IERG4190/IEMS5707 Course Summary

Bolei Zhou

The Chinese University of Hong Kong

Format for the final exam

- One A4 paper (one-side) as cheat-sheet
- Multiple-choice questions and short-answer questions through Blackboard
- Length: 1 hour or 1.5 hours (To be determined)
- Online exam invigilation:
  - Find a smooth and stable Internet condition
  - Prepare one webcam. Keep your web-camera open through the ZOOM (you will be monitored by TAs and me).
  - Stay in Blackboard browser all the time, no web browsing or other communication. Your screen will be recorded, and video will be sent back for check

Three components of the course

- Multimedia coding
- Multimedia processing
- Multimedia understanding

Multimedia Coding and Processing

multimedia

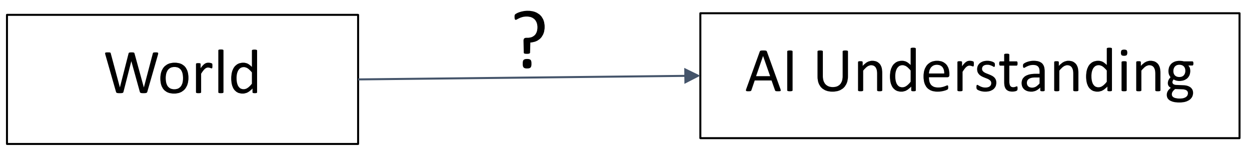
/ˈmʌltɪmiːdɪə/

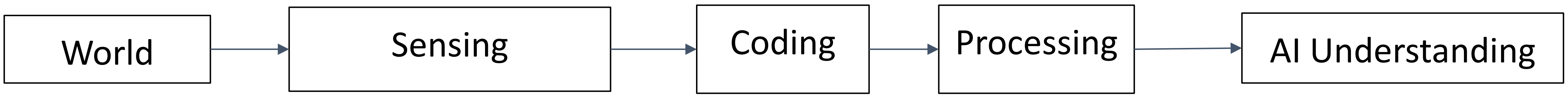
adjective

adjective: **multimedia**; adjective: **multi-media**

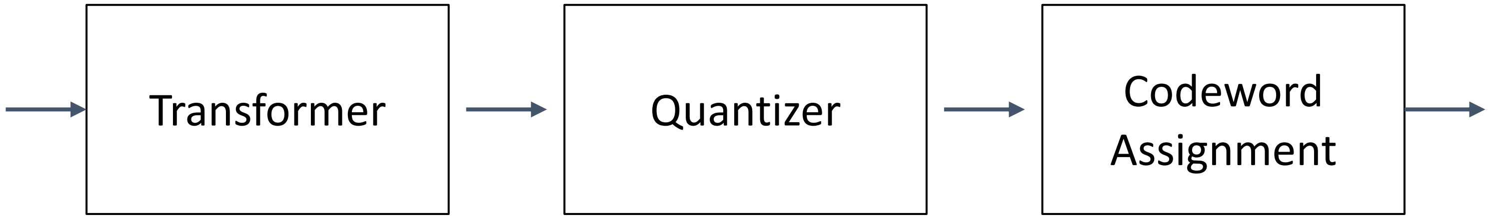
1. (of art, education, etc.) using more than one medium of expression or communication.
   - (of computer applications) incorporating audio and video, especially interactively.
     "multimedia applications"

Modern Multimedia System





Multimedia Coding: Elements of a Multimedia Coder

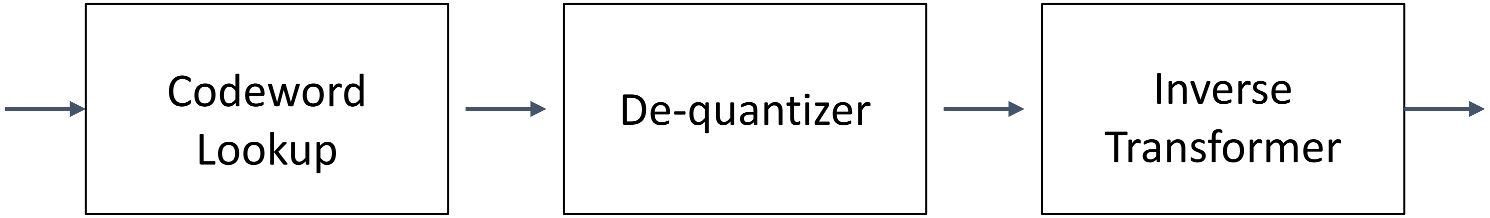


Transformer: transform the input data into a form more amenable to compression
e.g. Discrete Fourier Transforms (DFT), Discrete Cosine Transform (DCT) etc.

Quantizer: represent transformed signal with a limited number of levels/symbols; an irreversible operation;
E.g. scalar quantization, vector quantization

Codeword Assignment:
Assign codewords to the quantized output, creating a bit stream
e.g. fixed length coding v.s variable length coding

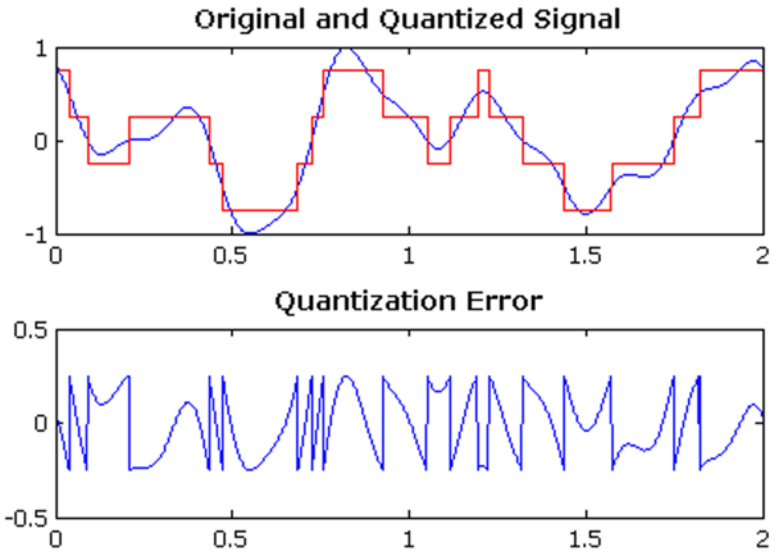
Multimedia Coding: Elements of a Multimedia Decoder



Codeword Lookup: Decodes bitstream into quantized levels
Dequantizer: Reverse the quantization of quantizer
Inverse Transformer: Reverse the transformation for display/playback

Scalar Quantization

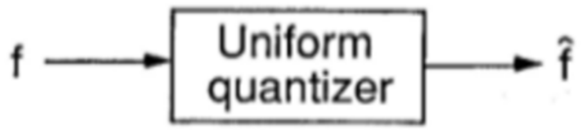
To represent a continuous scalar value *f* with a finite number of bits, only a finite number of quantization levels *L* can be used. If each scalar is quantized independently, the procedure is called scalar quantization.



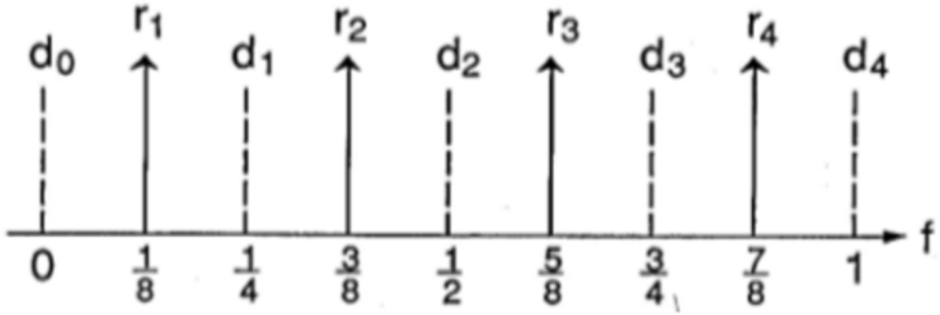
Uniform Quantization

Uniform quantization: equal spacing of reconstruction levels.
Example: image intensity f : 0~1

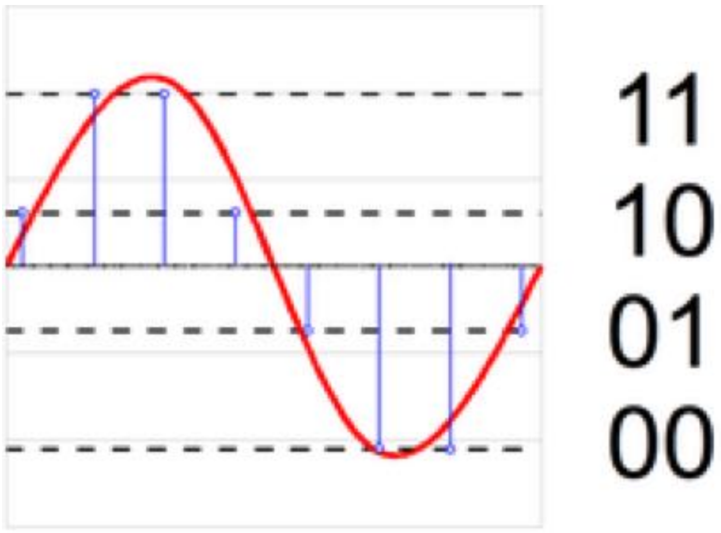
r_i : reconstruction levels
d_i : decision bounaries



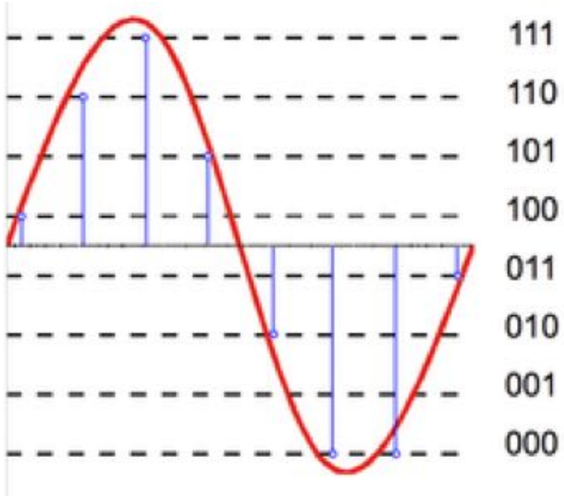
Number of reconstruction levels : 4



Relation between Bits and Levels



2-bit resolution with four levels



3-bit resolution with eight levels

Vector Quantization for RGB Images

- Each pixel is thus represented by an RGB vector [r,g,b] usually already quantized individually to 256 levels ([0,255])
- How many bits?

The three RGB colors are each **8**-bits (possible values [0.. 255], 2^8 = **256**)

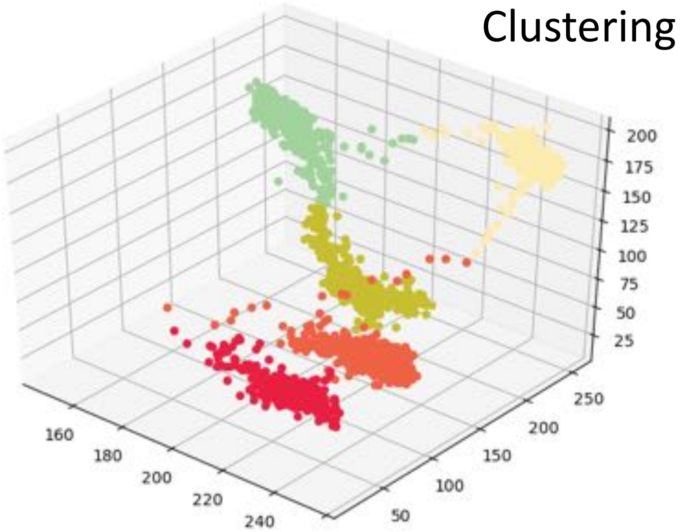
[110,165,247]

[12,169,125]

Vector Quantization for RGB Images

- In Vector Quantization we call the set of reconstruction levels a **codebook** or **dictionary** and the space with each decision boundary a **cell**
- Using our intuition, the reconstruction levels could be the **center** (or more properly **centroids)** of these cells

Clustering in RGB space



Vector Quantization for RGB Images

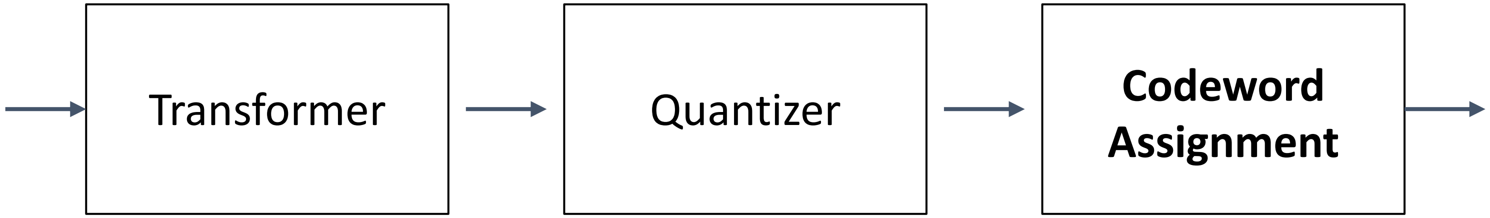
- K-means clustering (how it works?)
- Having K reconstruction levels means codebook of size K
    - Need only $\log_2(K)$ bits per pixel to store/transmit

Original
8 x 3 bits

K = 24 VQ
~5 bits

K = 64 VQ
6 bits

Codeword Assignment



- We have seen some simple design choices we have for **quantization**
- We will now move on to **codeword assignment**
  - Fixed-length coding (FLC)
  - Variable-length coding (VLC)

Fixed-Length Coding

- We code a quantized symbol into bits of a certain length
- Least trouble: fixed length coding
  - 8 symbols (levels of quantization)
  - 3 bits!

001010101000001111110

↓

001 010 101 000 001 111 110

↓

r1 r2 r5 r0 r1 r7 r6

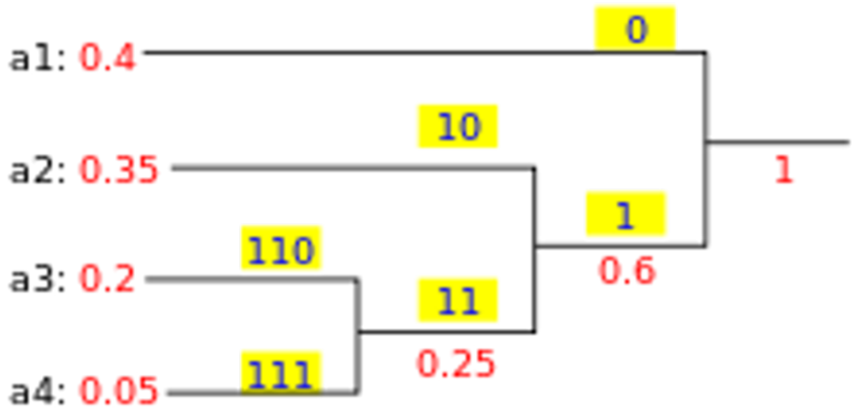
$L = 8$	3 bits
r_0	0 0 0
r_1	0 0 1
r_2	0 1 0
r_3	0 1 1
r_4	1 0 0
r_5	1 0 1
r_6	1 1 0
r_7	1 1 1

Variable-length Coding: Huffman Coding

Intuition: if we use more bits for more rare symbols and less bits for more frequent symbols, will we be using less bits, as a whole?

e.g. use 1 bit for most frequent color, and many bits for less used color

How to do Huffman Coding: See lecture slide

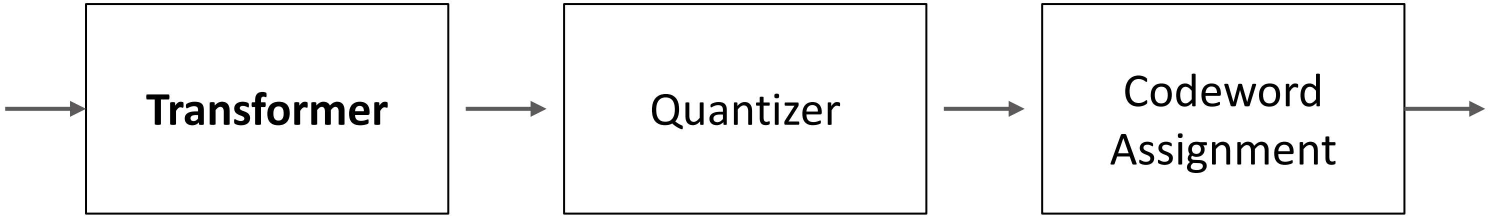


Symbol	Final Code
a1	0
a2	10
a3	110
a4	111

Variable-length Coding: Adaptive Dictionary Methods

- Make sure you understand how LZ77, LZ78, LZW work! (see examples in the Week2 slide), practice by yourself

Transformer



Transform coding: Discrete Fourier Transform (DFT) and Discrete Cosine Transform (DCT)

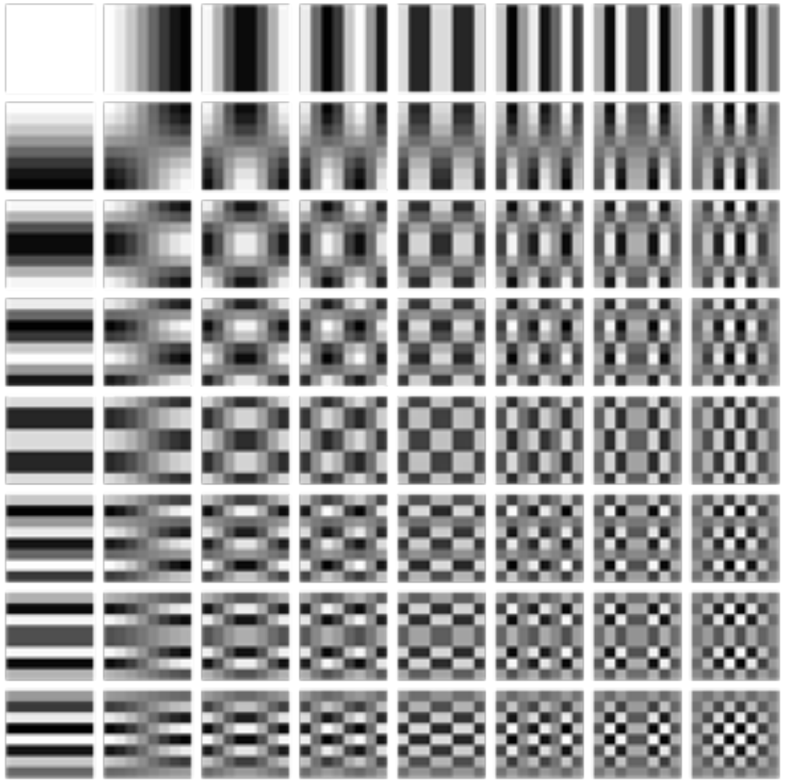
- Map signal from raw representation (e.g. pixels for images) to a set of linear transform coefficients

Success criteria

- Small number of transform coefficients should carry most signal energy
- => lossy compression

DCT and Transform Coding

- 8x8 DCT Basis



$$\text{DCT of the image} = \begin{bmatrix} 6.1917 & -0.3411 & 1.2418 & 0.1492 & 0.1583 & 0.2742 & -0.0724 & 0.0561 \\ 0.2205 & 0.0214 & 0.4503 & 0.3947 & -0.7846 & -0.4391 & 0.1001 & -0.2554 \\ 1.0423 & 0.2214 & -1.0017 & -0.2720 & 0.0789 & -0.1952 & 0.2801 & 0.4713 \\ -0.2340 & -0.0392 & -0.2617 & -0.2866 & 0.6351 & 0.3501 & -0.1433 & 0.3550 \\ 0.2750 & 0.0226 & 0.1229 & 0.2183 & -0.2583 & -0.0742 & -0.2042 & -0.5906 \\ 0.0653 & 0.0428 & -0.4721 & -0.2905 & 0.4745 & 0.2875 & -0.0284 & -0.1311 \\ 0.3169 & 0.0541 & -0.1033 & -0.0225 & -0.0056 & 0.1017 & -0.1650 & -0.1500 \\ -0.2970 & -0.0627 & 0.1960 & 0.0644 & -0.1136 & -0.1031 & 0.1887 & 0.1444 \end{bmatrix}$$

Original size, scaled 10x (nearest neighbor), scaled 10x (bilinear).

Source data 8x8 is transformed to a linear combination of these 64 frequency squares.

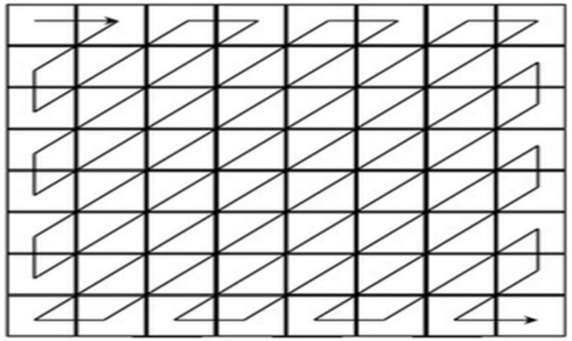
DCT and Transform Coding

- Not feasible to transform whole image so we divide into image blocks of 8 by 8 pixel or 16 by 16 pixel
- We seek to retain transform coefficients that are most significant to our image blocks
- Apart from cropping insignificant transform coefficients, we may wish to encode the amplitude of the transform more carefully
  - Zonal Coding
  - Threshold Coding

This is an example of DCT coefficient matrix:

$$\begin{bmatrix} -415 & -33 & -58 & 35 & 58 & -51 & -15 & -12 \\ 5 & -34 & 49 & 18 & 27 & 1 & -5 & 3 \\ -46 & 14 & 80 & -35 & -50 & 19 & 7 & -18 \\ -53 & 21 & 34 & -20 & 2 & 34 & 36 & 12 \\ 9 & -2 & 9 & -5 & -32 & -15 & 45 & 37 \\ -8 & 15 & -16 & 7 & -8 & 11 & 4 & 7 \\ 19 & -28 & -2 & -26 & -2 & 7 & -44 & -21 \\ 18 & 25 & -12 & -44 & 35 & 48 & -37 & -3 \end{bmatrix}$$

Zig-zag scanning

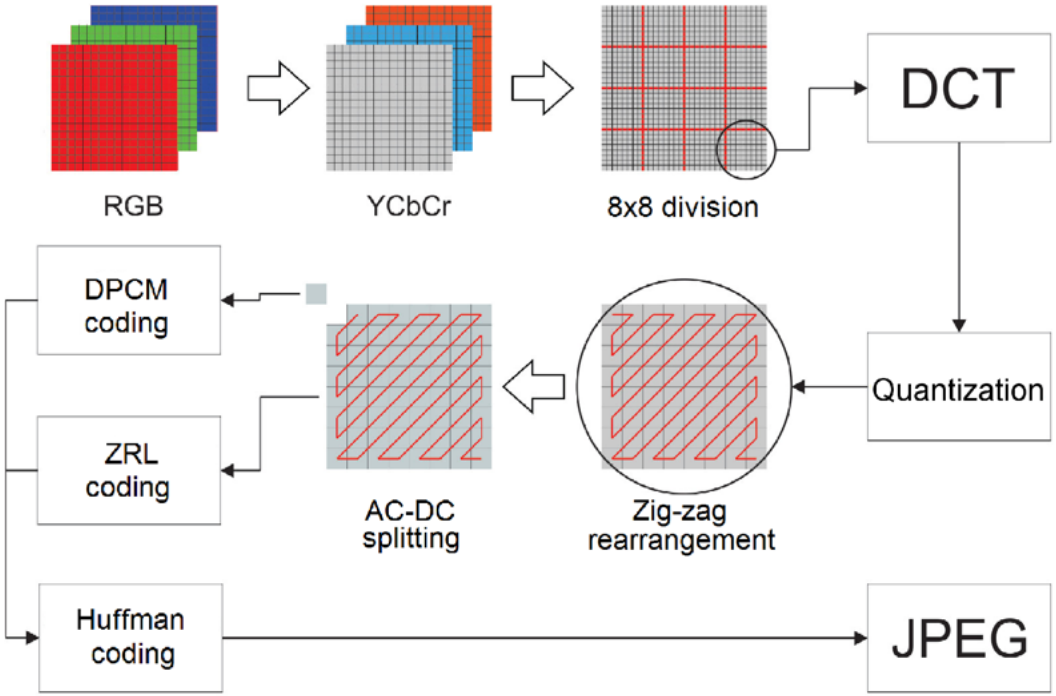


- Example:
  WWWWWWWWWWWWBWWWWWWWWWW
  WWBBBWWWWWWWWWWWWWWWWWWW
  WWWWWBWWWWWWWWWWWWWW

  Output: 12W1B12W3B24W1B14W

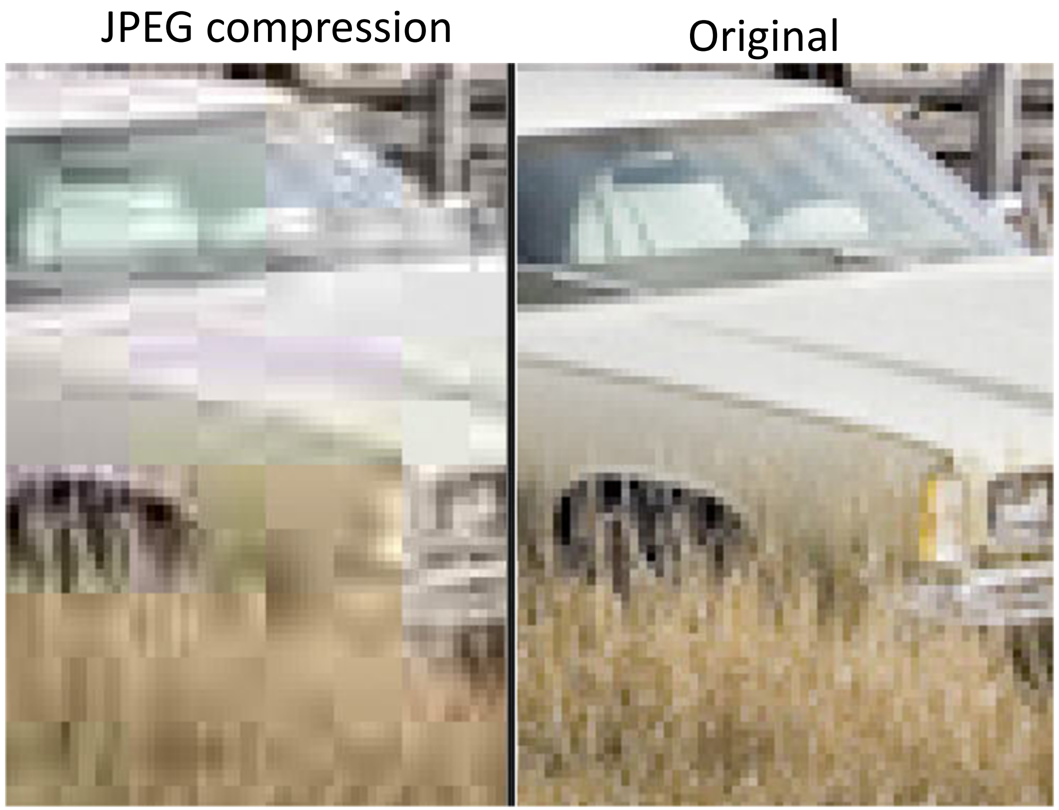
JPEG

- Joint Photographic Experts Group
- First standard issued 1992
  - Free code library *libjpeg* released in 1991



JPEG

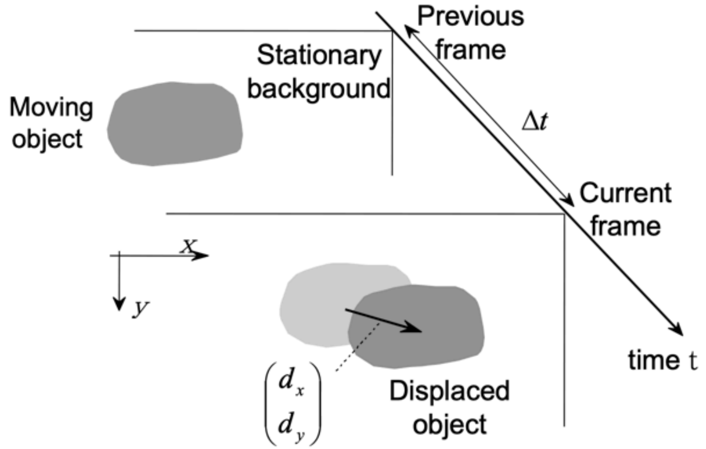
Blocks clearly visible when JPEG quality is extremely low



Video Coding

- Extra dimension: Time with Temporal redundancy
- Motion Estimation
- Different frame types:
  - I-picture
  - P-picture
  - B-picture

Motion estimation



Prediction for the luminance signal $s[x,y,t]$ within the moving object:

$$\boxed{\hat{s}[x,y,t] = s'(x-d_x, y-d_y, t-\Delta t)}$$

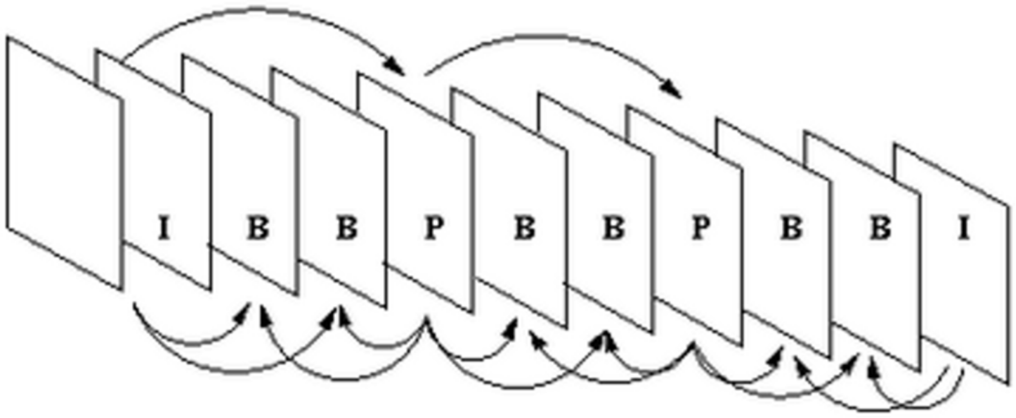
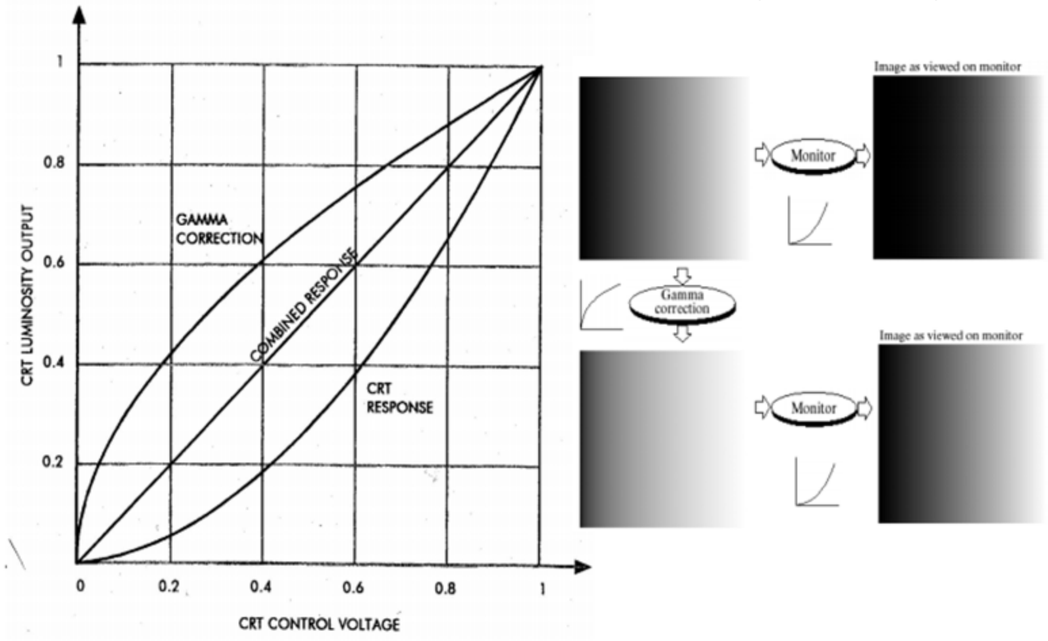


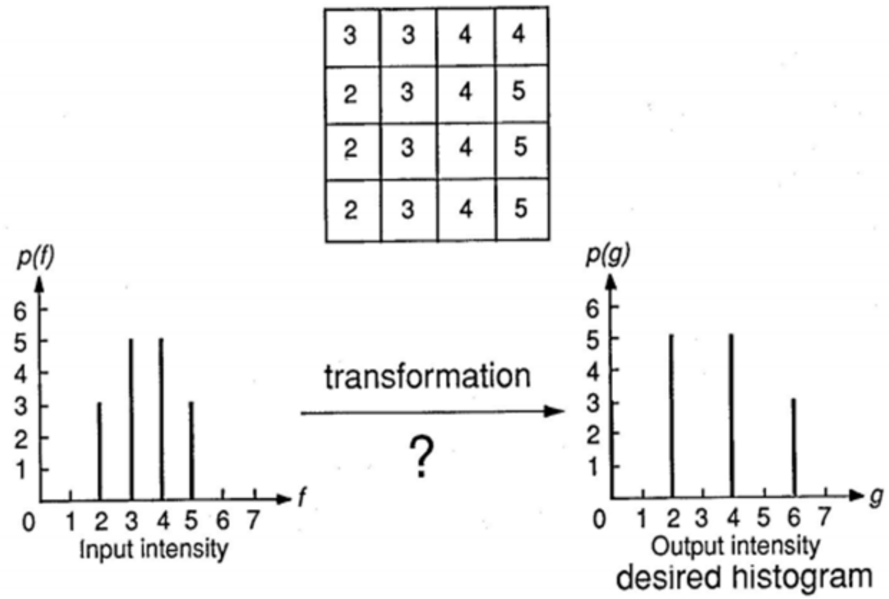
Image Processing

Gamma Correction (point processing)



Histogram Equalization

Histogram: number of pixels as a function of image intensity.



Color Spaces

- Transformation between RGB and YCbCr

The equivalent matrix manipulation is often referred to as the "color matrix":

$$\begin{bmatrix} Y' \\ P_B \\ P_R \end{bmatrix} = \begin{bmatrix} K_R & K_G & K_B \\ -\frac{1}{2}\cdot\frac{K_R}{1-K_B} & -\frac{1}{2}\cdot\frac{K_G}{1-K_B} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2}\cdot\frac{K_G}{1-K_R} & -\frac{1}{2}\cdot\frac{K_B}{1-K_R} \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

And its inverse:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 2-2\cdot K_R \\ 1 & -\frac{K_B}{K_G}\cdot(2-2\cdot K_B) & -\frac{K_R}{K_G}\cdot(2-2\cdot K_R) \\ 1 & 2-2\cdot K_B & 0 \end{bmatrix} \begin{bmatrix} Y' \\ P_B \\ P_R \end{bmatrix}$$

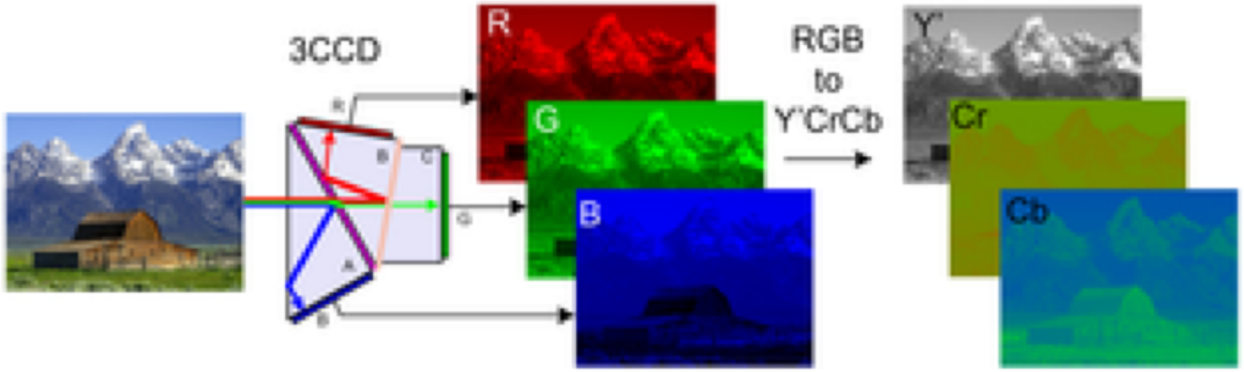
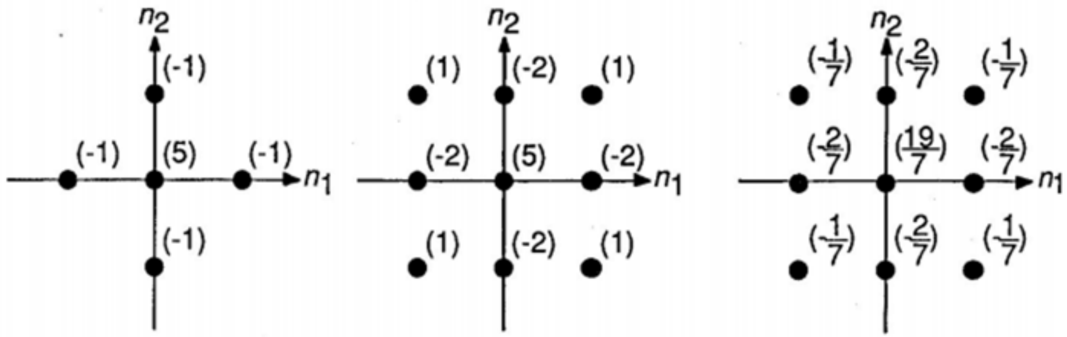


Image Filtering

Typical highpass filters used:



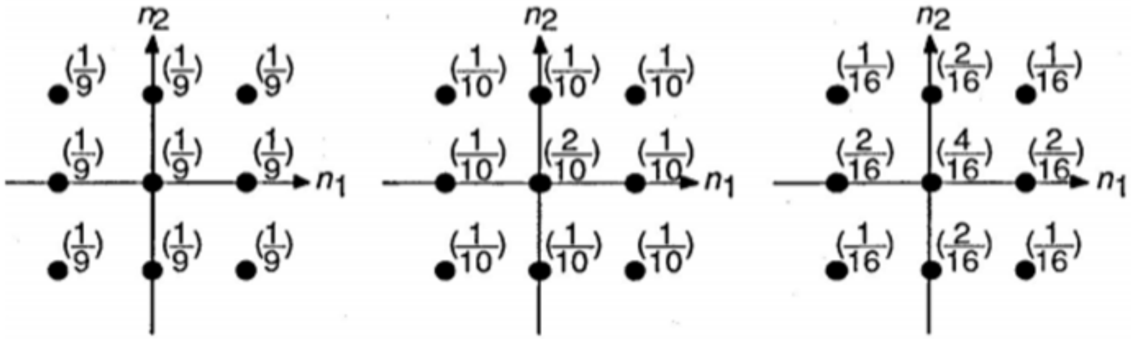
0	-1	0
-1	5	-1
0	-1	0

1	-2	1
-2	5	-2
1	-2	1

$\frac{1}{7}$

-1	-2	-1
-2	19	-2
-1	-2	-1

Typical lowpass filters used:

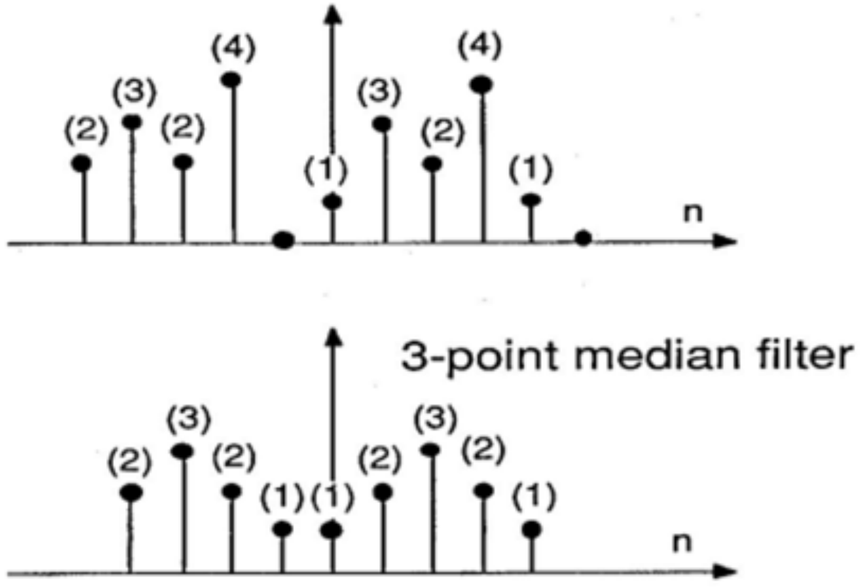


What is the visual output of the filtering?

Image Filtering

In a median filter, a window slides along the image, and the median intensity value of the pixels within the window becomes the output intensity of the pixel being processed.

Example: median [4, 0, 1] = 1



Applications of Image Filtering

- Edge detection: how it works? which filter to use?
- Denoising: which filter to use?

Introduction to Deep Learning

- Convolutional Neural Networks

PROC. OF THE IEEE, NOVEMBER 1998 7

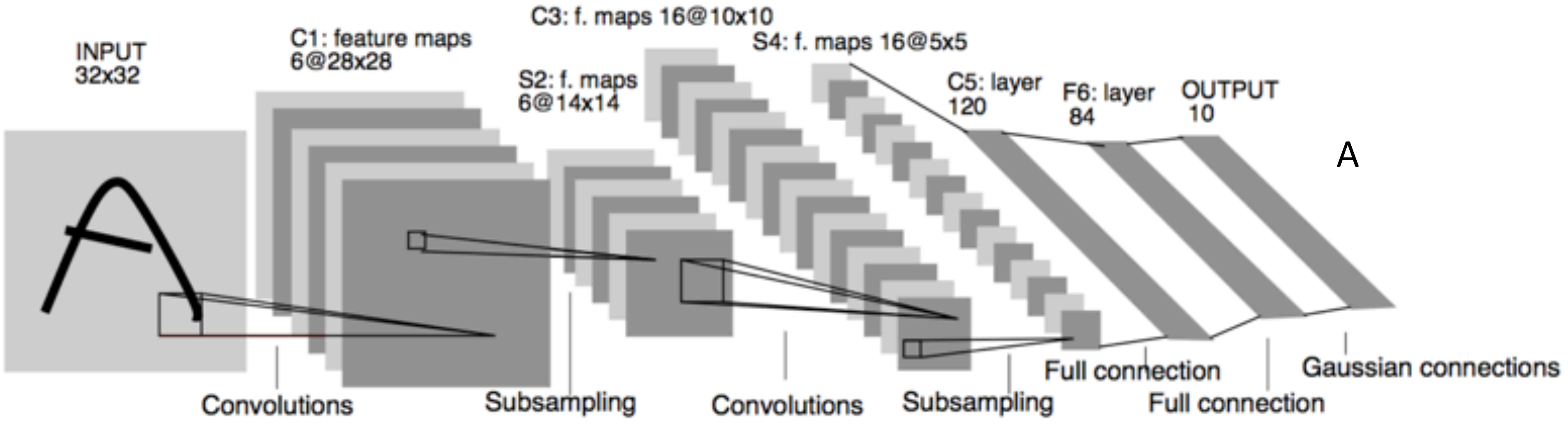
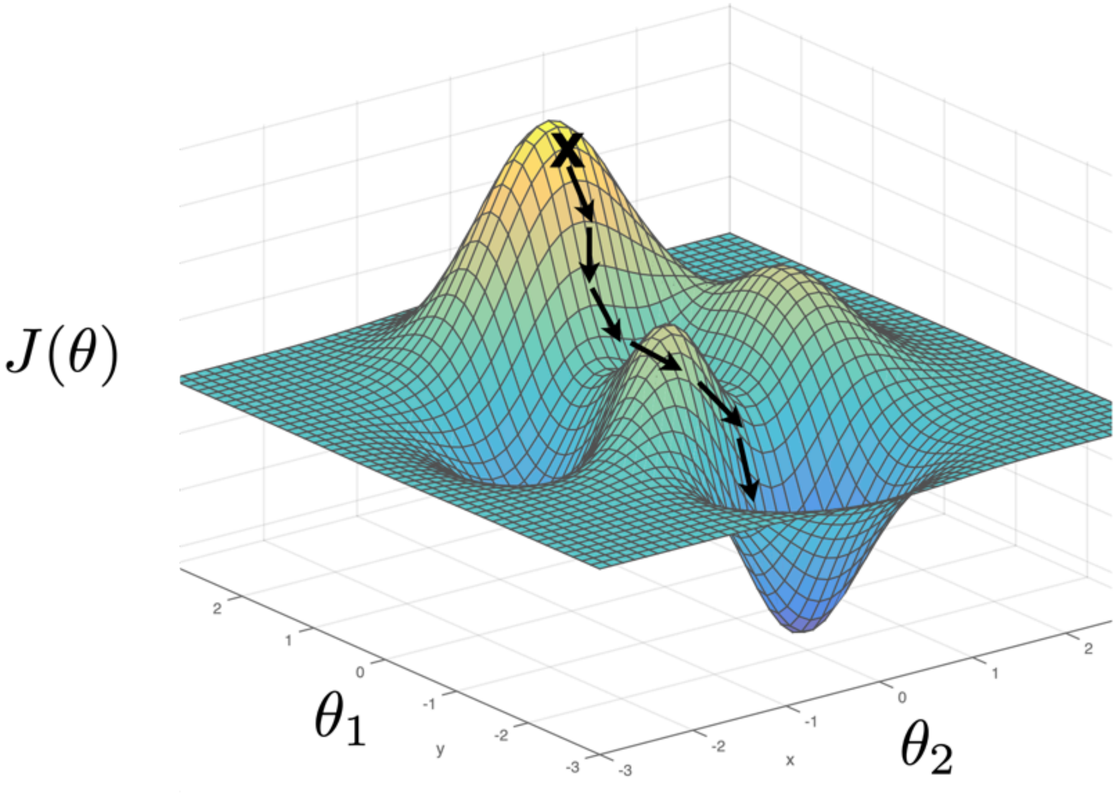


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Gradient Descend



$$\boxed{\theta^* = \arg\min_{\theta} \underbrace{\sum_{i=1}^{N} \mathcal{L}(f_\theta(\mathbf{x}_i), \mathbf{y}_i)}_{J(\theta)}}$$

One iteration of gradient descent:

$$\theta^{t+1} = \theta^t - \eta_t \left.\frac{\partial J(\theta)}{\partial \theta}\right|_{\theta=\theta^t}$$

learning rate

Convolution Filters

When mapping from

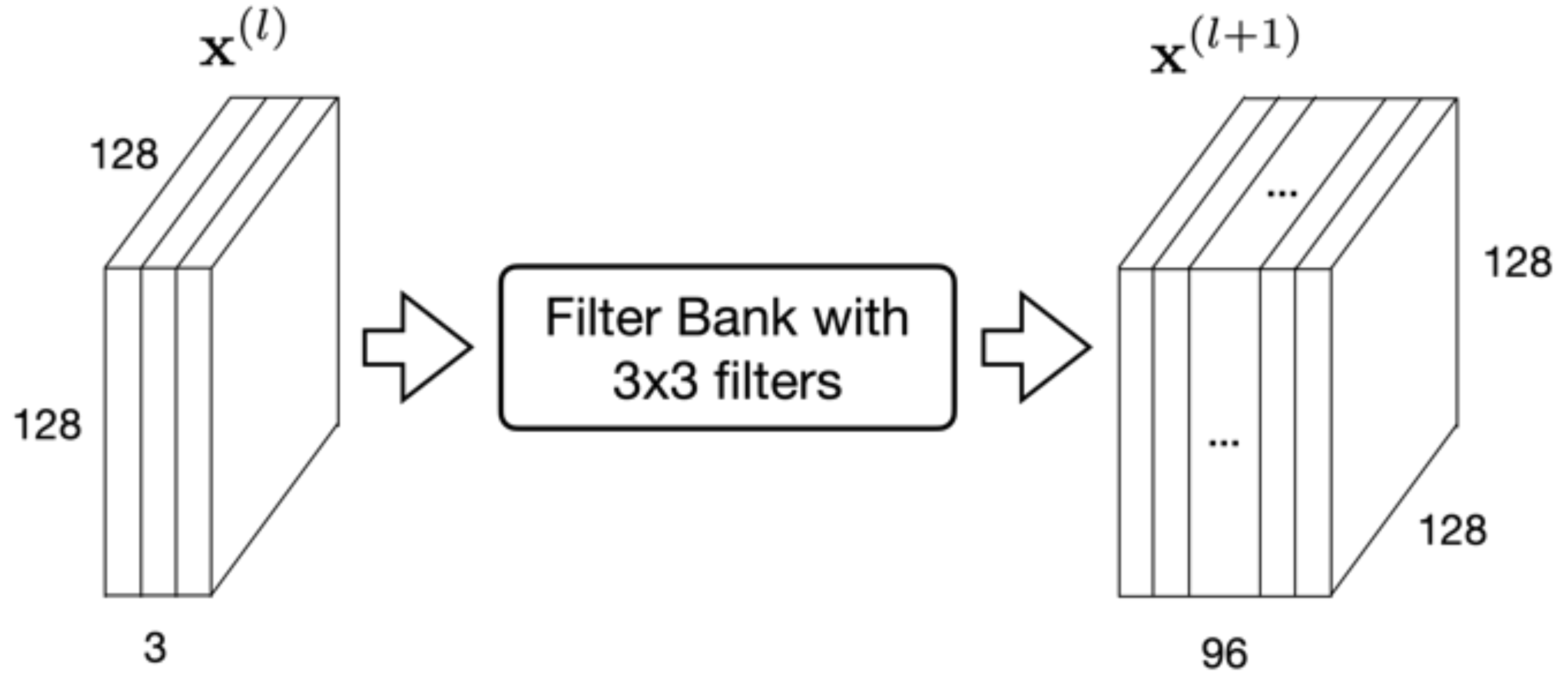
$$\mathbf{x}^{(l)} \in \mathbb{R}^{H \times W \times C^{(l)}} \rightarrow \mathbf{x}^{(l+1)} \in \mathbb{R}^{H \times W \times C^{(l+1)}}$$

using an filter of spatial extent $M \times N$

Number of parameters per filter: $M \times N \times C^{(l)}$

Number of filters: $C^{(l+1)}$

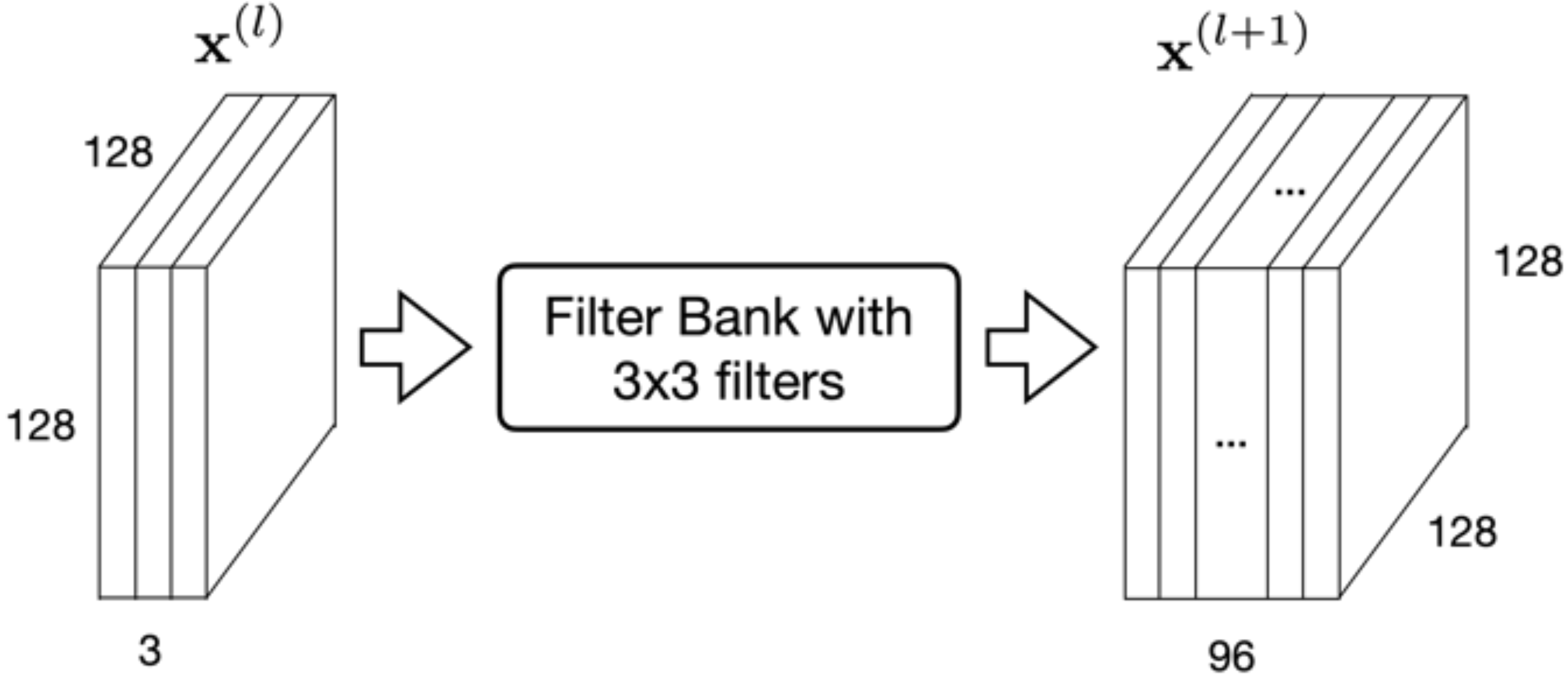
Convolution Filters



How many parameters does *each filter* have?

(a) 9 (b) 27 (c) 96 (d) 864

Convolution Filters

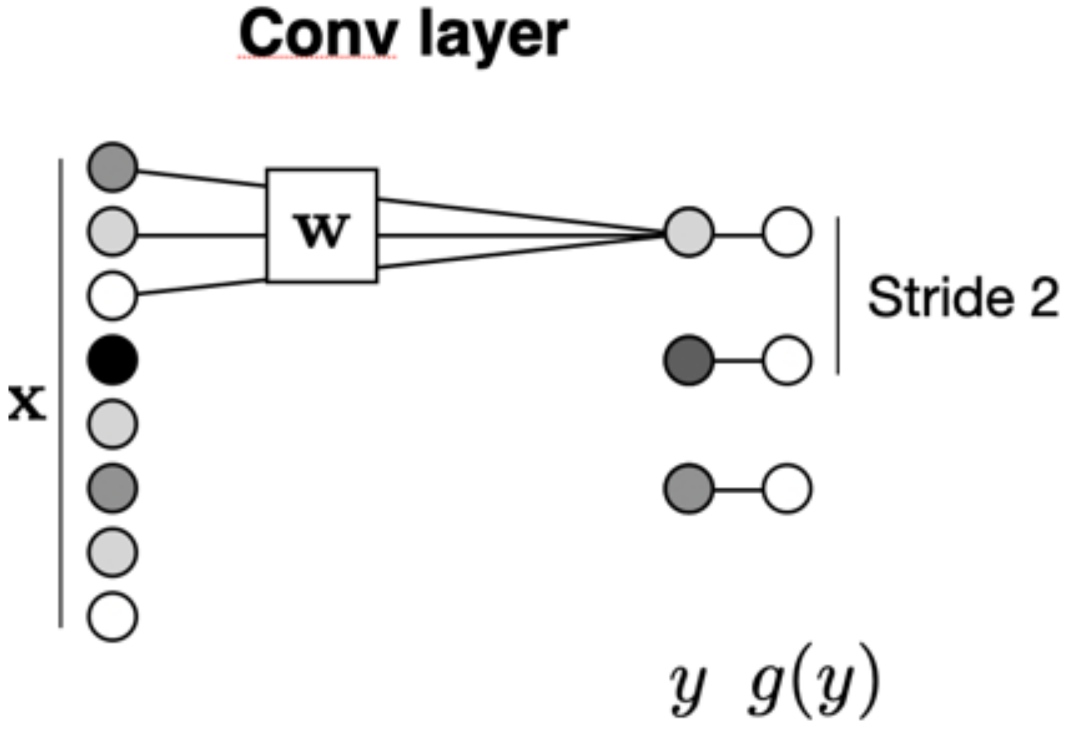


How many filters are in the bank?

(a) 3 (b) 27 (c) 96 (d) can't say

Strided Operations

- **Strided operations** combine a given operation (convolution or pooling) and downsampling into a single operation.



Pooling Operations

Max pooling

$$z_k = \max_{j \in \mathcal{N}(j)} g(y_j)$$

Mean pooling

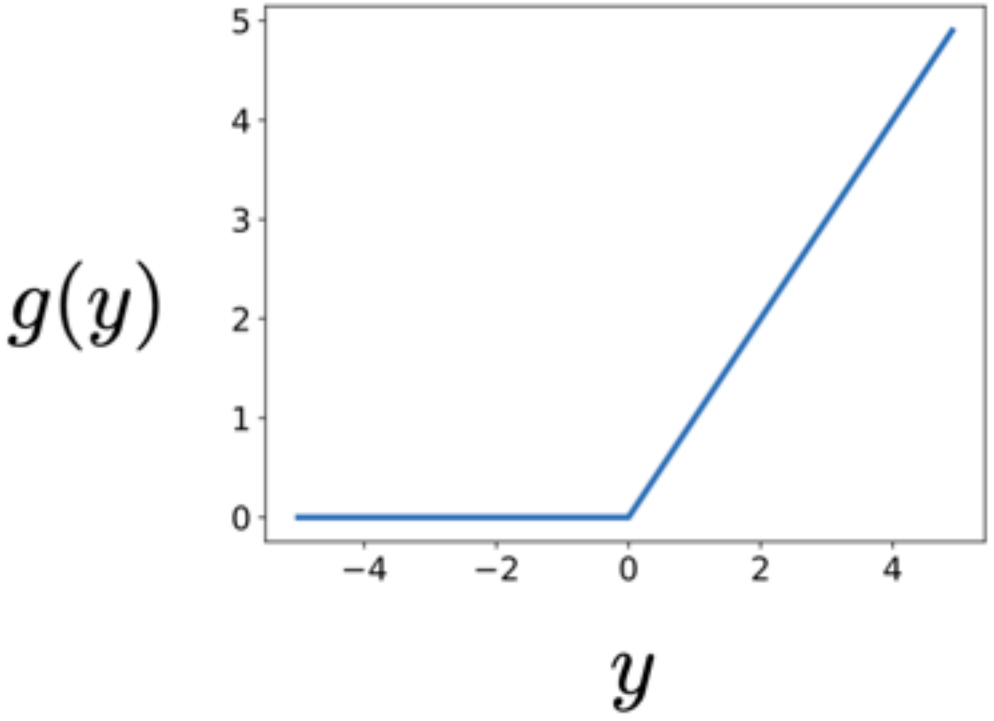
$$z_k = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}(j)} g(y_j)$$

Global average pooling:
average each elements on a feature map into a scalar

Nonlinear Operation: ReLU

Rectified linear unit (ReLU)

$$g(y) = \max(0, y)$$



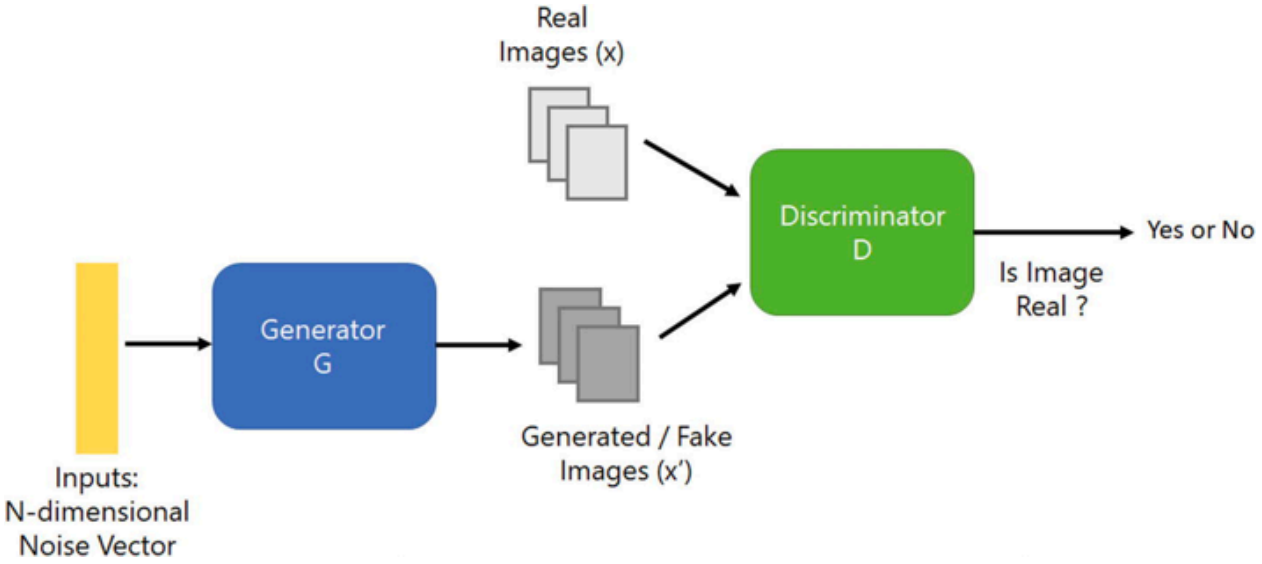
Programming examples

- There will be some programming examples
- Check Tutorial Session’s slides

Generative Modeling

- Different generative models (see corresponding lecture slide)
  - Likelihood-based models: Autoregressive models, RNN, PixelCNN, PixelRNN, WaveNet
  - Latent variable models: Variational Autoencoder (VAE)
  - Implicit generative models: Generative Adversarial Networks (GANs)
- Their properties: Inference capability? Strengths and weaknesses? Diversity of the generated images

Generative Adversarial Networks



Training objective for discriminator:

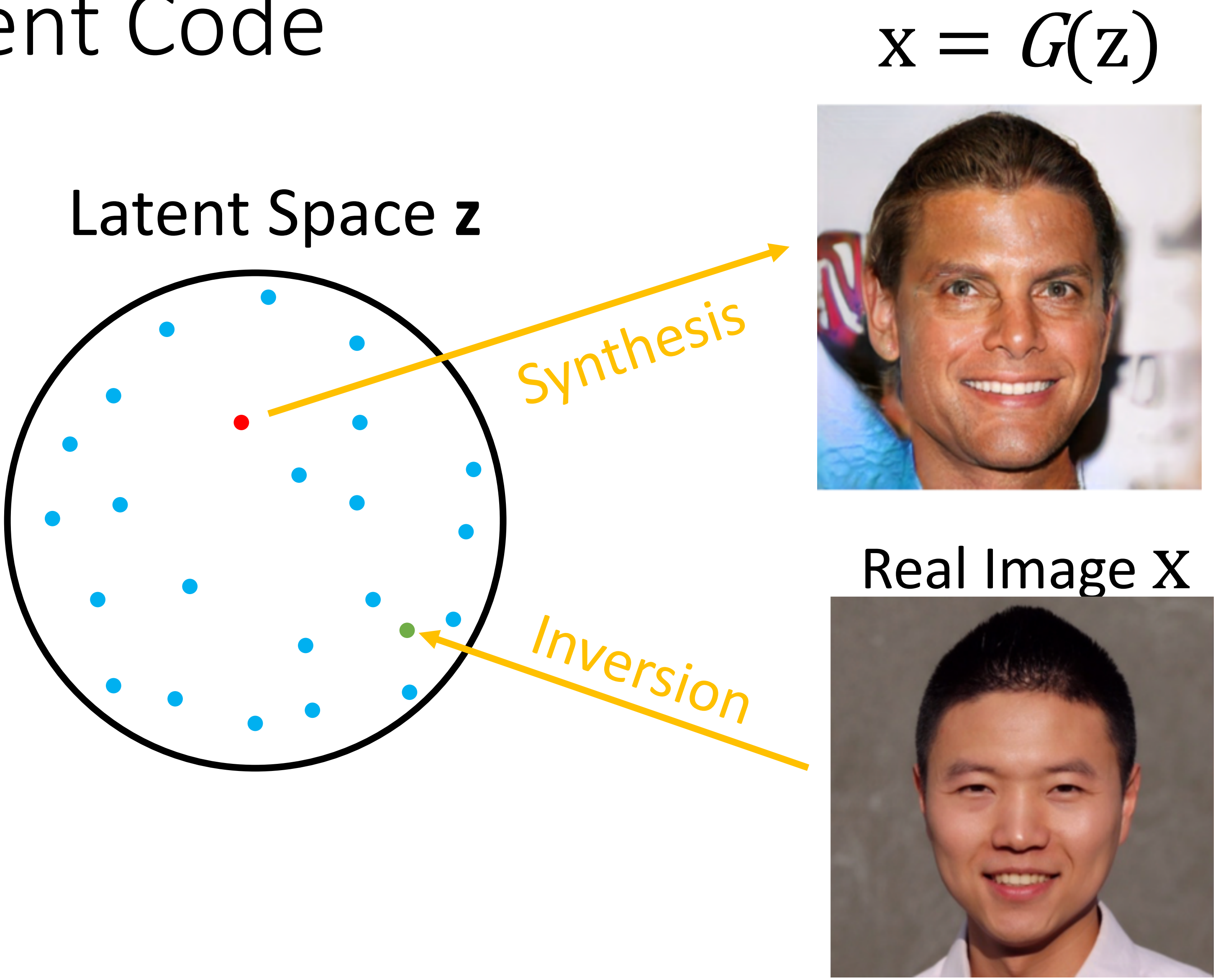
$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

Training objective for generator:

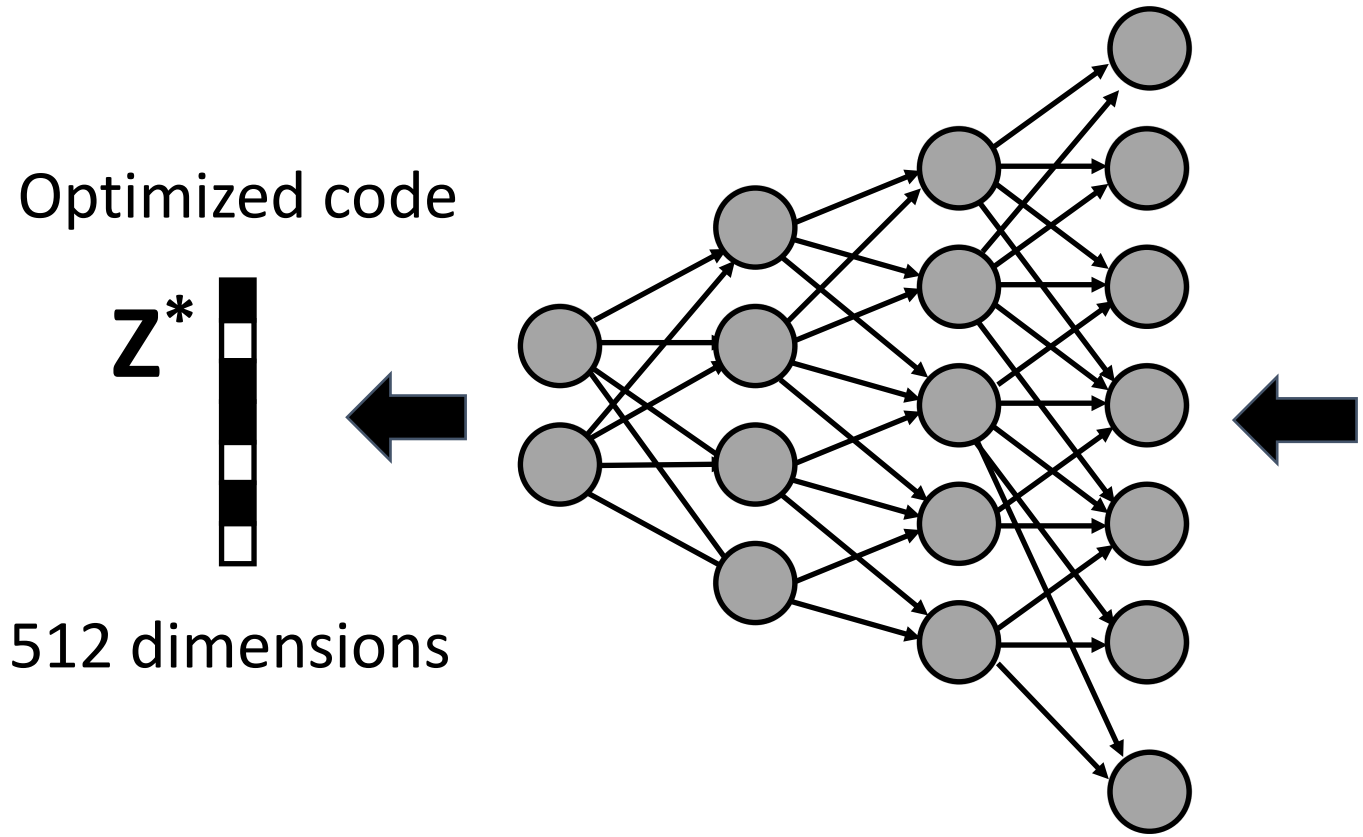
$$\min_G V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

$$\min_\theta \max_\phi V(G_\theta, D_\phi) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z} \sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

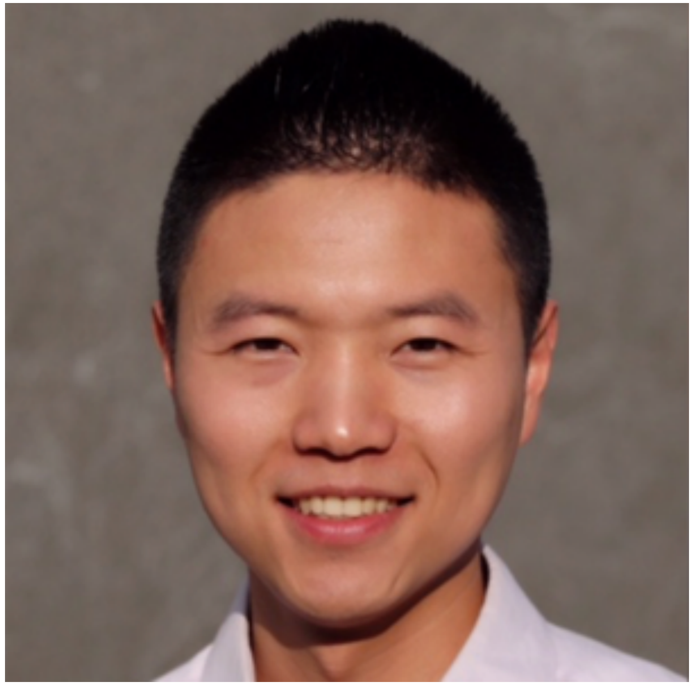
Inverse Problem in GANs: Inverting Real Face to Latent Code



Inverse Problem in GANs: Inverting Real Face to Latent Code

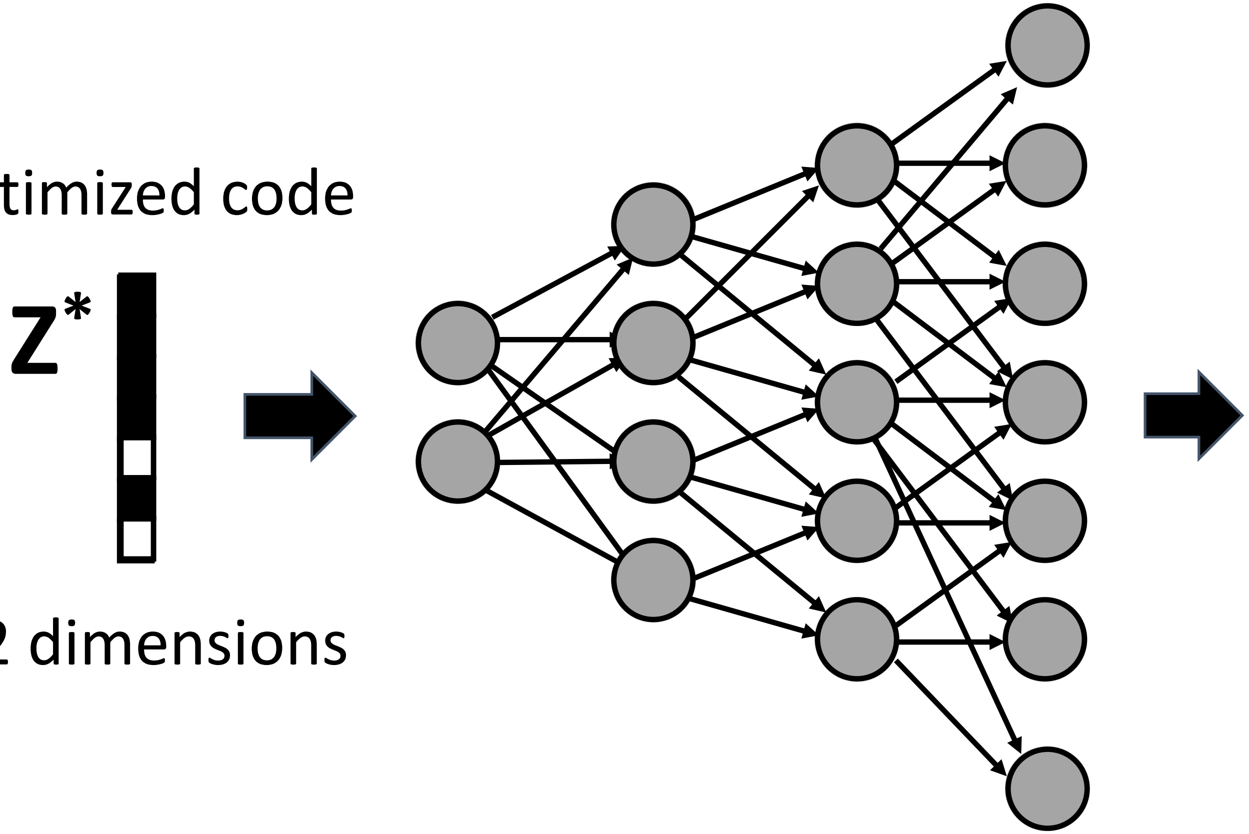


Real face x



Inverse Problem in GANs: Inverting Real Face to Latent Code

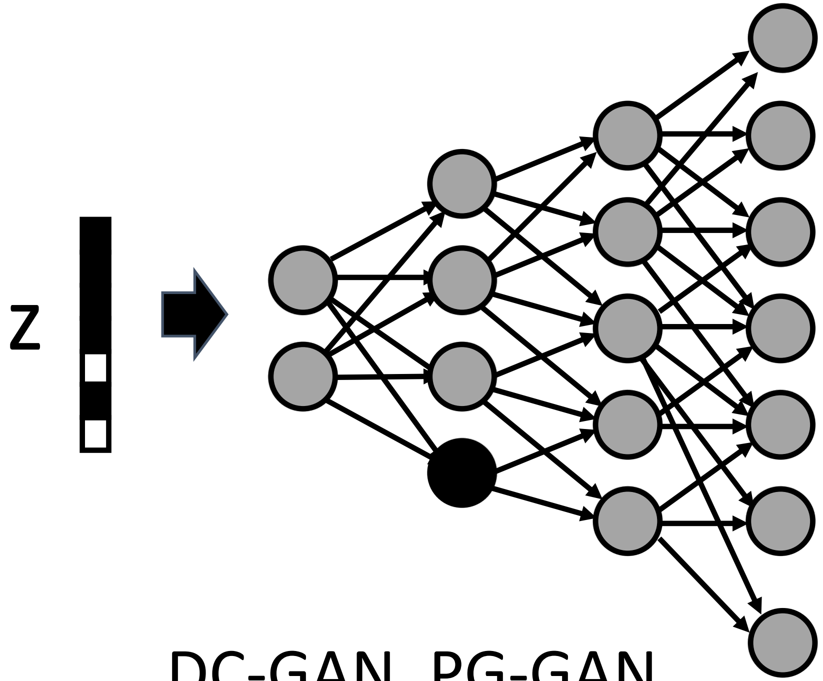
Optimized code



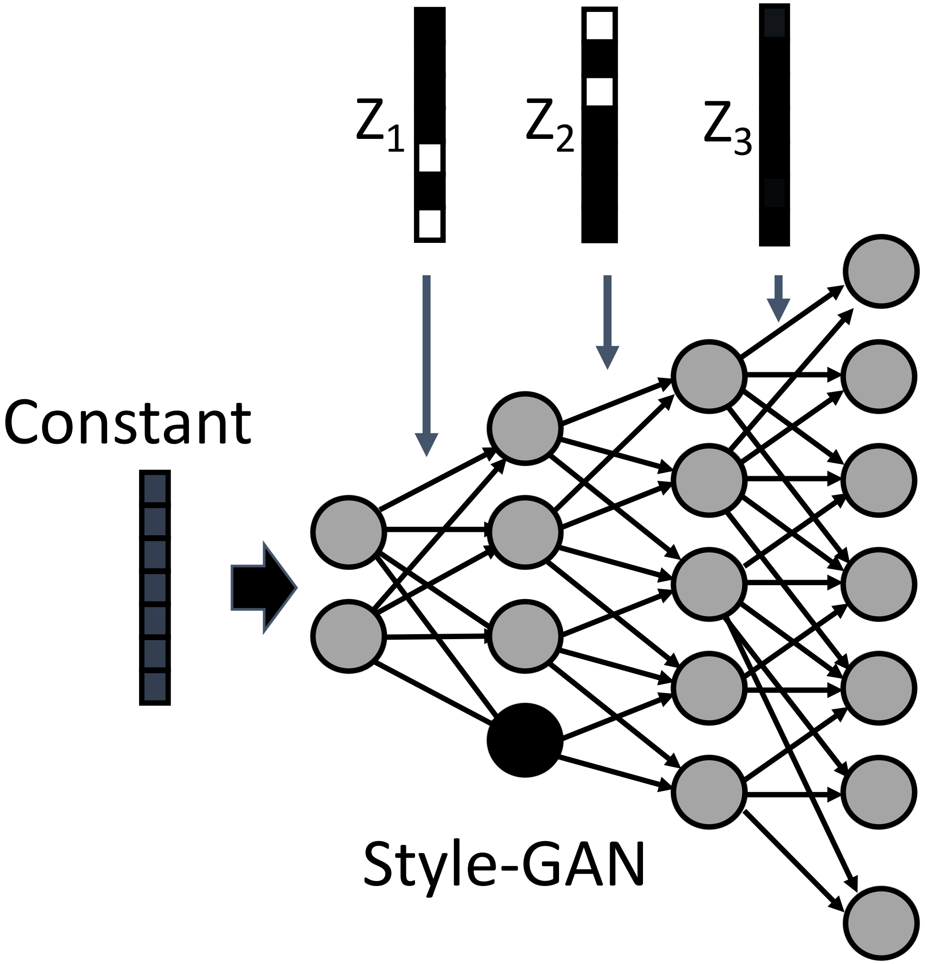
512 dimensions

Reconstructed face

Architectures of the generator in GANs



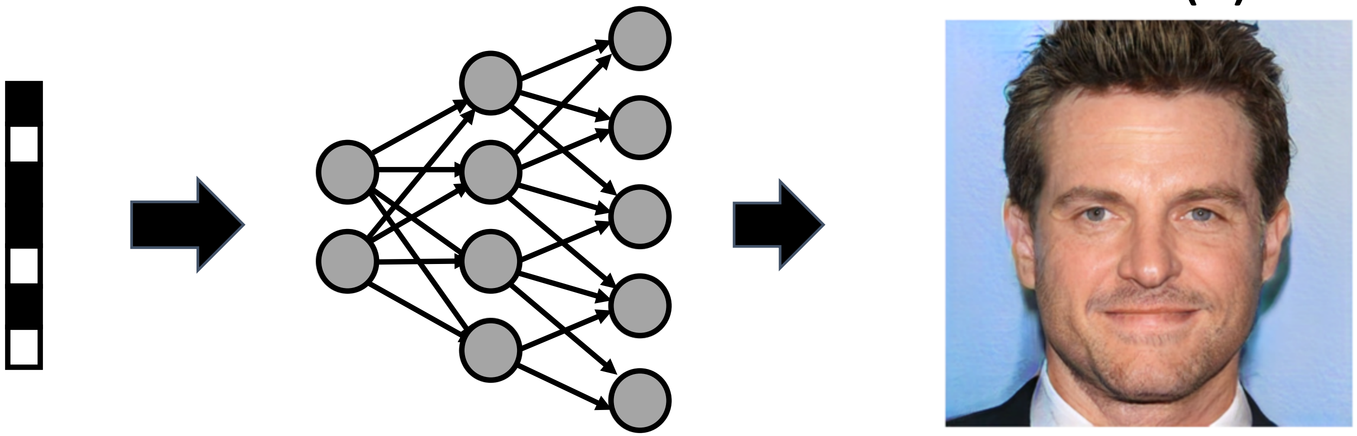
DC-GAN, PG-GAN



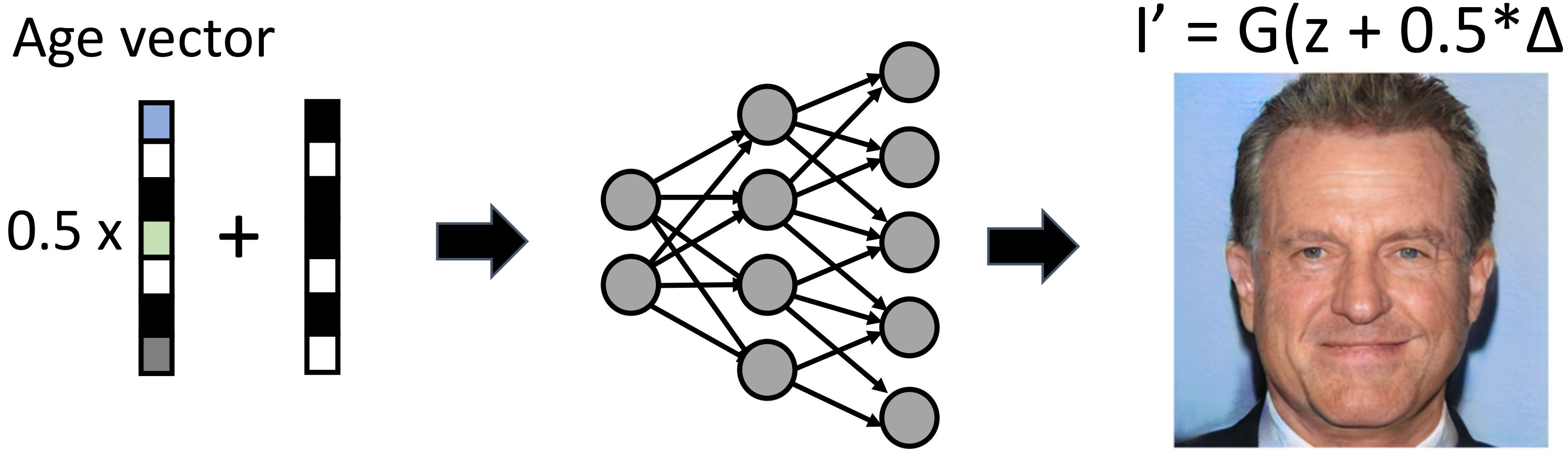
Style-GAN

Image Manipulation through Pretrained GANs

$I = G(z)$



$I' = G(z + 0.5 * \Delta z)$



InterFaceGAN, Shen, Gu, Tang, Zhou, CVPR’20

Deep Generative Prior for Image Processing

To optimize latent code z such that the following objective function can be minimized:

$$L_{SR} = L(down(x^{inv}), I_{LR})$$

$$L_{inp} = L(x^{inv} \circ \mathbf{m}, I_{ori} \circ \mathbf{m})$$

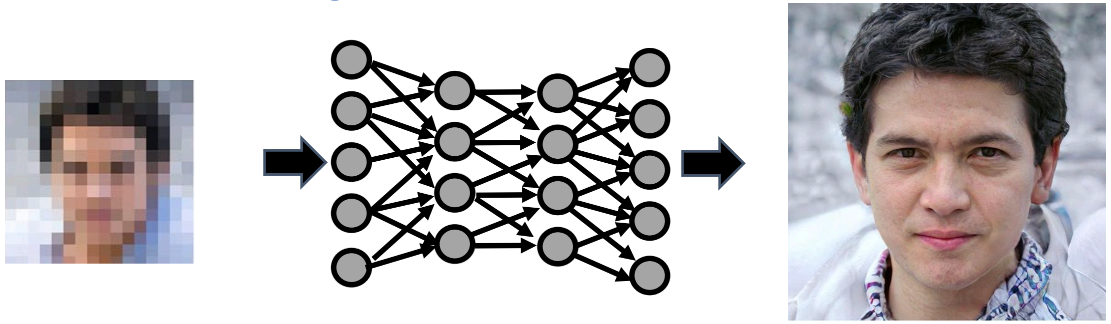
$$L_{color} = L(gray(x^{inv}), I_{gray})$$


(b) Colorization
(c) Super-Resolution
(e) Image inpainting
(f) Semantic image manipulation

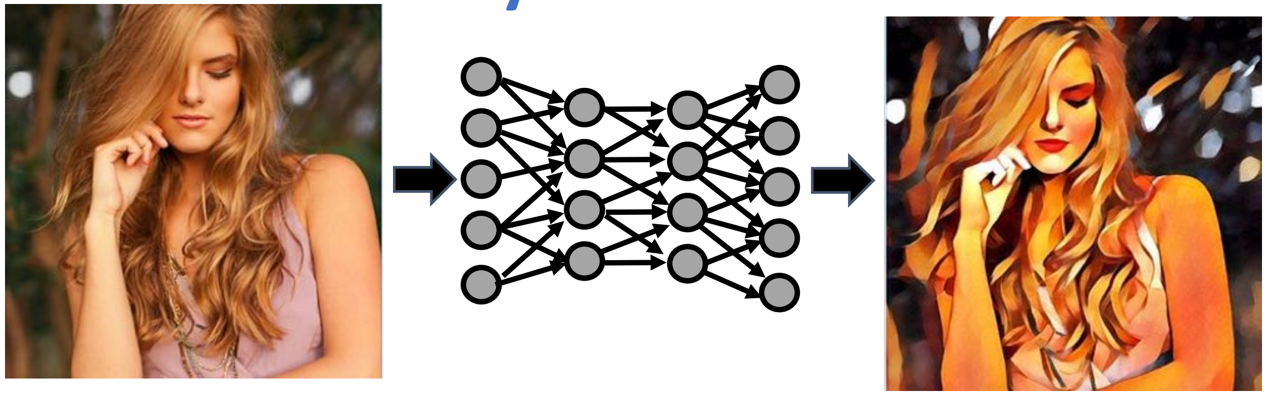
Jinjin Gu, Yujun Shen, Bolei Zhou. CVPR'20

Neural Image Processing

Super-resolution



Style transfer



Colorization

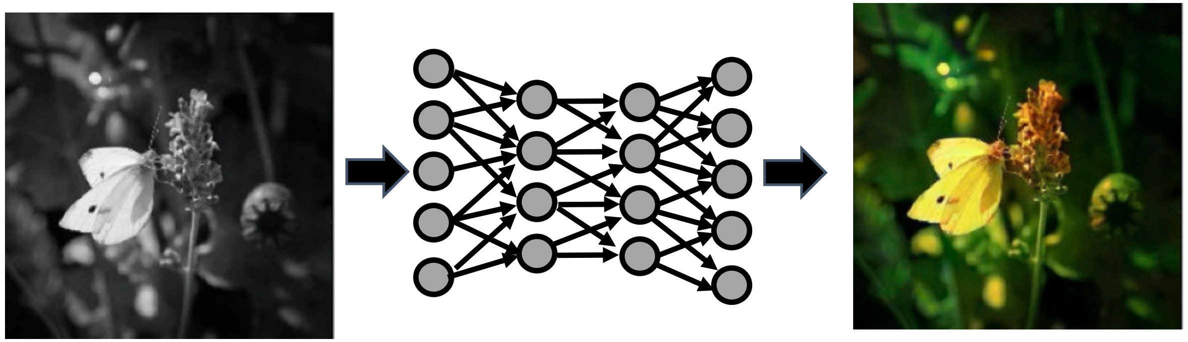
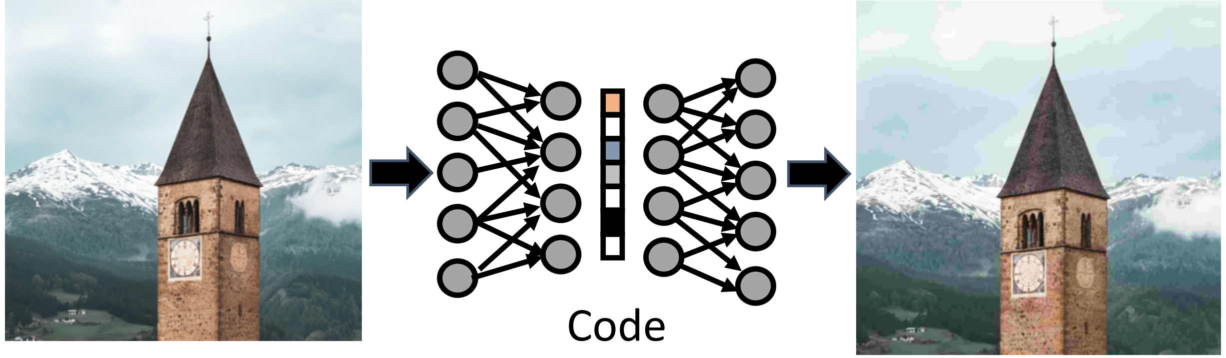
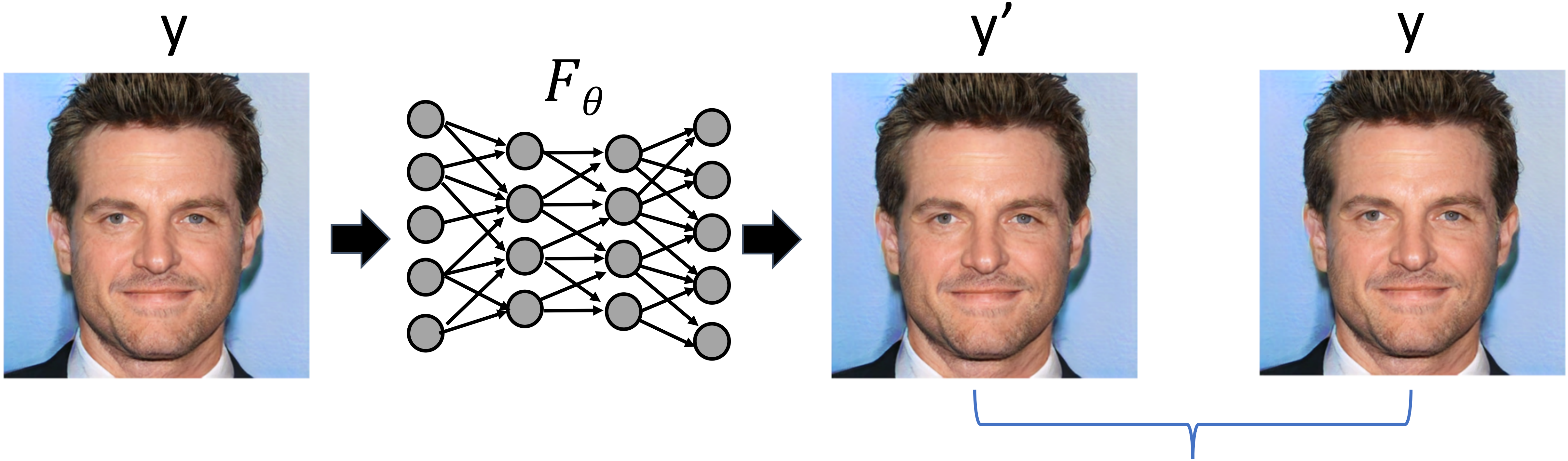


Image compression



Reconstruction Loss d(y’, y)

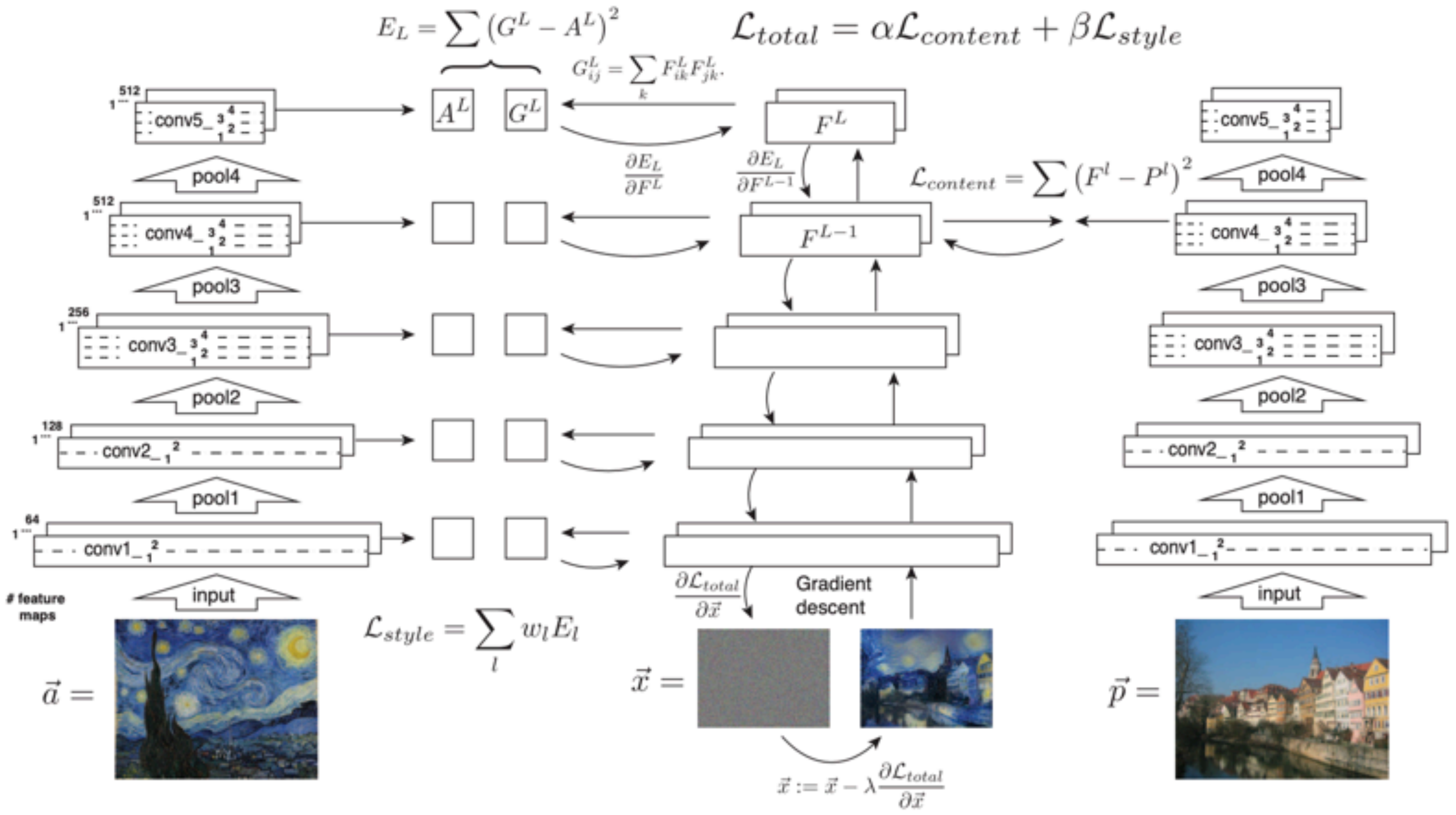


Per-pixel loss d(y’, y) = $||\, y' - y \,||^2$

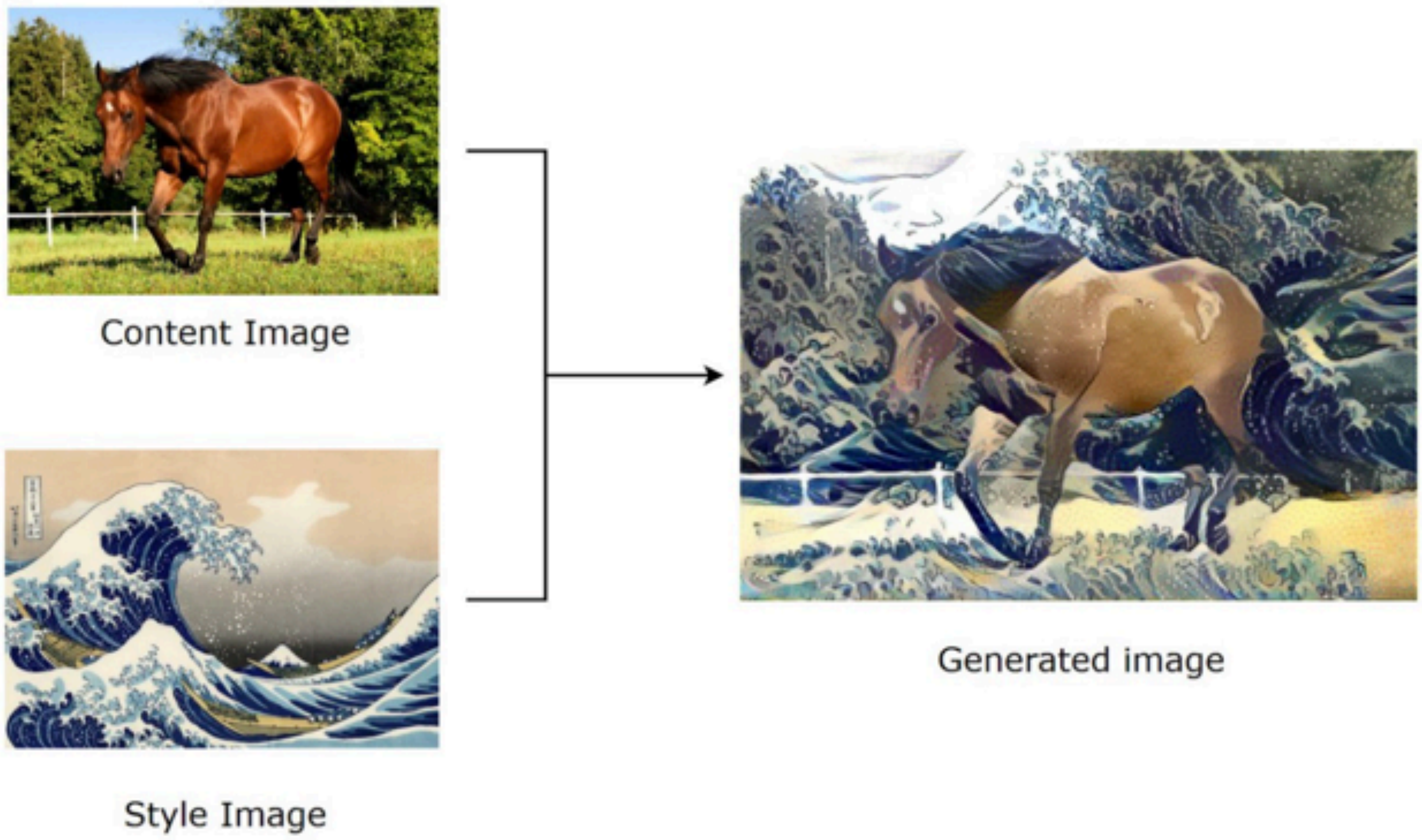
$$\theta^* = \mathrm{argmin}\, \|F_\theta(y) - y\|$$

Loss Functions for Image Reconstruction

- Per-pixel loss: pixel difference
- Feature loss: pretrained network's intermediate features
- Style loss: Gram Matrix

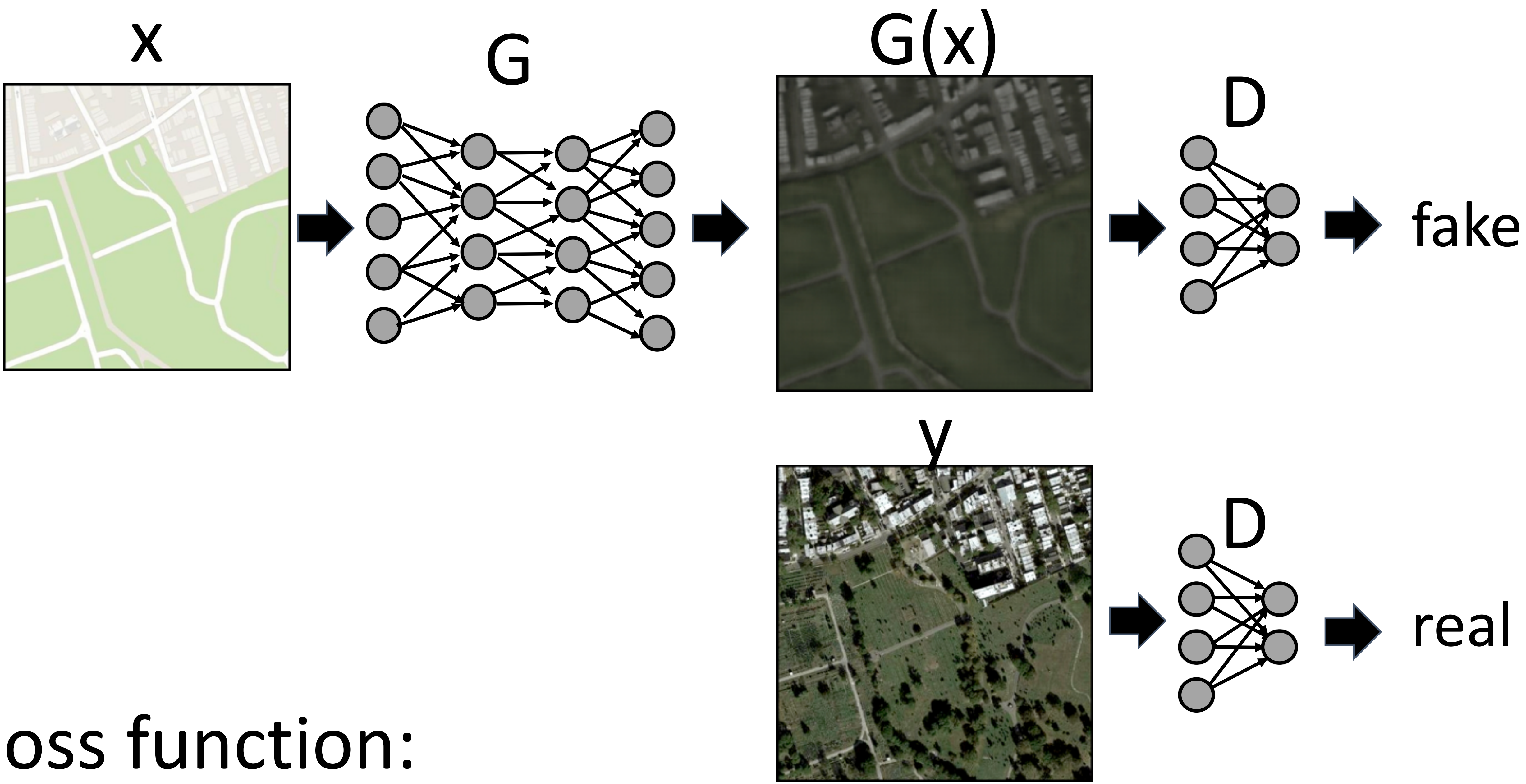


Neural Style Transfer



Metrics to Evaluate Image Reconstruction

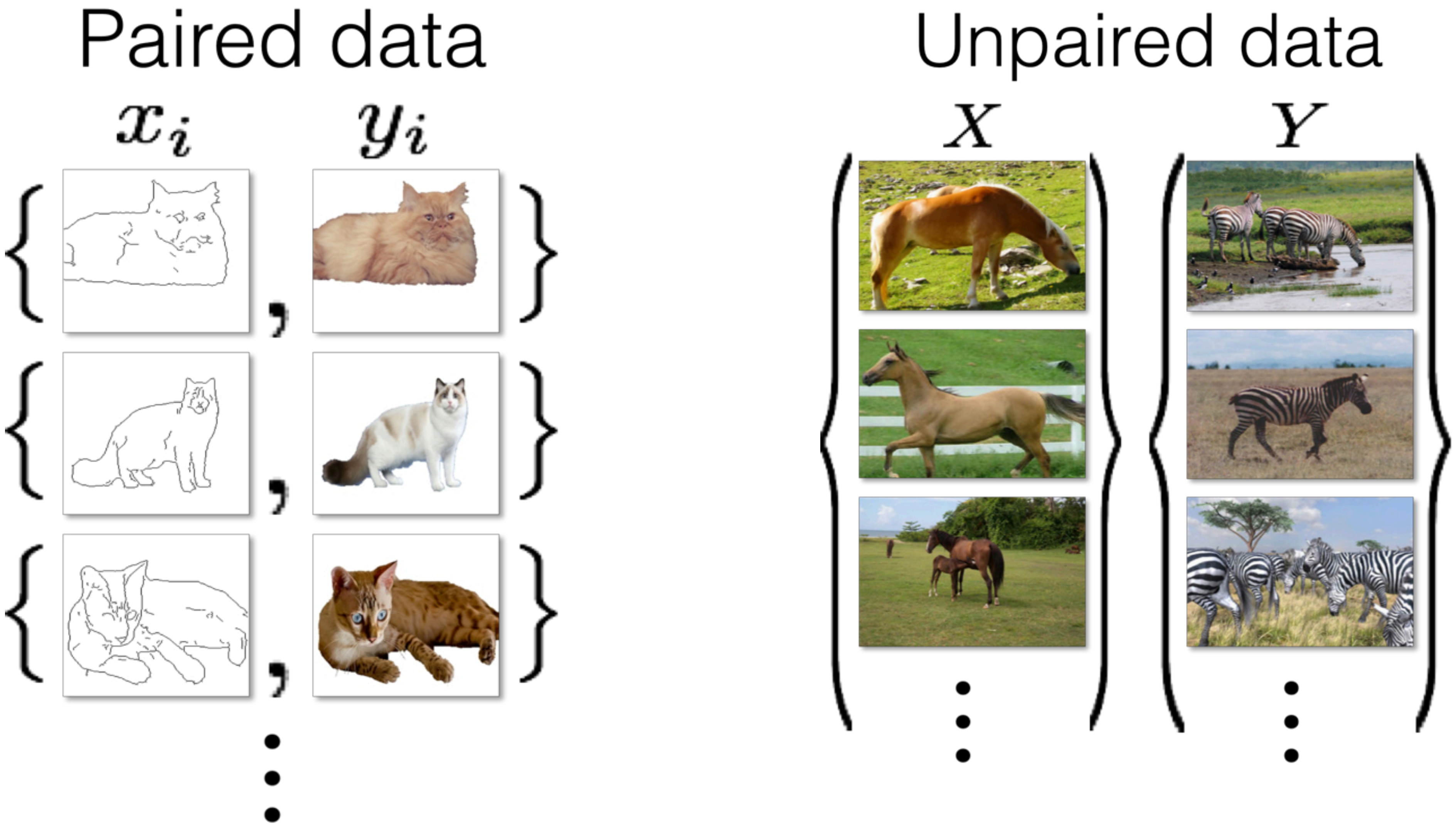
- Pairwise similarity:
  - PSNR
  - MS-SSIM
- Distribution similarity (commonly used in GANs):
  - FID



loss function:

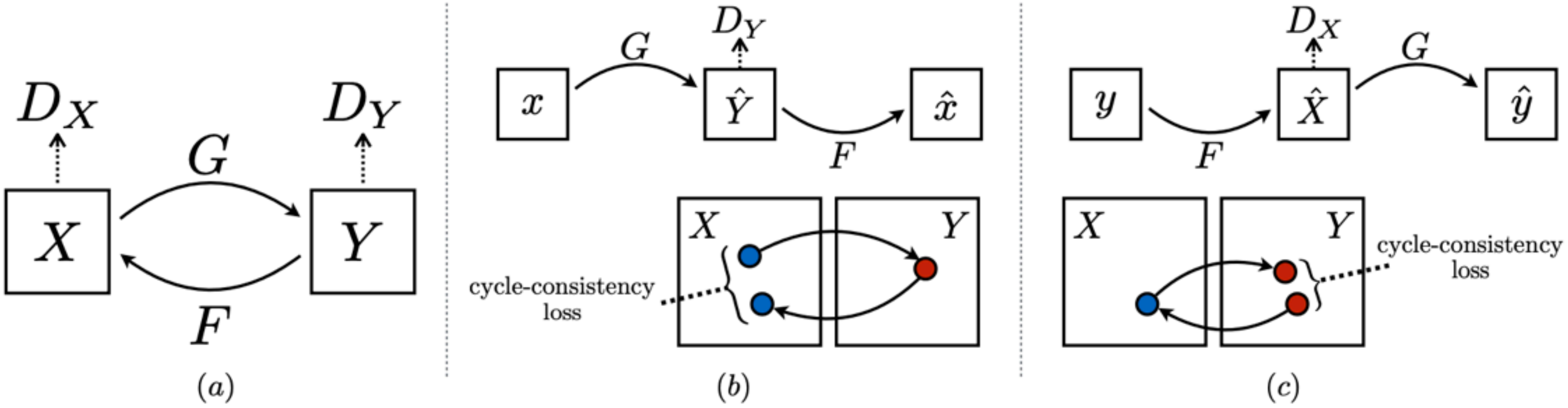
$$\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x})) \quad + \quad \log(1 - D(\mathbf{y}))\ \ ]$$

Cycle Consistency for Unpaired Data



slides from Phillip Isola

Cycle Consistency for Unpaired Data



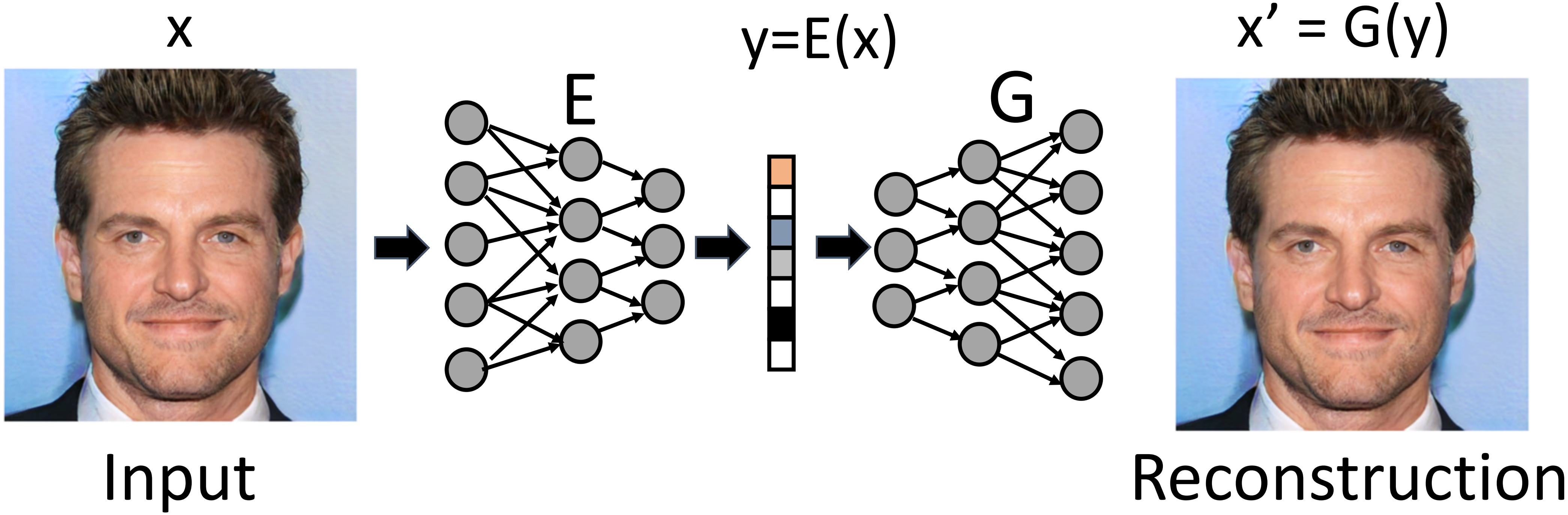
$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Our full objective is:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

Neural Image Compression with Adversarial Loss



Objective:

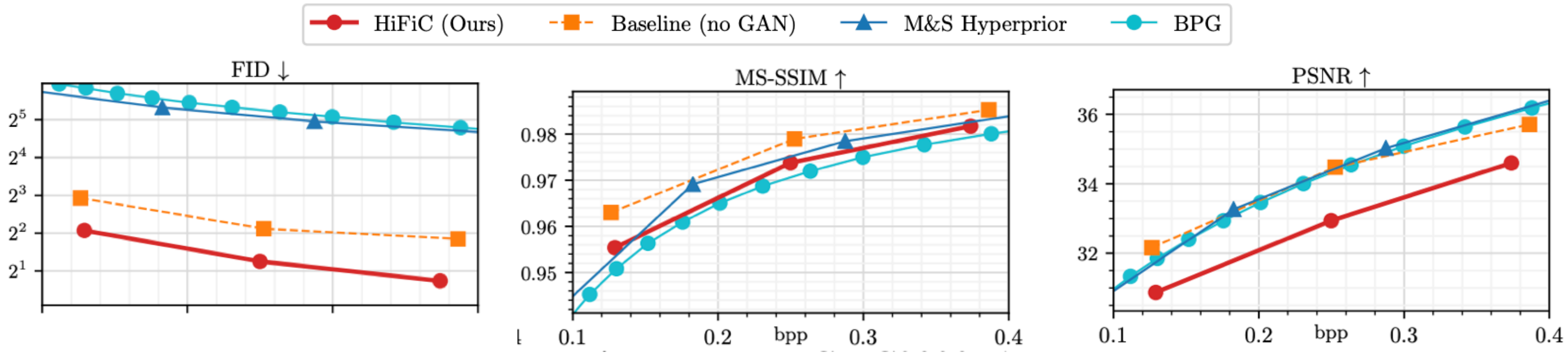
$$\mathcal{L}_{EGP} = \mathbb{E}_{x \sim p_X}[\lambda r(y) + d(x, x') - \beta \log(D(x', y))],$$
$$\mathcal{L}_D = \mathbb{E}_{x \sim p_X}[-\log(1 - D(x', y))] + \mathbb{E}_{x \sim p_X}[-\log(D(x, y))].$$

* P is the distribution of code y, then an entropy coding algorithm on y

Experiments of High-Fidelity Compression

- Training set: It consists of a large set of high-resolution images collected from the Internet
- Testing set:
  - Kodak [23] dataset (24 images),
  - CLIC2020 [46] testset (428 images)
  - DIV2K [2] validation set (100 images)

bpp: bit per pixel



Calculating bpp for an image

- Say we have a jpg image with 420x920, its size is 20635 bytes. What is the bpp in this image?
- We know 1 byte = 8 bits, bpp (bits per pixel) = 20635 * 8 bits / (420*920)
- Other knowledge about units:
  - 1 Mb = 1024 kb
  - 1 kb = 1024 bytes

Exam Coverage

- All the course slides, not limited to this summary slide
- Assignments 1-3
- Coding examples used in TA tutorial sessions
- Coding examples used in the course

Thank you!

- Hope this course will be useful for your future career!
- Let me know how you apply what you learn to your future projects!
- http://bzhou.ie.cuhk.edu.hk/