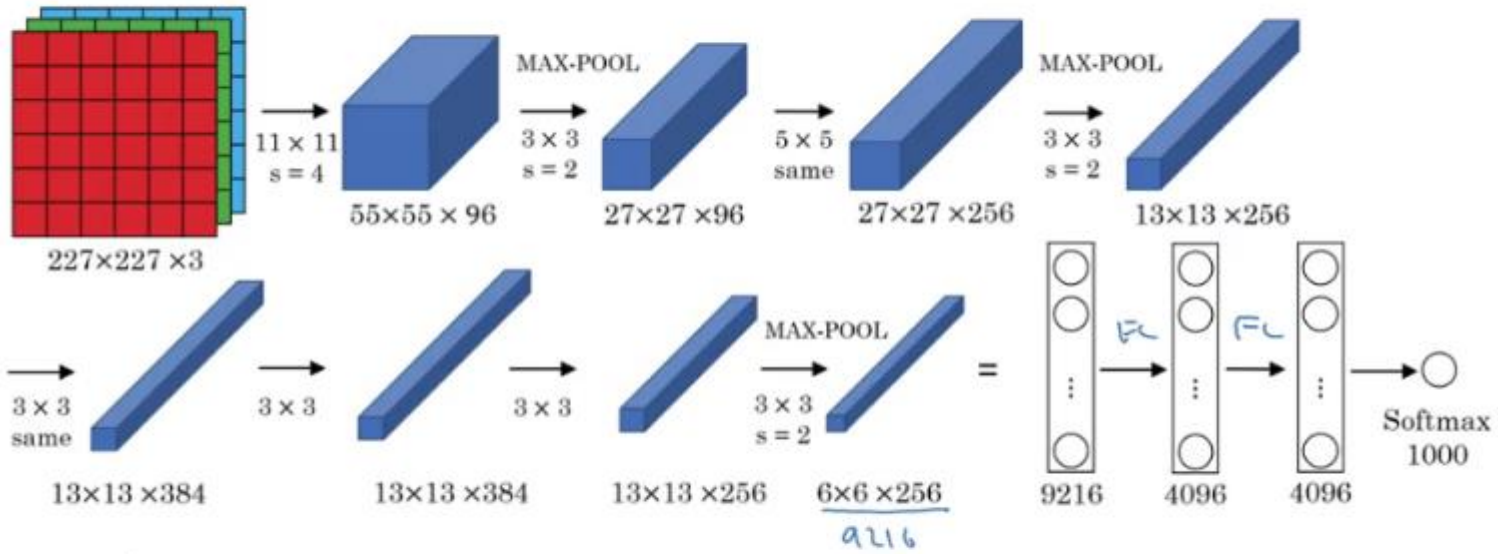


It takes a grayscale image as input. Once we pass it through a combination of convolution and pooling layers, the output will be passed through fully connected layers and classified into corresponding classes. The total number of parameters in LeNet-5 are:

- **Parameters:** 60k
- **Layers flow:** Conv -> Pool -> Conv -> Pool -> FC -> FC -> Output
- **Activation functions:** Sigmoid/tanh and ReLu

AlexNet

An illustrated summary of AlexNet is given below:



This network is similar to LeNet-5 with just more convolution and pooling layers:

- **Parameters:** 60 million
- **Activation function:** ReLu

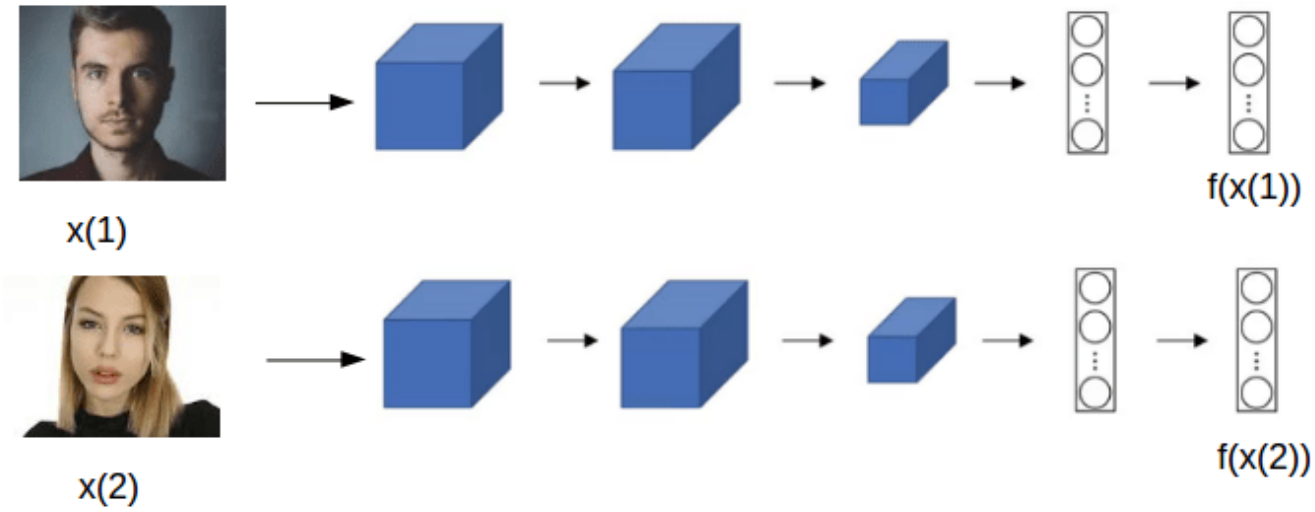
How to calculate if 2 images are of the same person using a CNN network?

Siamese Network

We will use a Siamese network to learn the function which we defined earlier:

d(img1,img2) = degree of difference between images

Suppose we have two images, x(1) and x(2), and we pass both of them to the same ConvNet. Instead of generating the classes for these images, we extract the features by removing the final softmax layer. So, the last layer will be a fully connected layer having, say 128 neurons:



Here, f(x(1)) and f(x(2)) are the encodings of images x(1) and x(2) respectively. So,

$$d(x(1),x(2)) = || f(x(1)) - f(x(2)) ||^2_2$$

We train the model in such a way that if x(i) and x(j) are images of the same person, $|| f(x(i)) - f(x(j)) ||^2$ will be small and if x(i) and x(j) are images of different people, $|| f(x(i)) - f(x(j)) ||^2$ will be large. This is the architecture of a Siamese network.

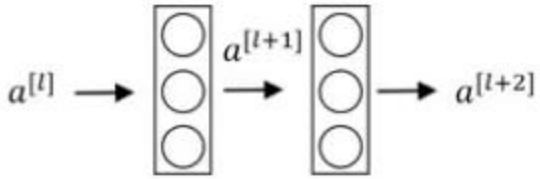
Next up, we will learn the loss function that we should use to improve a model's performance.

ResNet

Training very deep networks can lead to problems like vanishing and exploding gradients. How do we deal with these issues? We can use skip connections where we take activations from one layer and feed it to another layer that is even more deeper in the network. There are residual blocks in ResNet which help in training deeper networks.

Residual Blocks

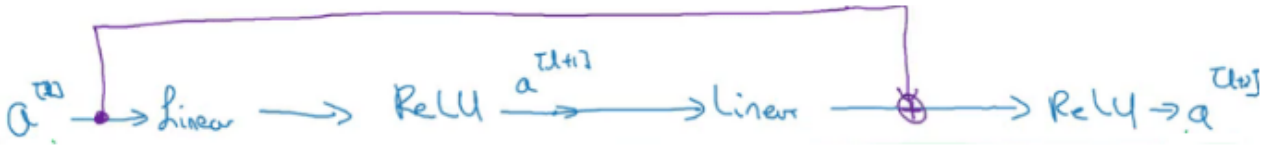
The general flow to calculate activations from different layers can be given as:



$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]}$$
$$a^{[l+1]} = g(z^{[l+1]})$$
$$z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$
$$a^{[l+2]} = g(z^{[l+2]})$$

This is how we calculate the activations $a^{[l+2]}$ using the activations $a^{[l]}$ and then $a^{[l+1]}$. $a^{[l]}$ needs to go through all these steps to generate $a^{[l+2]}$:

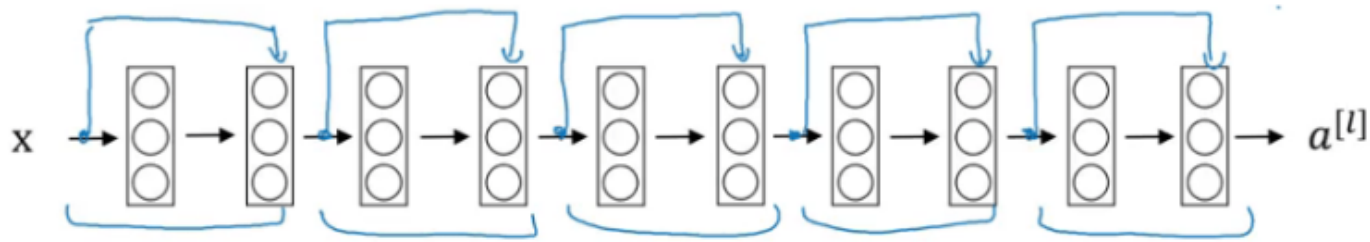
In a residual network, we make a change in this path. We take the activations $a^{[l]}$ and pass them directly to the second layer:



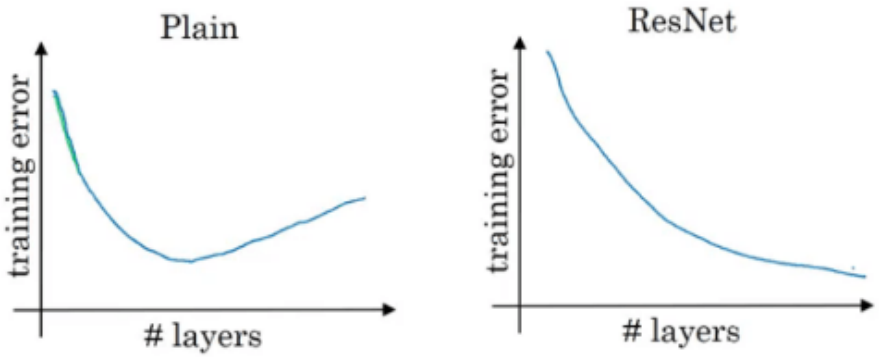
So, the activations $a^{[l+2]}$ will be:

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

The residual network can be shown as:



The benefit of training a residual network is that even if we train deeper networks, the training error does not increase. Whereas in case of a plain network, the training error first decreases as we train a deeper network and then starts to rapidly increase:



(IJACSA) International Journal of Advanced Computer Science and Applications,
Vol. 9, No. 6, 2018

Multi-Class Breast Cancer Classification using Deep Learning Convolutional Neural Network

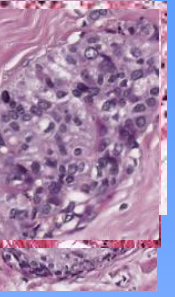
Majid Nawaz, Adel A. Sewissy, Taysir Hassan A. Soliman
Faculty of Computer and Information, Assiut University

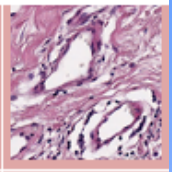
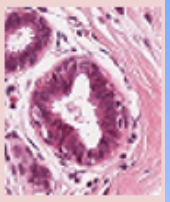
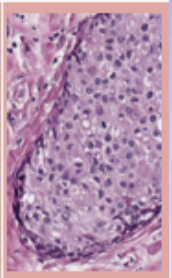
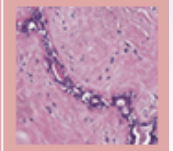
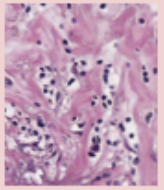
***Abstract*—Breast cancer continues to be among the leading causes of death for women and much effort has been expended in the form of screening programs for prevention. Given the exponential growth in the number of mammograms collected by these programs, computer-assisted diagnosis has become a necessity. Computer-assisted detection techniques developed to date to improve diagnosis without multiple systematic readings have not resulted in a significant improvement in performance measures. In this context, the use of automatic image processing techniques resulting from deep learning represents a promising avenue for assisting in the diagnosis of breast cancer. In this paper, we present a deep learning approach based on a Convolutional Neural Network (CNN) model for multi-class breast cancer classification. The proposed approach aims to classify the breast tumors in non-just benign or malignant but we predict the subclass of the tumors like Fibroadenoma, Lobular carcinoma, etc. Experimental results on histopathological images using the BreakHis dataset show that the DenseNet CNN model achieved high processing performances with 95.4% of accuracy in the multi-class breast cancer classification task when compared with state-of-the-art models.**

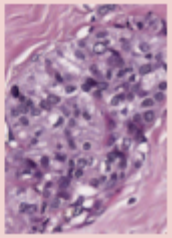
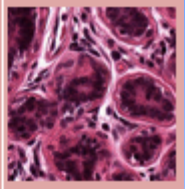
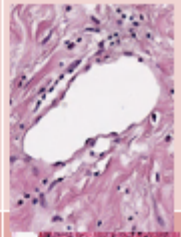
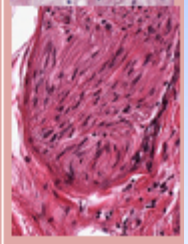
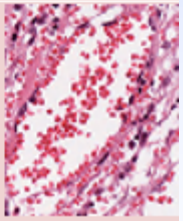
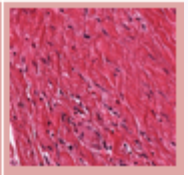
Keywords*—*Breast cancer classification; Convolutional Neural Network (CNN); deep learning; medical image processing; histopathological images

topical research topic to address. Mammographic imaging is one of the most commonly used modalities [5]. This tool that we are interested in this paper has become an indispensable tool for any clinical examination related to breast cancer. In the field of computational medical imaging, methods of deep convolutional neural networks (CNN) [6] have proved successful for the hierarchical unsupervised learning of imaging features of increasingly complex data directly from raw images, allowing to discover the relevant characteristics, instead of extracting features defined a priori by the user.

A selection of variables can be done in an integrated way with the learning of the characteristics, and this, both on the raw data and on the learned characteristics [6], [7]. Similarly, the supervised classification can also be integrated into the same architecture with the two previous steps to optimize and automate the process [8], [9]. Studies have compared the conventional multi-step computational imaging methods with deep learning methods, and showed a better classification accuracy and mortality prediction with deep learning methods in the case of screening mammograms breast cancer [8]. Deep learning refers to advanced statistical learning methods organized in multiple layers, to extract representations of data on multiple levels, and whose layers are not predefined by the user but learned directly from the data by the algorithm, thus



Area	Abbreviation	Explanation	Tissue sample
Blood vessel	BV	Either a small artery or vein. Arteries commonly have thicker walls than veins. Blood vessels are important infiltration routes for tumors.	
Duct	D	Lactiferous ducts are tubular milk transport structures in breasts. In histopathology slices lactiferous ducts look like holes or cylinders depending on the cut direction.	
Ductal carcinoma in situ	DCIS	Ductal carcinoma in situ is stands for the most common noninvasive breast cancer that originates from a lactiferous duct. DCIS is a preliminary stage of cancer where cells already seem malignant and contain genetic changes but the cells are still contained inside the duct and have not invaded the stroma.	
Fat	F	Fat looks like empty drops because the fat dissolves in the slice preparation process before the cutting and dyeing of paraffin blocks.	
Fibroadenoma	FA	Fibroadenomas are the most common breast tumors in adolescent women. They are benign tumors that are composed of glandular and fibrous breast tissue.	
Inflammatory cells	IC	Inflammatory tissue contains lymphocytes, neutrophils and eosinophils. Lymphocytes look like dark, little and round cells and their dark nucleus fills almost the whole cell body.	

Invasive ductal carci-noma	IDC	Invasive ductal carcinoma is a malignant breast cancer which originates from a lactiferous duct and invades the stroma.	
Lobule	L	Lobule (terminal duct-lobular unit (benign)) is a unit in the end of lactiferous duct from which milk is se-creted. Lobule consists of little glands, which form a round structure, and of the small distal part of the duct.	
Lymph vessel	LV	Lymph vessels are part of the lymphatic system where lymph passes through lymph nodes and returns to bloodstream. Lymph vessels have a thin wall. Lymph vessels are important infiltration routes for tumors.	
Nerve	N	Nerves are important structures in tumor diagnostics because benign changes do not normally grow near the nerves. Invasion to a nerve surrounding traditionally means that the tumor is malign, even though there are exceptions.	
Red Blood cells	RBC	Red blood cells have a biconcave disc shape. They are red cells that do not have nuclei. Red blood cells have a diameter of 5 micrometers and they are usually found in the lumen of blood vessels.	
Stroma	STROMA	Stroma consists of connective tissue surrounding and supporting biological tissues, cells and organs. Whereas parenchyma refers to the functional parts of the tissue (e.g. the actual mutant cells).	

III. Proposed CNN Model for Multi-Class Breast Cancer Classification

A Convolutional Neural Network (CNN) is feedforward neural network introduced by Kunihiko Fukushima in 1980 [30] and improved by Yann LeCan et al. in 1998 [35], [36]. A CNN is composed of 6 types of layers: an input layers, a convolutional layer, a non-linear layer, a pooling layer, fully connected layer, and an output layers. Fig. 1 illustrates a traditional CNN architecture.

Convolutional Neural Networks (CNN) are one of the most remarkable approaches of deep learning, in which multiple layers of neurons are formed in a robust manner. They have shown that they are capable of demonstrating an impressive generalization capability on large data sets with millions of images [37], [38]. These results come mainly from the particular architecture of CNNs that takes into account the specific topology of tasks related to the field of computer vision that exploit two-dimensional images. Other dimensions can also be taken into account when it comes to color images with multiple channels.

To train a CNN we determine the mapping function using the feedforward operation and we optimize the loss function using retro propagation techniques in particular, the gradient decent algorithm. The CNN that we choose for the task of breast cancer classification is not a traditional CNN model. DenseNet [12] is a CNN model which they replace convolution non-linear and pooling layers with dense blocks and transition layers using the original CNN layers except the first convolutional layer. Fig. 2 presents the original DenseNet model with three dense blocks and two transition layers.

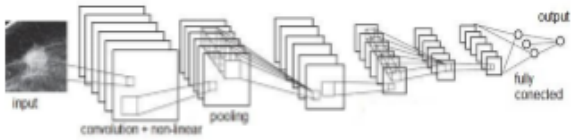


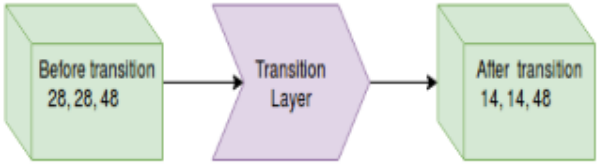
Fig. 1. Convolutional neural network.

Fig. 2. DenseNet with three dense blocks architectures [12].

The dense block proposed by DenseNet contains convolution and non-linear layers. Also, they apply some optimization techniques like dropout and batch normalization. In addition, in the dense block proposed by DenseNet, outputs from the previous layers are concatenated instead of using the summation. So, assume that an input image has the shape of (28, 28, 3), in which three represents the RGB color space. First, we spread image to initial N channels and receive the image (28, 28, N). Every next convolution layer will generate k features, and remain the same height and width. The feature concatenation process is illustrated by the Fig. 3. If we assume that we have N= 24 and K= 12 we will receive the image with same dimension, but with plenty of features (28, 28, 48).

Fig. 3. DenseNet concatenation process inside the dense block.

To reduce the size, DenseNet uses transition layers. These layers contain convolution with kernel size = 1 followed by 2x2 average pooling with stride = 2. It reduces height and width dimensions but leaves feature dimension the same. The transition layer is presented in Fig. 4. As a result, if the input is an image with shape (28, 28, 48), we receive an output image with shapes (14, 14, 48).The DenseNet scale naturally to hundreds of layers, while exhibiting no optimization difficulties. Thus, that makes DenseNet one of the most powerful models in image recognition tasks.

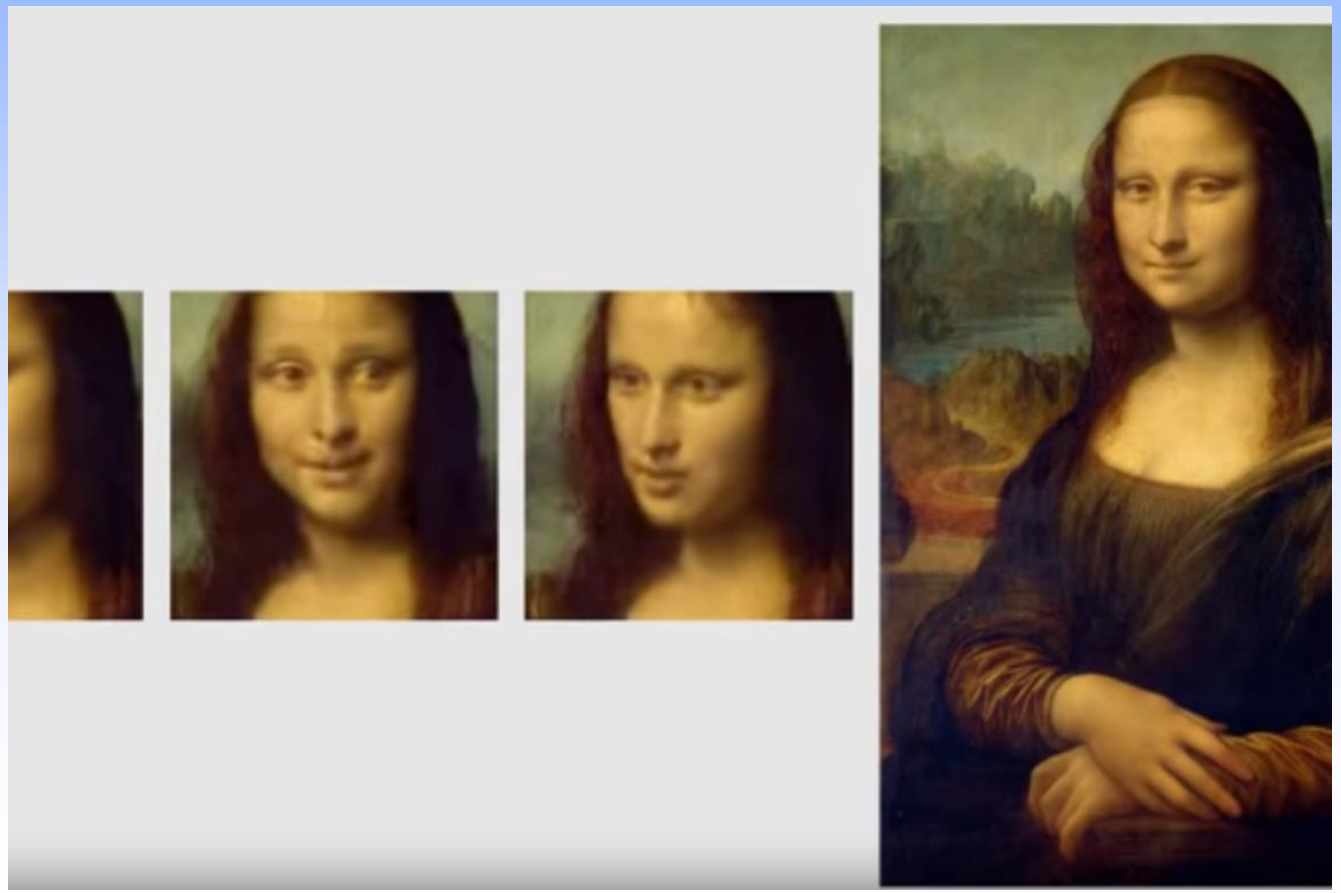


Transition layer

TABLE III. COMPARISON WITH SOME POPULAR CNNS IN THE MULTI-CLASS BREAST CANCER CLASSIFICATION

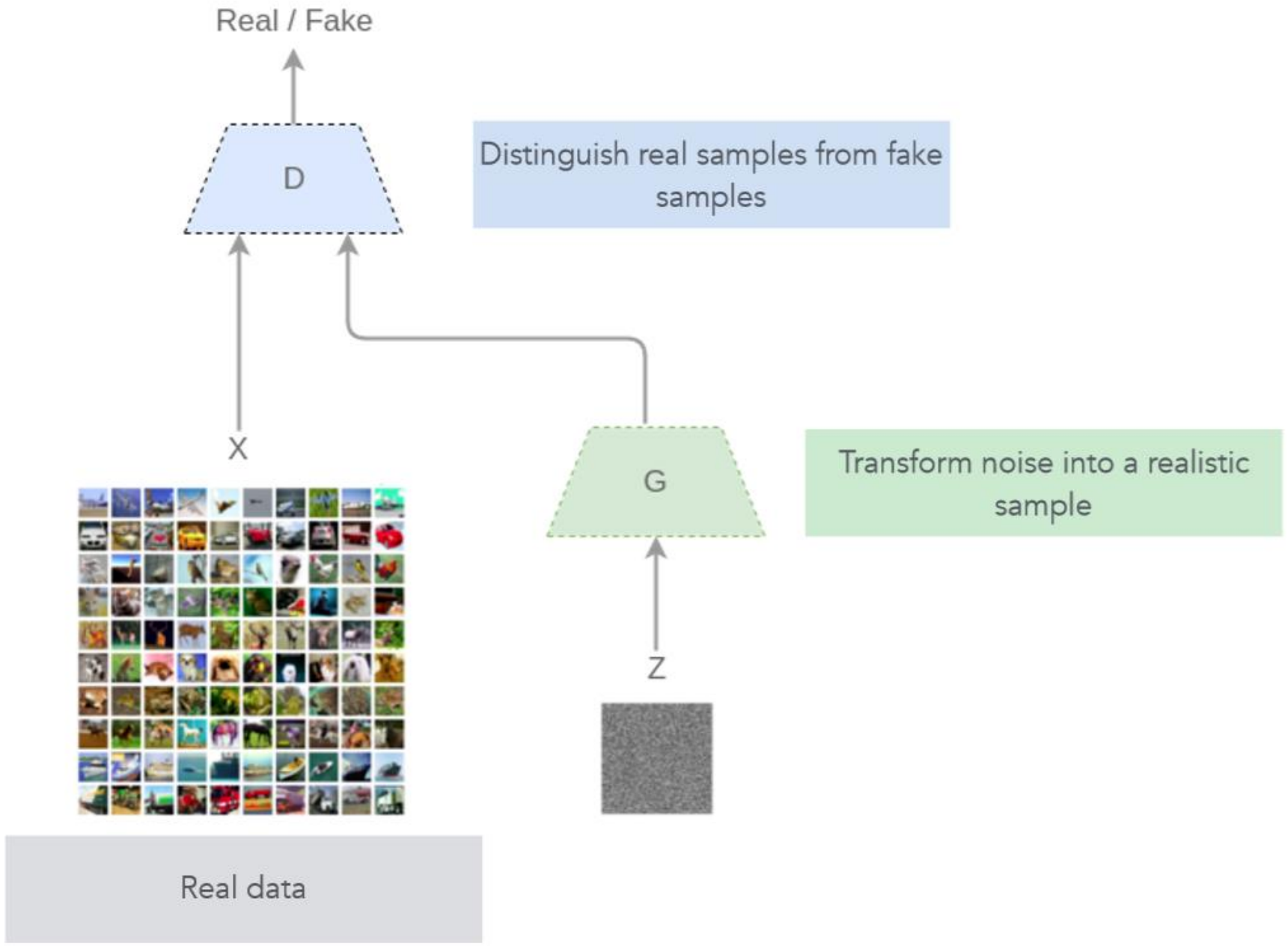
Accuracy (%)	**Model**	**Magnification Factors**				
		40x	***100x***	***200x***	***400x***	***average***
Image level	***LeNet [43]***	46.4	47.34	46.5	45.2	46.36
	AlexNet [28]	86.4	75.8	72.6	84.6	79.85
	CSDCNN[30]	92.8	93.9	93.4	92.9	93.25
	DenseNet (our's)	93.64	97.42	95.87	94.67	95.4
Patient level	***LeNet [43]***	48.2	47.6	45.5	45.2	46.62
	AlexNet [28]	74.6	73.8	76.4	79.2	76
	CSDCNN[30]	94.1	93.2	94.7	93.5	93.87
	DenseNet (our's)	94.23	97.86	96.35	95.24	96.48

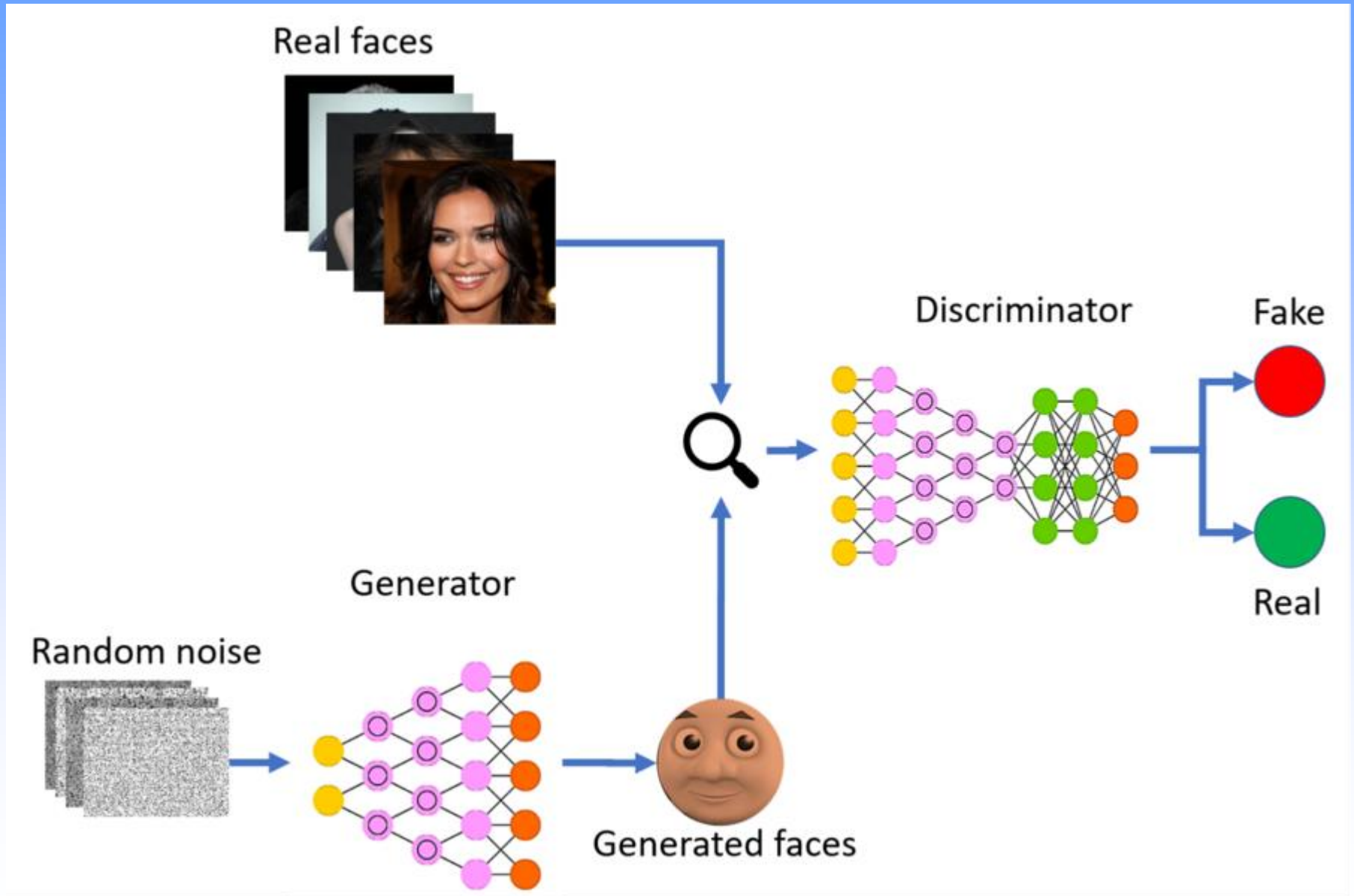
Generative Adversarial Networks (GAN)



What are GANs?

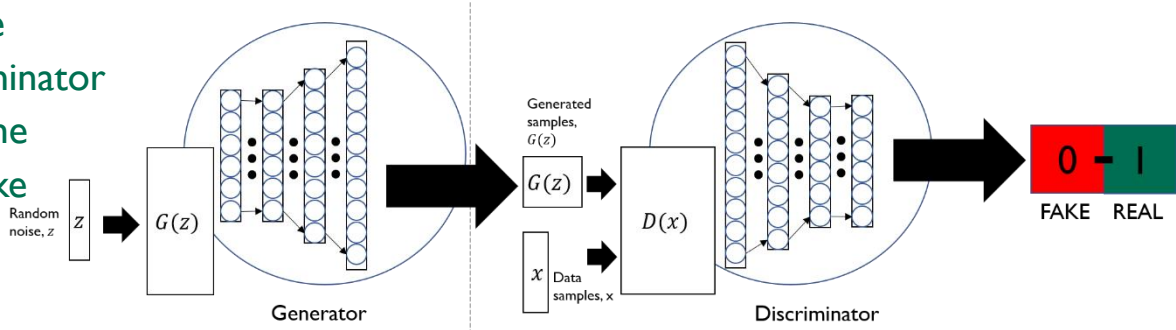
- System of two neural networks competing against each other in a zero-sum game framework.
- They were first introduced by Ian Goodfellow *et al.* in 2014.
- Can learn to draw samples from a model that is similar to data that we give them.


Real / Fake
D
Distinguish real samples from fake samples
X
G
Transform noise into a realistic sample
Z
Real data


Real faces
Discriminator
Fake
Real
Generator
Random noise
Generated faces

Generative Adversarial Networks

- Original paper: Generative Adversarial Nets by Goodfellow et al.
  - Team from the University of Montreal
  - Presented at NIPS in 2014
- Two deep neural networks: Generator and Discriminator
- Uses "adversarial nets" framework
  - Trained to play a minimax game
  - Generator tries to fool Discriminator
  - Discriminator tries to determine whether samples are real or fake



The Generator

- The Generator's job is to fool the Discriminator
  - Produce fake samples, $G(z)$, from noise, z, the Discriminator, $D(x)$ can't distinguish from real samples, x
- During training:
  - Input is random noise samples z from probability distribution, p_z
  - Use SGD or other gradient update algorithm like Adam to update parameters, θ_G with $\nabla_{\theta_G} J_G(W, b)$

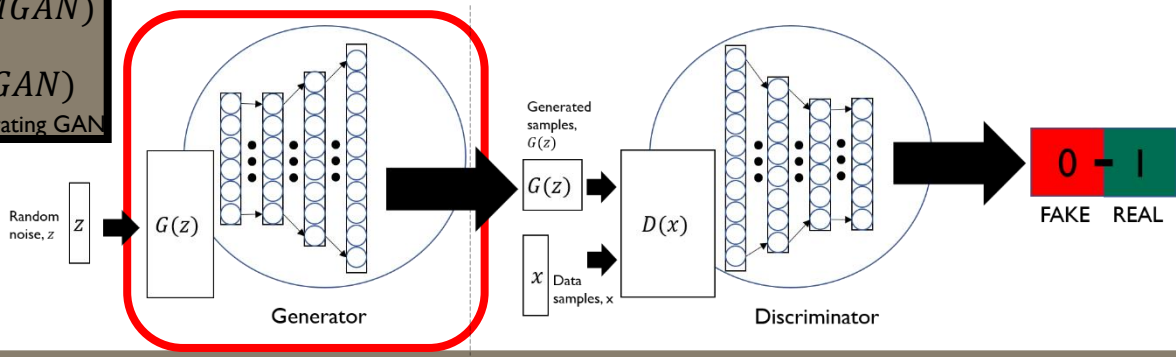
$$J_G(W,b) = \frac{1}{m}\sum_{i=1}^{m}\begin{cases}\log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) & (MGAN) \\ -\log\left(D\left(G\left(z^{(i)}\right)\right)\right) & (NSGAN)\end{cases}$$

Minimax GAN

Non-Saturating GAN

- Outputs fake samples, $G(z)$ with probability distribution p_g

In practice Minimax GAN saturates quickly, Non-Saturating GAN alleviates that problem



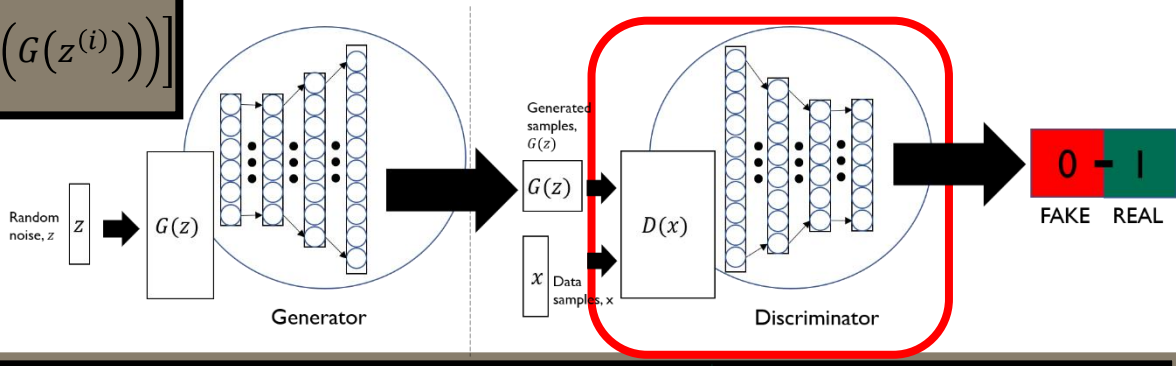
Reference: Generative Adversarial Nets

The Discriminator

- The Discriminator's job is detect fake samples
  - Determine probability input is real or fake
- During training:
  - Input is both fake samples, $G(z)$, and real samples, x, from data probability distribution, p_{data}
  - Update parameters θ_D by gradient ascent with $\nabla_{\theta_D} J_D(W, b)$

$$J_D(W,b) = \frac{1}{m}\sum_{i=1}^{m}\left[\log\left(D\left(x^{(i)}\right)\right) - \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right]$$

- Output is probability $D(x)$



Reference: Generative Adversarial Nets

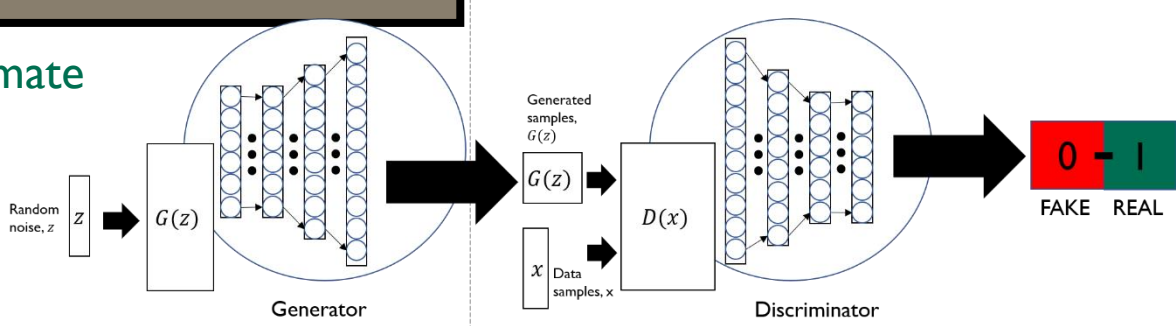
The Game GANs Play

- The Generator wants $D(G(z))$ to be high
- The Discriminator wants $D(G(z))$ to be low
- Generator and Discriminator play minimax game with value function $V(D, G)$:

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

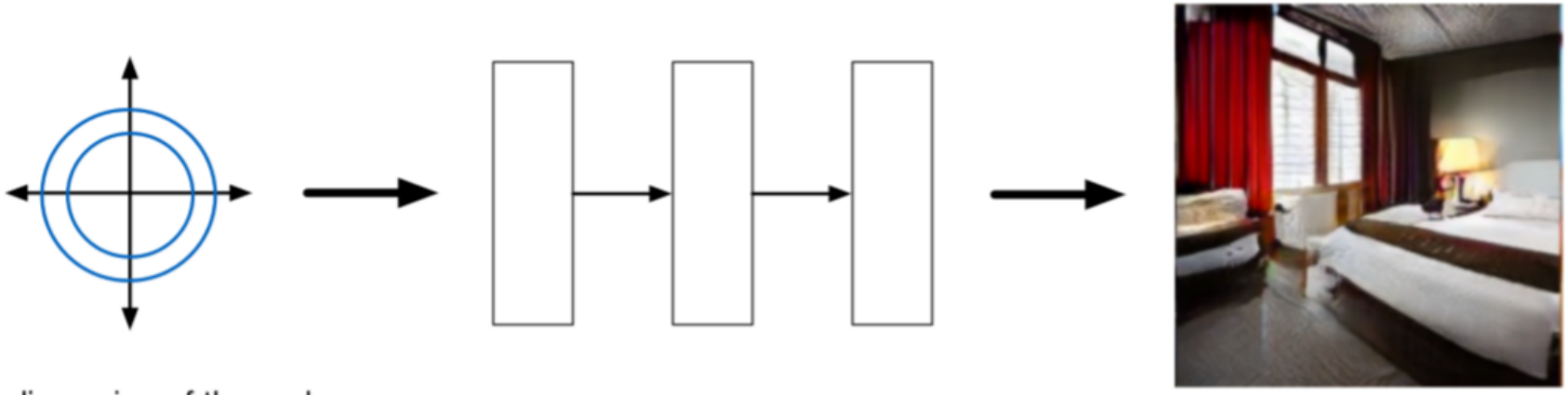
- GANs are trained to approximate this relationship:

$$p_g(z) = p_{data}(x)$$



Reference: Generative Adversarial Nets

Implicit Generative Models



Each dimension of the code vector is sampled independently from a simple distribution, e.g. Gaussian or uniform.

This is fed to a (deterministic) generator network.

The network outputs an image.

This sort of architecture sounded preposterous to many of us, but amazingly, it works.

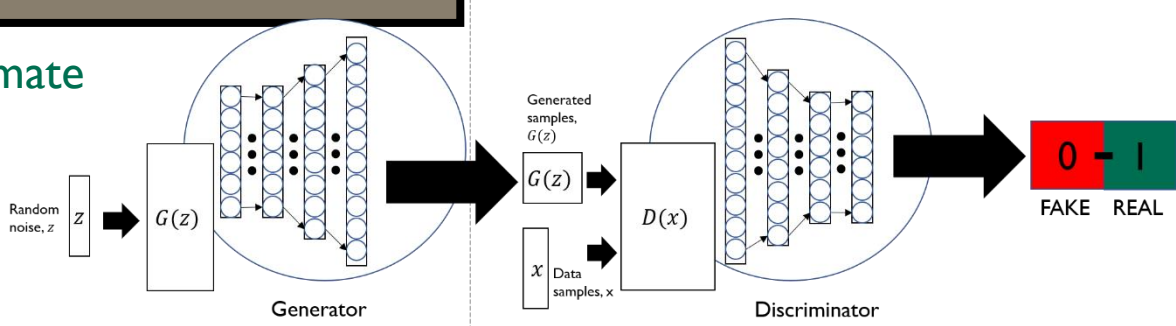
The Game GANs Play

- The Generator wants $D(G(z))$ to be high
- The Discriminator wants $D(G(z))$ to be low
- Generator and Discriminator play minimax game with value function $V(D, G)$:

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

- GANs are trained to approximate this relationship:

$$p_g(z) = p_{data}(x)$$



Reference: Generative Adversarial Nets

The Algorithm

for # of training epochs **do**

Reference: Generative Adversarial Nets

for *k* steps **do**

- Sample minibatch of *m* noise samples $\{z^{(1)}, z^{(2)}, \dots, z^{m}\}$ from distribution $p_z(z)$
- Sample minibatch of *m* data examples $\{x^{(1)}, x^{(2)}, \dots, x^{m}\}$ from distribution $p_{data}(x)$
- Update the discriminator through gradient ascent:

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

end **for**

- Sample minibatch of *m* noise samples $\{z^{(1)}, z^{(2)}, \dots, z^{m}\}$ from distribution $p_z(z)$
- Update the generator through gradient descent:

$$\nabla_{\theta_G}\left[\frac{1}{m}\sum_{i=1}^{m}\begin{cases}\log\left(1-D\left(G\left(z^{(i)}\right)\right)\right) & (MGAN)\\ -\log\left(D\left(G\left(z^{(i)}\right)\right)\right) & (NSGAN)\end{cases}\right]$$

end **for**

k is a hyperparameter

Updating the Discriminator before the Generator necessary to avoid Mode Collapse

Gradient updates can be done through any gradient-based learning rule

Generative Adversarial Networks

- Pros:
  - State-of-the-art Image Generation
  - Gradients calculated through Backpropagation
- Cons:
  - No explicit representation of generator distribution, $p_g(x)$
  - G and D need to be synchronously trained to avoid Mode Collapse
    - Mode Collapse happens when the Generator finds a "crack" in the Discriminator's armor and continues to attack the weakness
      - Produces similar outputs with little variation between examples or between features in examples
      - Unstable/Difficult to train

Generative Adversarial Networks

GAN trained on MNIST

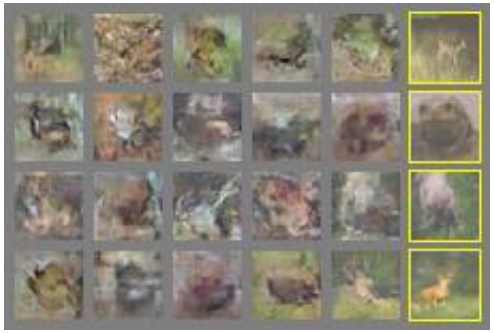
The five frames on the left of each box are generated examples. The yellow frame is the nearest training example to the sample on its left.

Reference: [Generative Adversarial Nets]

GAN trained on TFD

GAN trained on CIFAR-10 with fully connected architecture

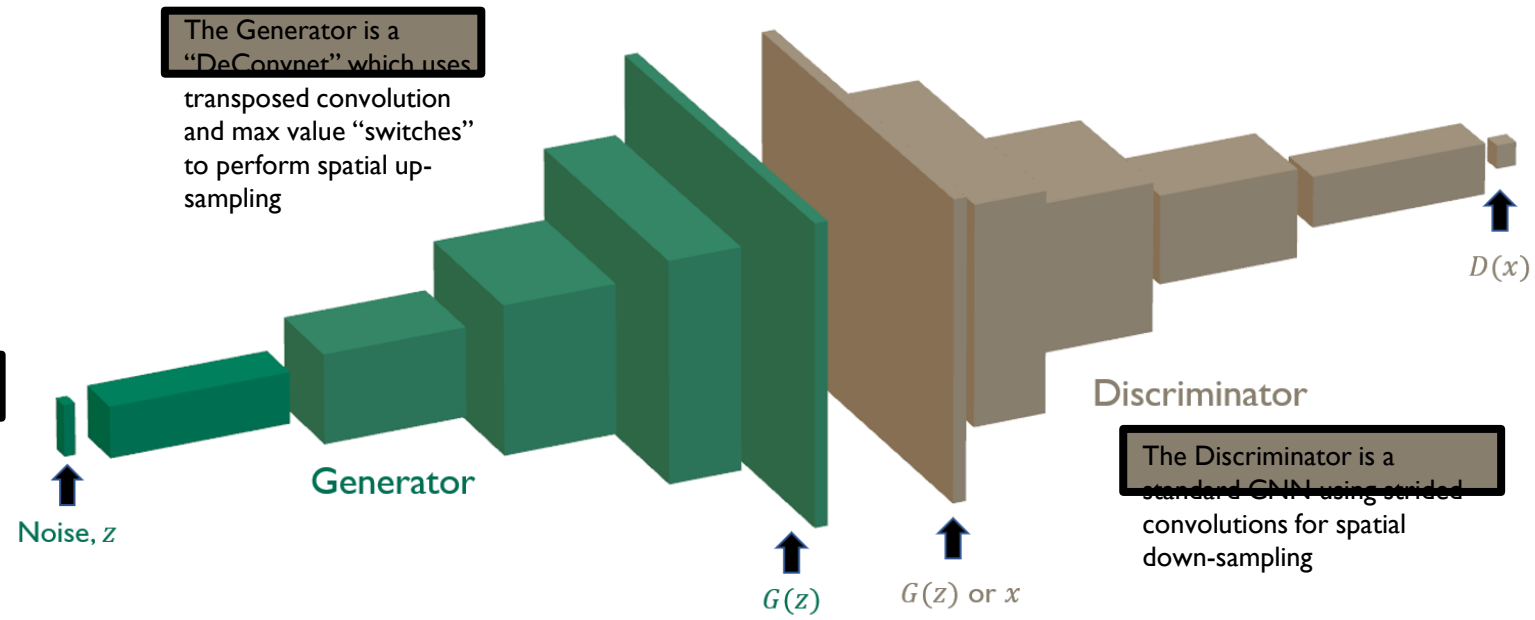
GAN trained on CIFAR-10 with Convolutional Discriminator And "Deconvolutional" Generator



Upgrade: DCGAN

- Original paper: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks by Alec Radford, Luke Metz (indigo Research) and Soumith Chintala (Facebook AI Research)
  - Conference paper at ICLR 2016
- Contributions:
  - Deep Convolutional Generative Adversarial Network (DCGAN) architecture
    - Originally proposed in Generative Adversarial Nets but architecture wasn't discussed
    - Continued GAN training optimization
  - Visualized what kernels/filters learned
  - Showed Discriminator to be near state-of-the-art classifier
  - Showed interesting vector arithmetic properties of the Generator

DCGAN: Architecture



Vector Arithmetic is done with Generated example's representation (embedding) in the z space

Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN: Convolutional Layers

- DCGAN – CNN architecture specs: (USES ORIGINAL TRAINING ALGORITHM)
  1. Use "All-Convolutional" Nets: No pooling operations for (down/up) sampling
  2. Eliminate fully-connected layers at the end of convolutional layers
  3. Apply Batch Normalization
     - DON'T apply to Generator output layer or Discriminator input layer
  4. Activation Functions
     - Uses ReLU throughout Generator except Tanh at output
     - Use Leaky ReLU throughout Discriminator (especially for high-res applications)

Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN for Classification?

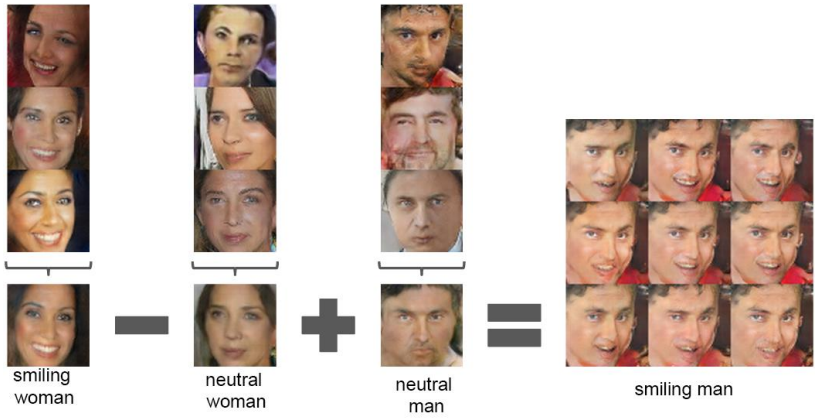
- Radford, Metz, and Chintala's DCGAN trained on Imagenet-1k dataset
  - Discriminator's convolutional features (all layers) down-sampled through max-pooling to 4x4 grid
    - 4x4 grid flattened, concatenated into 28,672 dimensional vector and fed into L2-SVM for classification

Model	Accuracy	Accuracy (400 per class)	Max # of features units
Layer K-means	80.6%	63.7% (± 0.7%)	4800
Layer K-means Learned RF	82.0%	70.7% (± 0.7%)	3200
View Invariant K-means	81.9%	72.6% (± 0.7%)	6400
Exemplar CNN	84.3%	77.4% (± 0.2%)	1024
DCGAN + L2-SVM (R,M,C)	82.8%	73.8% (± 0.4%)	512

Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

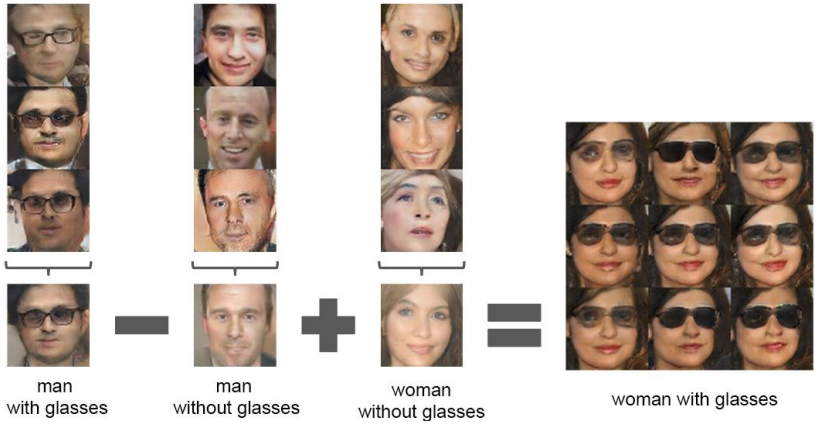
DCGAN: Vector Arithmetic

- Generated images are classified by the human eye
  - "Smiling Woman"
  - "Neutral Man"
- The Generator input vector z, that created these images can be thought of (after training) as the latent variables that represent generated image traits/features
- The human classified examples are stored as their latent representation z
- The latent representations are averaged by category
  - Produces the latent representation, z, of the "average" "smiling woman", etc.
- Averaged latent representations are passed into the Generator to generate the image representation of the model's "average" "smiling woman" etc.



Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN: Vector Arithmetic



- Demonstrated it's possible to perform arithmetic in the latent "z space"
  - Uniform noise added to average categorical representation to yield variable results shown to the right
- Because the input to the Generator is random noise (latent "z space") this shows that the model has learned a representation of "man" and "woman" inside

Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

DCGAN: LSUN Bedrooms

After one pass through the training set!

Reference: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

Welcome to the GAN Jungle

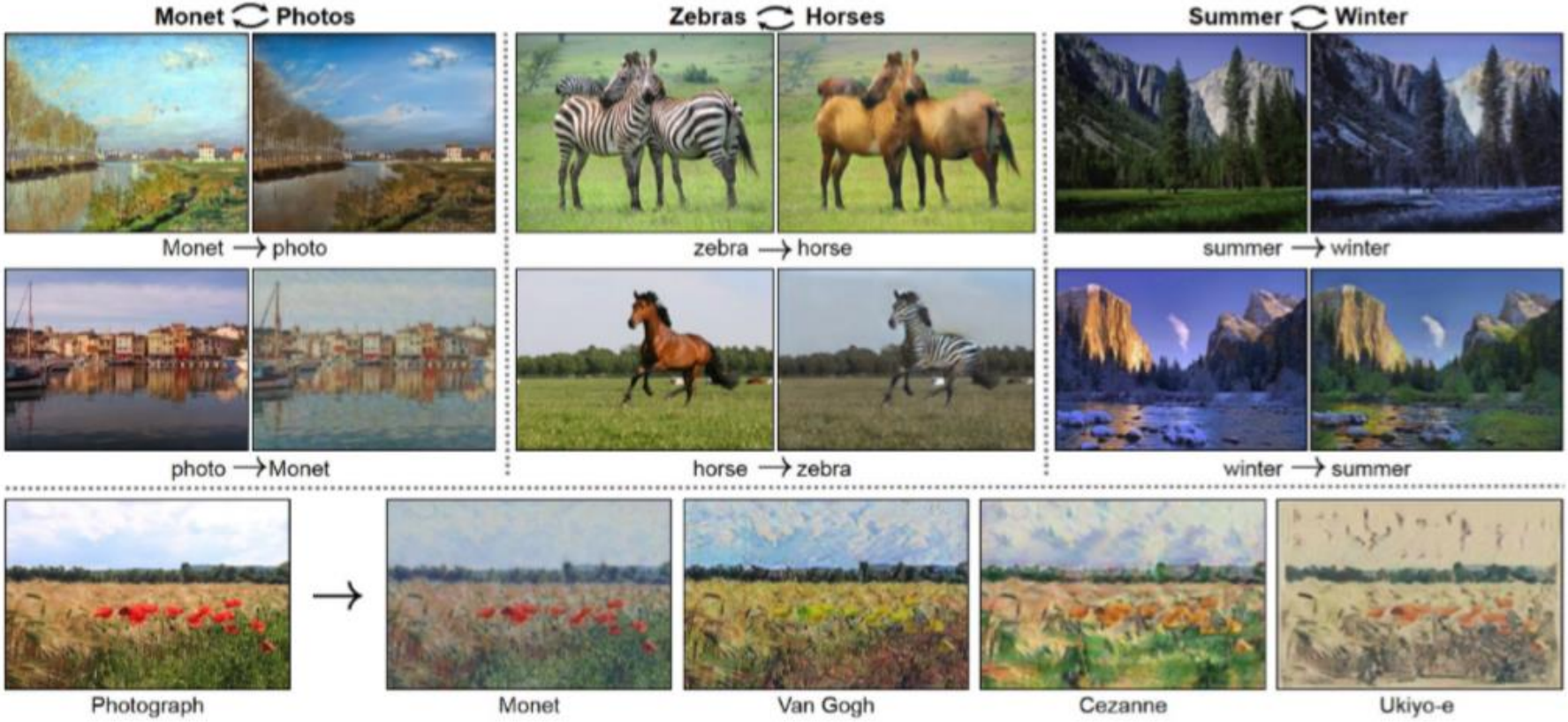
Reference: The GAN Zoo on GitHub - Hindu Puravinash ~ 300 GAN Papers!!!! Mostly GAN variations

•3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
•3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
•3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
•3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
•ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks(github)
•ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
•AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
•acGAN - Face Aging With Conditional Generative Adversarial Networks
•ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
•AdaGAN - AdaGAN: Boosting Generative Models
•AdvGAN - Generating adversarial examples with adversarial networks
•AE-GAN - AE-GAN: adversarial eliminating with GAN
•AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
•AffGAN - Amortised MAP Inference for Image Super-resolution
•AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
•ALI - Adversarially Learned Inference (github)
•AlignGAN - AlignGAN: Learning to Align Cross-Domain Images with Conditional Generative Adversarial Networks
•AM-GAN - Activation Maximization Generative Adversarial Nets
•AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
•APE-GAN - APE-GAN: Adversarial Perturbation Elimination with GAN
•ARAE - Adversarially Regularized Autoencoders for Generating Discrete Structures (github)
•ARDA - Adversarial Representation Learning for Domain Adaptation
•ARIGAN - ARIGAN: Synthetic Arabidopsis Plants using Generative Adversarial Network
•ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
•AttGAN - Arbitrary Facial Attribute Editing: Only Change What You Want
•AttnGAN - AttnGAN: Fine-Grained Text to Image Generation with Attentional Generative Adversarial Networks
•TV-GAN - TV-GAN: Generative Adversarial Network Based Thermal to Visible Face Recognition
•UGACH - Unsupervised Generative Adversarial Cross-modal Hashing
•UGAN - Enhancing Underwater Imagery using Generative Adversarial Networks
•Unim2im - Unsupervised Image-to-Image Translation with Generative Adversarial Networks (github)
•Unrolled GAN - Unrolled Generative Adversarial Networks (github)
•VAE-GAN - Autoencoding beyond pixels using a learned similarity metric
•VariGAN - Multi-View Image Generation from a Single-View
•VAW-GAN - Voice Conversion from Unaligned Corpora using Variational Autoencoding Wasserstein Generative Adversarial Networks
•VEEGAN - VEEGAN: Reducing Mode Collapse in GANs using Implicit Variational Learning (github)
•VGAN - Generating Videos with Scene Dynamics (github)
•VGAN - Generative Adversarial Networks as Variational Training of Energy Based Models (github)
•VGAN - Text Generation Based on Generative Adversarial Nets with Latent Variable
•ViGAN - Image Generation and Editing with Variational Info Generative Adversarial Networks
•VIGAN - VIGAN: Missing View Imputation with Generative Adversarial Networks
•VoiceGAN - Voice Impersonation using Generative Adversarial Networks
•VRAL - Variance Regularizing Adversarial Learning
•WaterGAN - WaterGAN: Unsupervised Generative Network to Enable Real-time Color Correction of Monocular Underwater Images
•WaveGAN - Synthesizing Audio with Generative Adversarial Networks
•weGAN - Generative Adversarial Nets for Multiple Text Corpora
•WGAN - Wasserstein GAN (github)
•WGAN-GP - Improved Training of Wasserstein GANs (github)
•WS-GAN - Weakly Supervised Generative Adversarial Networks for 3D Reconstruction
•XGAN - XGAN: Unsupervised Image-to-Image Translation for many-to-many Mappings
•ZipNet-GAN - ZipNet-GAN: Inferring Fine-grained Mobile Traffic Patterns via a Generative Adversarial Neural Network
•α-GAN - Variational Approaches for Auto-Encoding Generative Adversarial Networks (github)
•Δ-GAN - Triangle Generative Adversarial Networks

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Style transfer problem: change the style of an image while preserving the content.



Data: Two unrelated collections of images, one for each style

- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image.

Real zebra image

The discriminator tries to distinguish generated zebra images from real ones

D

Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real

Reconstruction loss: squared error between the original image and the reconstruction

Input image (real horse image)

G_1

Generator 1 learns to map from horse images to zebra images while preserving the structure

Generated sample

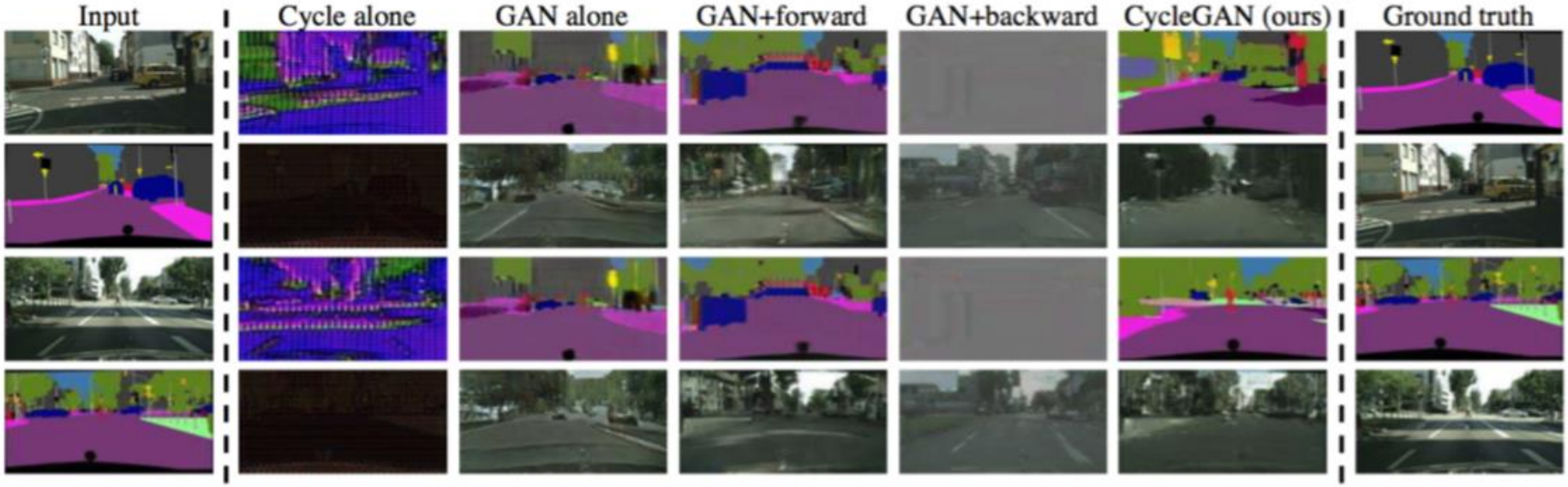
G_2

Generator 2 learns to map from zebra images to horse images while preserving the structure

Reconstruction

Total loss = discriminator loss + reconstruction loss

Style transfer between road scenes and semantic segmentations (labels of every pixel in an image by object category):



How to train GANs?

- Objective of generative network - increase the error rate of the discriminative network.
- Objective of discriminative network – decrease binary classification loss.
- Discriminator training - backprop from a binary classification loss.
- Generator training - backprop the negation of the binary classification loss of the discriminator.

Loss Functions

$$\mathcal{L}(\hat{x}) = \min_{x \in data} (x - \hat{x})^2$$

Generator

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

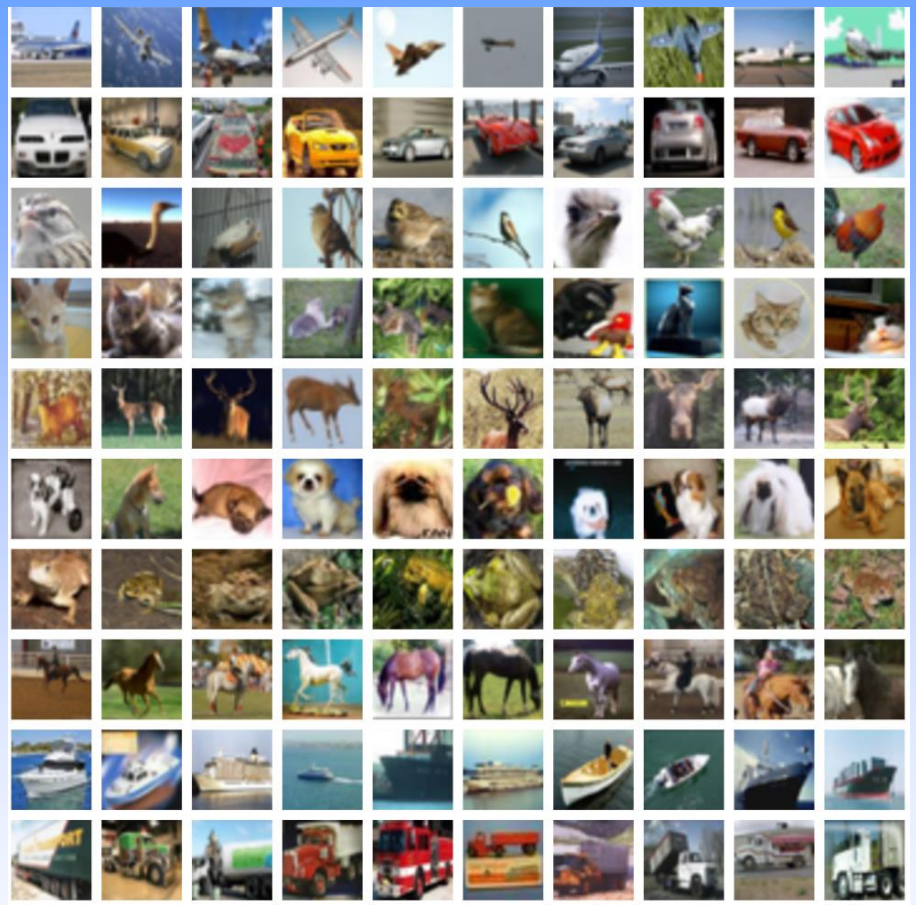
Discriminator

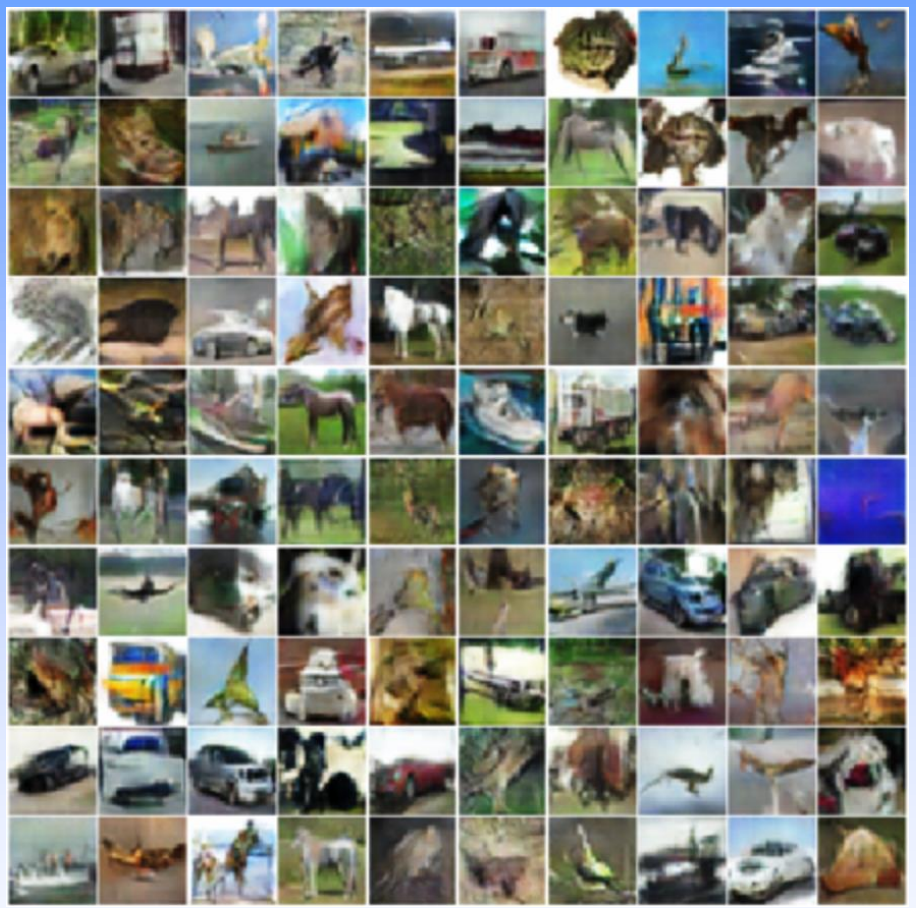
Generated bedrooms. Source: "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" https://arxiv.org/abs/1511.06434v2

"Improved Techniques for Training GANs" by Salimans et. al

- One-sided Label smoothing - replaces the 0 and 1 targets for a classifier with smoothed values, like .9 or .1 to reduce the vulnerability of neural networks to adversarial examples.

- Virtual batch Normalization - each example x is normalized based on the statistics collected on a reference batch of examples that are chosen once and fixed at the start of training, and on x itself.



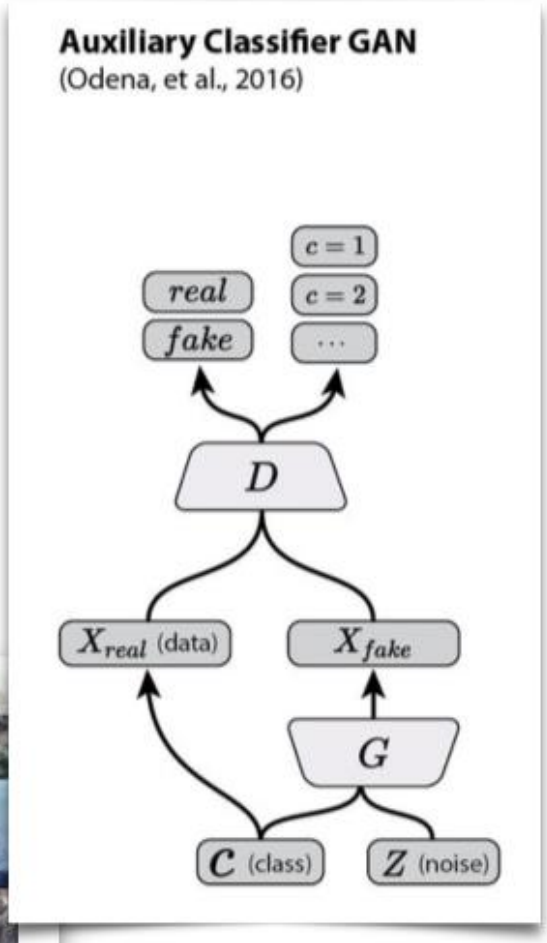


Original CIFAR-10 vs. Generated CIFAR-10 samples

Source: "Improved Techniques for Training GANs" https://arxiv.org/abs/1606.03498

Auxiliary Classifier GAN (ACGAN)

- Similar to CGAN, latent variable is passed to the generator
- Discriminator is tasked with jointly learning real-vs-fake and the ability to reconstruct the latent variable being passed in

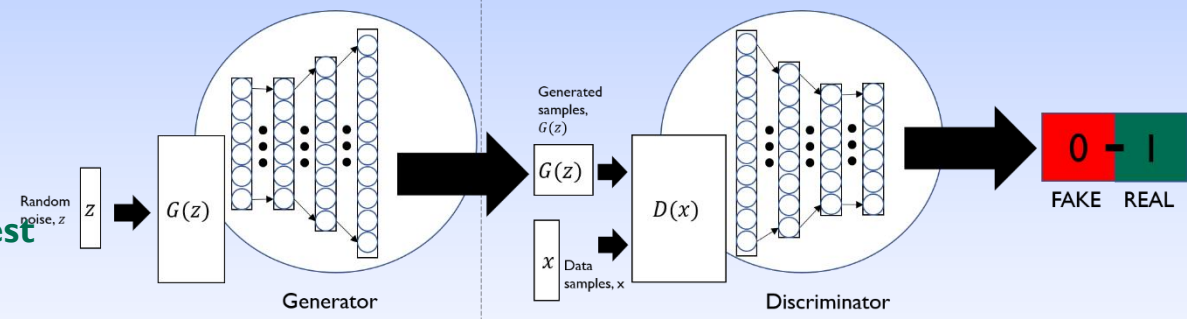


Summary and GANclusions

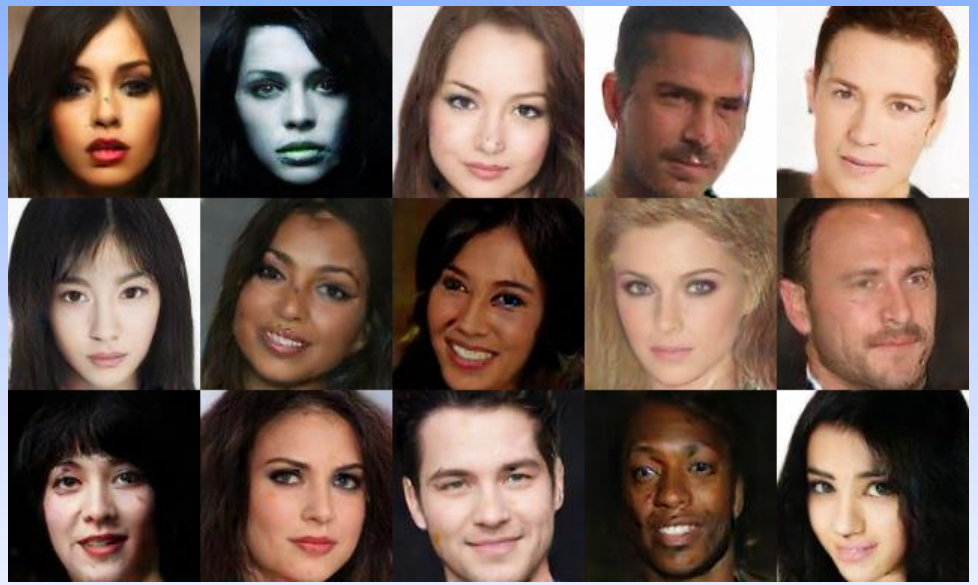
- Generator and Discriminator (sometimes called Critic)
  - Generator wants to produce samples that fool the Discriminator
    - Generator wants to learn real data distribution
  - Discriminator wants to catch the fakes
  - Original concept played minimax game during training
- DCGANs sparked GAN growth
- The ever long search for optimization
  - Training GANs is tough!
    - Instability
    - Mode Collapse
- FID: Useful GAN comparison
  - **Different GANs may have advantages but no one is best**

$$p_g(z) = p_{data}(x)$$

For training tips and tricks: [GAN Hacks (GitHub)]

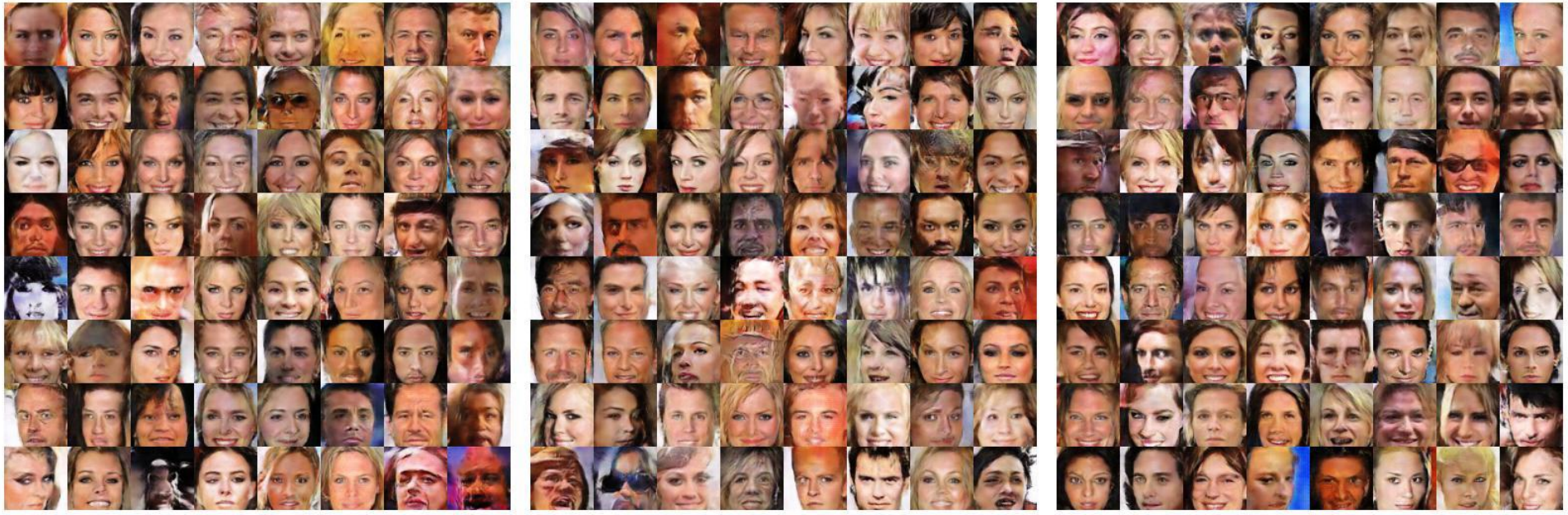


Pretty Pictures from GANs



BEGAN trained on CelebA - BEGAN: Boundary Equilibrium Generative Adversarial Networks

Pretty Pictures from GANs



MMGAN trained on CelebA - MMGAN: Manifold-Matching Generative Adversarial Network

Pretty Pictures from GANs

LSGAN - Least Squares Generative Adversarial Networks

References

(1) Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. *Generative adversarial nets. NIPS, 2014.*

(2) A. Radford, L. Metz and S. Chintala, *Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.*

(3) Mirza, Mehdi and Osindero, Simon. *Conditional generative adversarial nets. 2014.*

(4) Augustus Odena, Christopher Olah, Jonathon Shlens, *Conditional Image Synthesis with Auxiliary Classifier GANs. ICML, 2017.*

(5) Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, Pieter Abbeel. InfoGAN: *Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. 2016.*

(6) Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. *Improved Techniques for Training GANs. NIPS, 2016.*

https://www.kdnuggets.com/2019/08/video-introduction-generative-adversarial-networks.html

Learn how to morph faces with a Generative Adversarial Network!

https://www.youtube.com/watch?v=dCKbRCUyop8

Creating and Training a Generative Adversarial Networks (GAN) in Keras (7.2)

https://www.youtube.com/watch?v=T-MCludVNn4