P9120 - Statistical Learning and Data Mining

Lecture 7 - Convolutional Neural Networks (CNN, ConvNet)

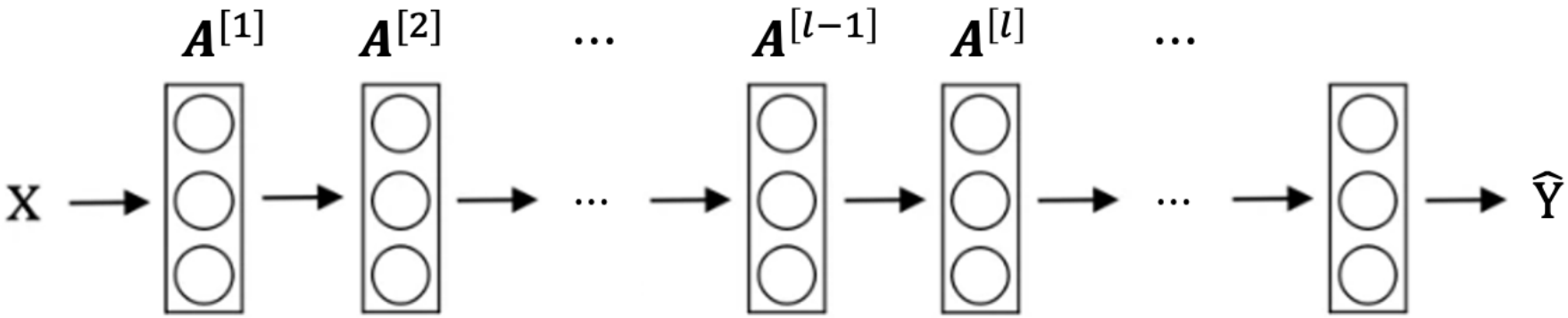
Min Qian

Department of Biostatistics, Columbia University

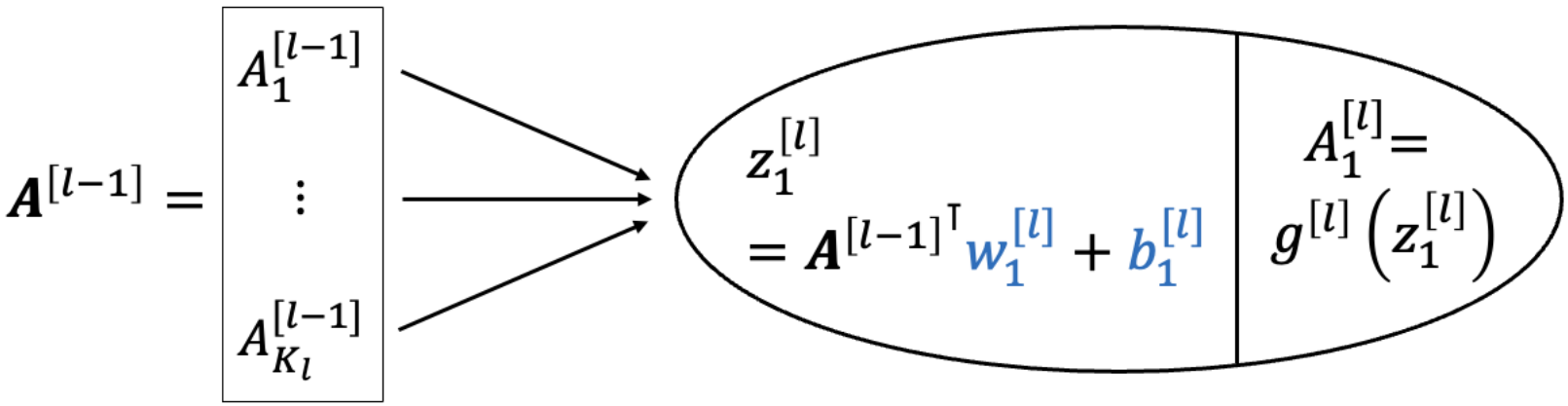
October 17th, 2024

Outline

A Generic Deep Neural Network Architecture

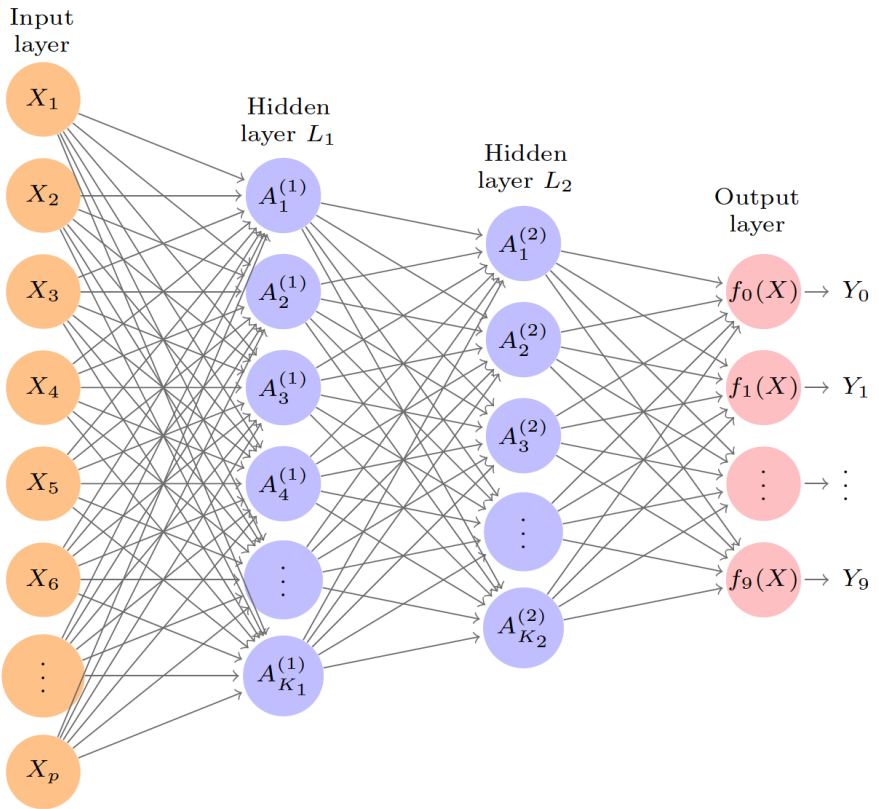


From Layer $[l-1]$ to first unit in layer $[l]$:



Parameters (weights) are learned by $\min \sum_{i=1}^{n} l(Y_i, \hat{Y}_i)$ using GD.

Image Classification



- Input: $p = 28 * 28 = 784$
- Output: $Y \in \{0, 1, \ldots, 9\}$.
- Sample size $n = 60,000$.
- Two hidden layers: $K_1 = 256, K_2 = 128$.
- Number of parameters:
  $(p+1) \times K_1 + (K_1+1) \times K_2 + (K_2+1) \times 10 = 235,146$.

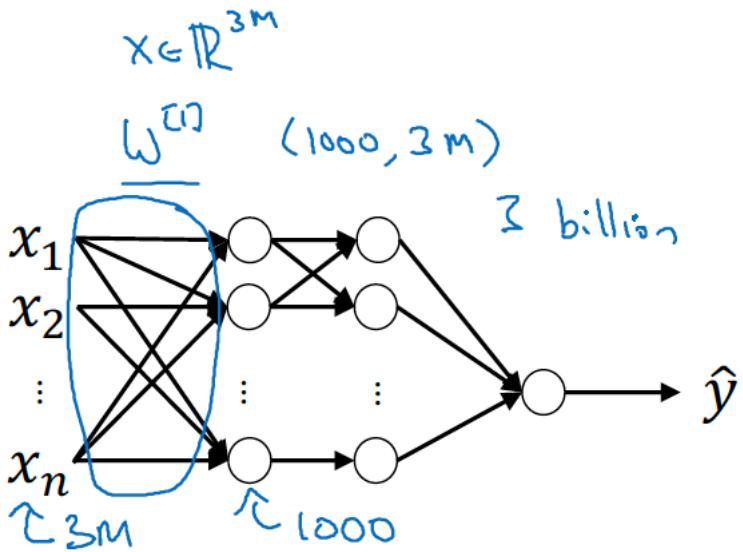
Deep Learning on large images

64x64 × 3

Cat? (0/1)

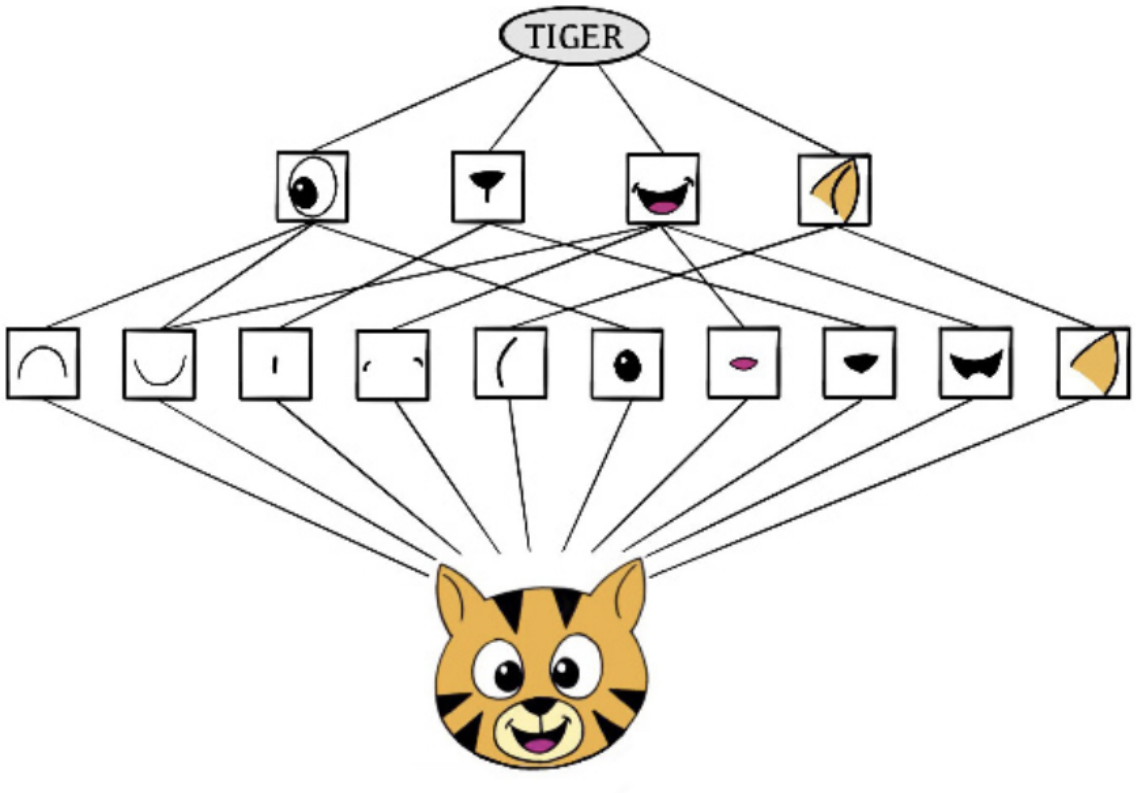
12288

1000 × 1000 × 3
= 3 million



Andrew Ng

Convolutional Neural Networks

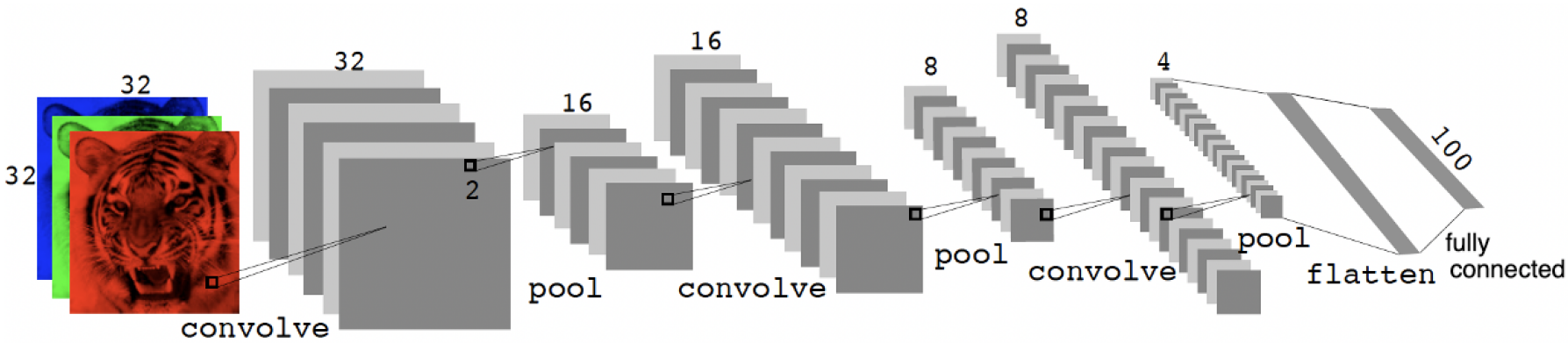


- Identify local features (edges).
- Combine local features to create compound features (eyes and ears).
- Use compound features to output the label “tiger”.

Architecture of a CNN

Types of layers:

- **Convolution !!!**
- Pooling
- Fully connected



- Input: 3-d feature map: 3 channels (colors), each of 32×32 pixels.
- Output figure label: 100 categories of animals

Convolution Layers

$$\text{Original Image} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \\ j & k & l \end{bmatrix} \qquad \text{Convolution Filter} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$$

When we *convolve* the image with the filter, we get the result

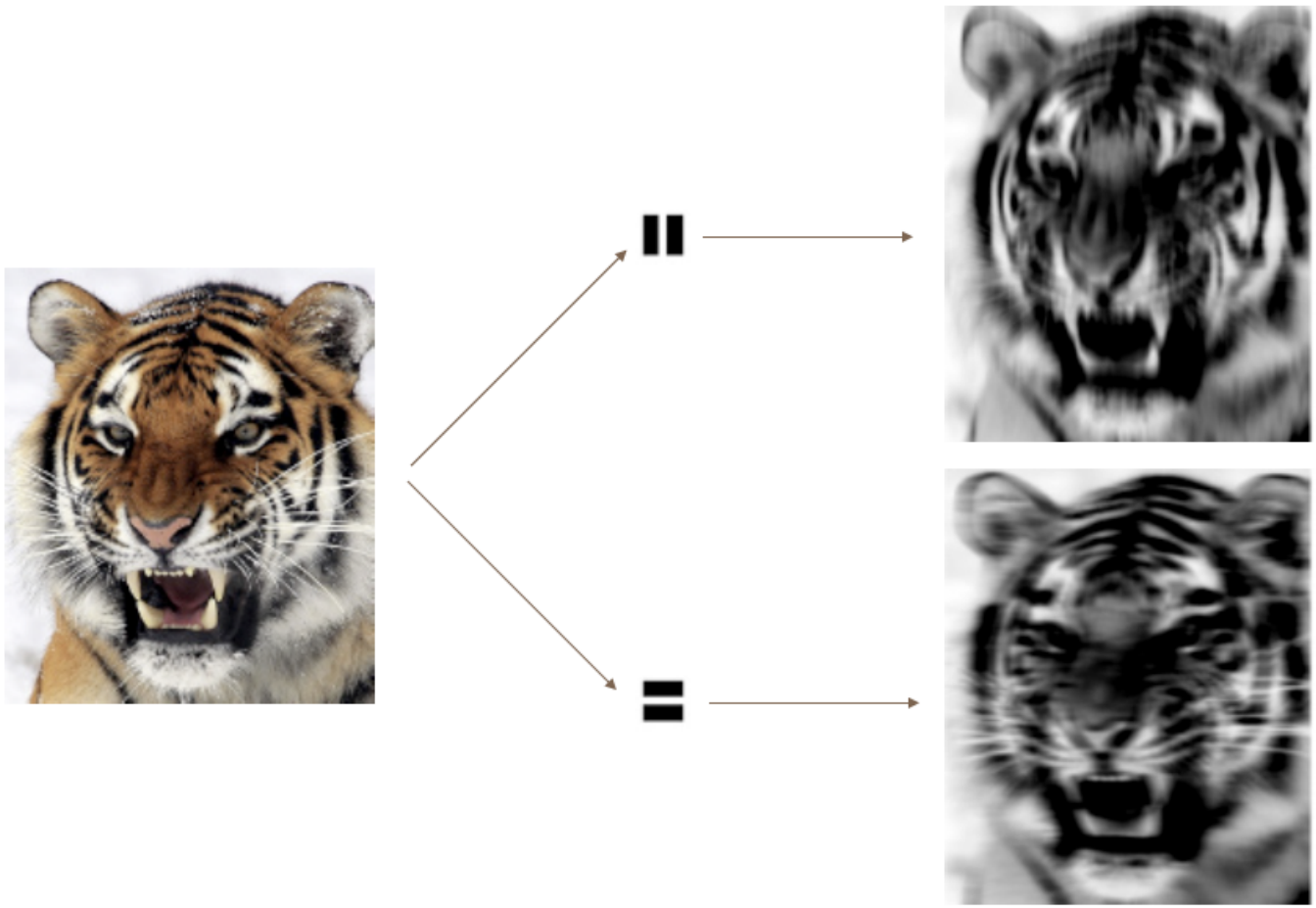
$$\text{Convolved Image} = \begin{bmatrix} a\alpha + b\beta + d\gamma + e\delta & b\alpha + c\beta + e\gamma + f\delta \\ d\alpha + e\beta + g\gamma + h\delta & e\alpha + f\beta + h\gamma + i\delta \\ g\alpha + h\beta + j\gamma + k\delta & h\alpha + i\beta + k\gamma + l\delta \end{bmatrix}$$

where $(\alpha, \beta, \gamma, \delta)$ in the convolution filter are parameters.

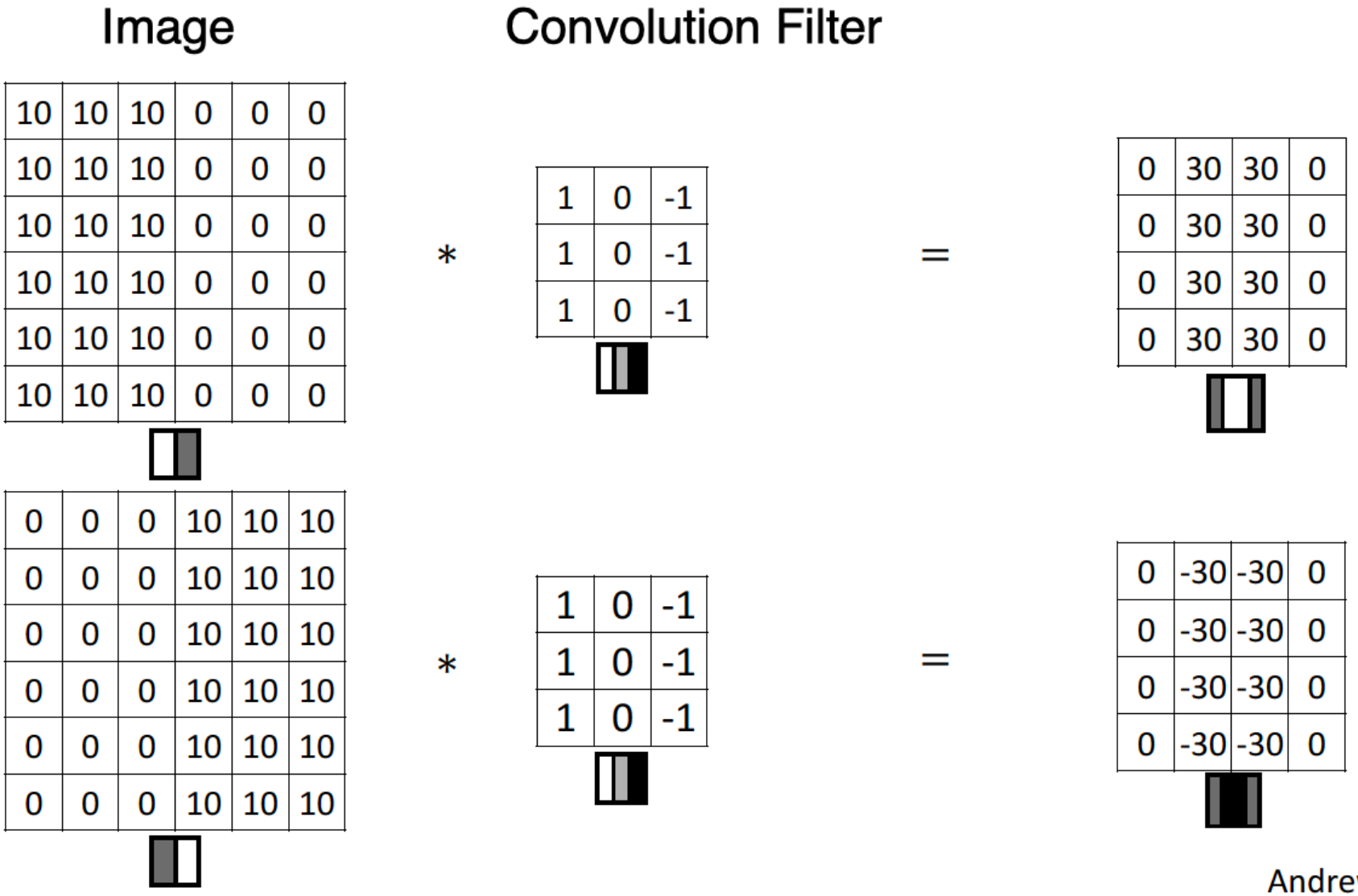
The convolved image highlights regions of the original image that resemble the convolution filter.

Convolution Filter in The Tiger Example

Applying two convolution filters to a 192×179 image of a tiger

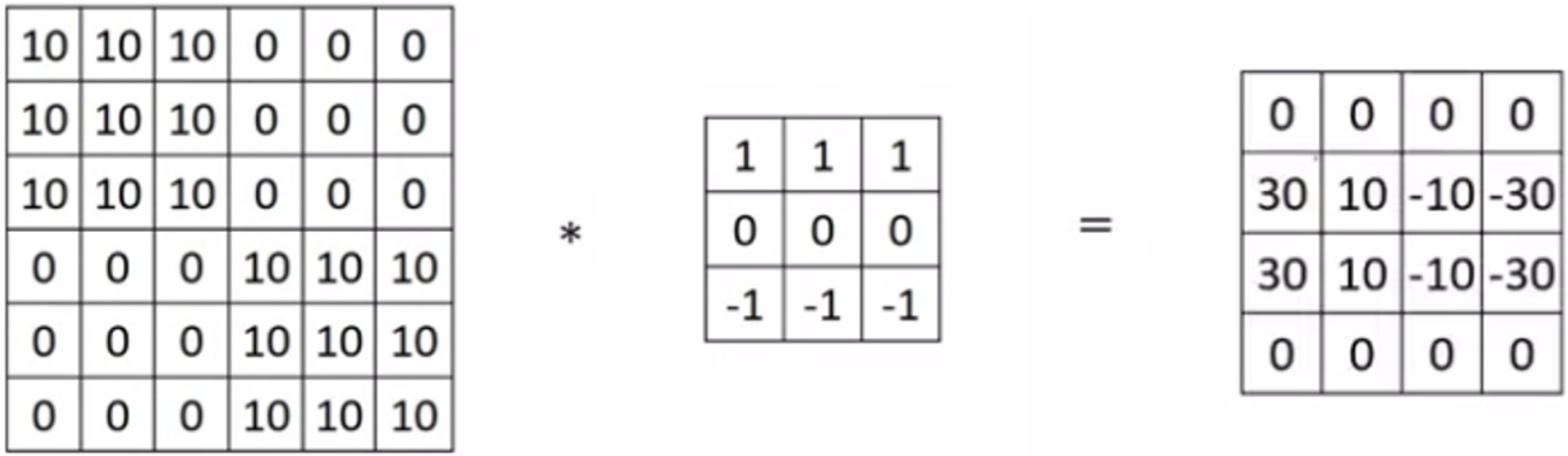


Example Convolution Filter for Vertical Edge Detection



More on Edge Detection

- Convolution filter to detect horizontal edges



- In general, learn the filter to detect edges

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Padding

Without padding,

- shrinking output - problematic for deep NN with many layers of filters.
- Throwing away information from edge/corner of an image.

Padding:

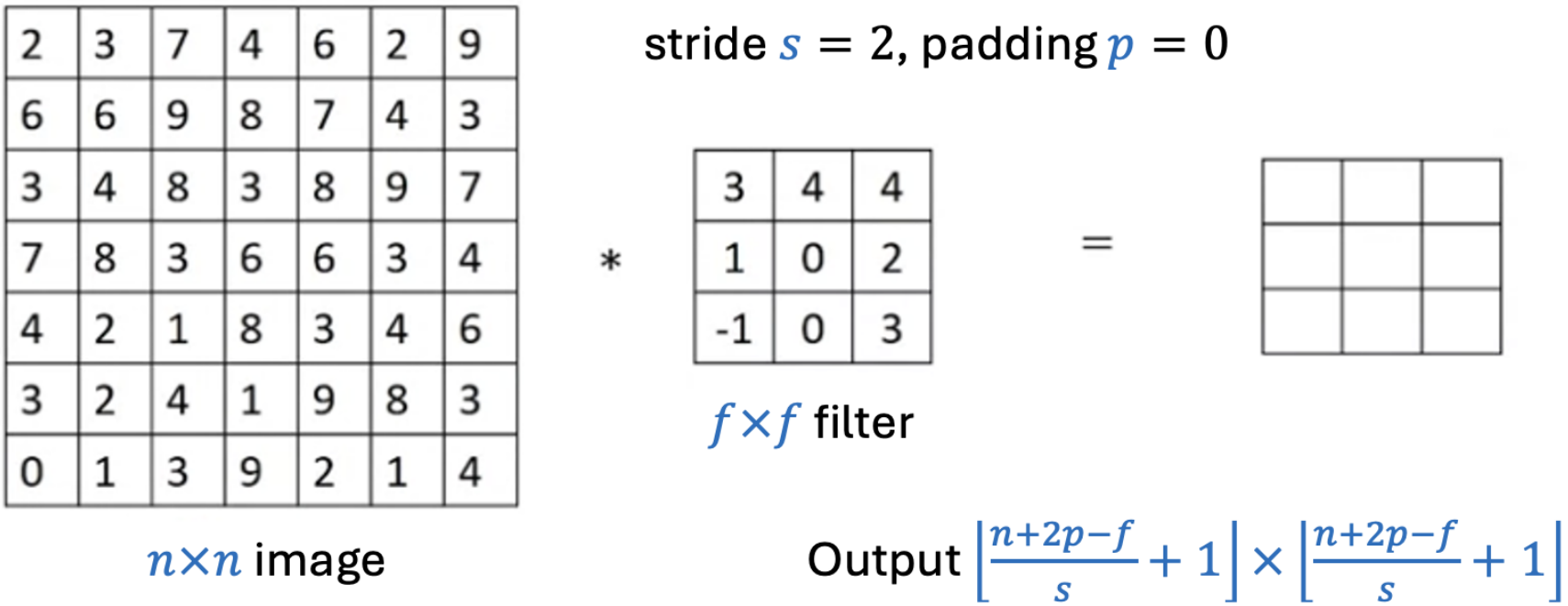
- Add p column/row of zeros at each side of the image: $n \times n$ image $\rightarrow (n+2p) \times (n+2p)$ image.
- Applying an $f \times f$ filter to an $n \times n$ image with padding amount p results in a $(n+2p-f+1) \times (n+2p-f+1)$ output (image).

How much to pad: Valid vs Same convolutions

- "Valid": no padding $p = 0$.
- "Same": choose p so that $2p - f + 1 = 0$ (f is usually odd).

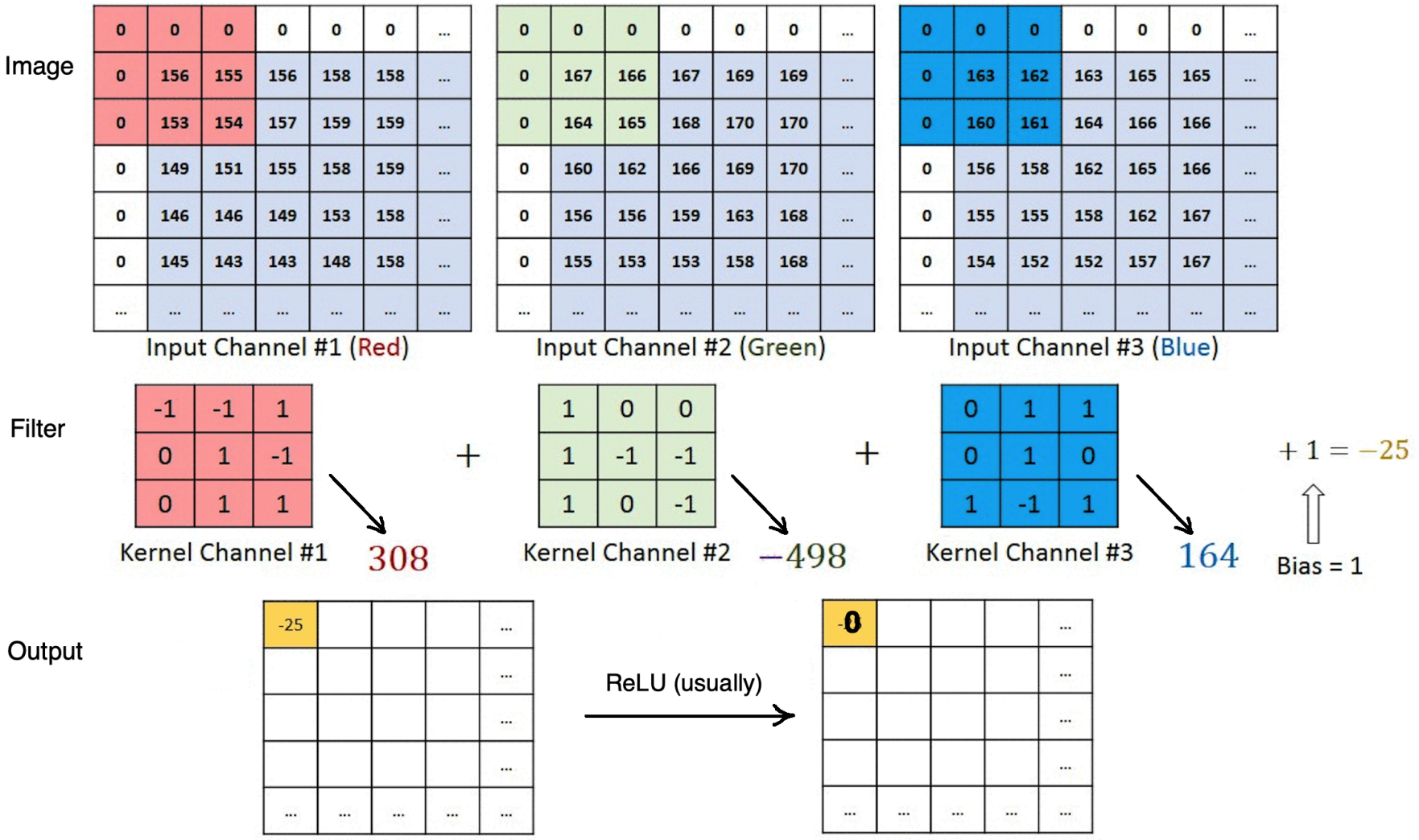
Strided Convolution

Stride variable controls the step of the filter moving over the input.

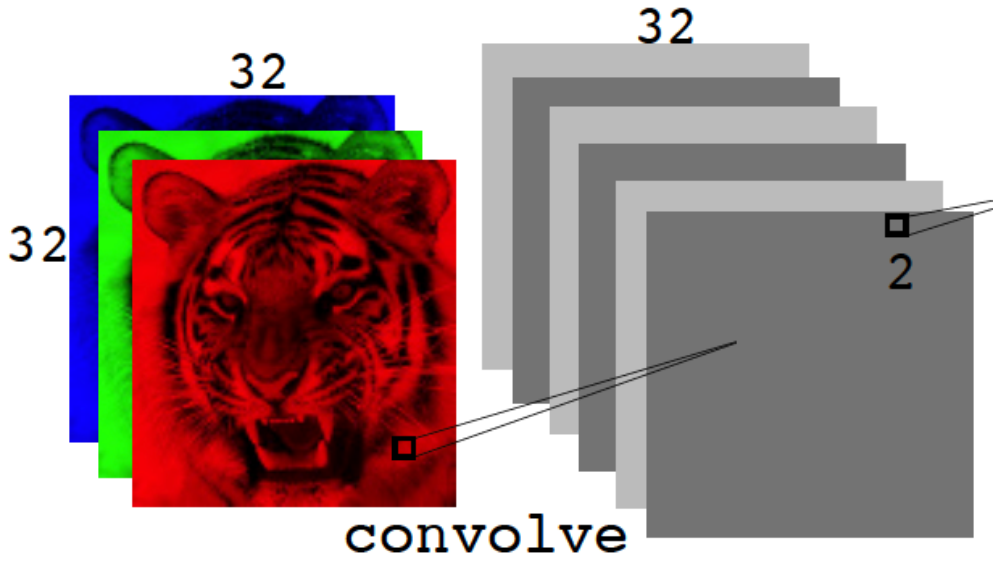


- Downsampling pushes the network to focus on the most discriminative features, ignoring redundant information.
- Reduces computational complexity.

One Convolution Filter on RGB Image



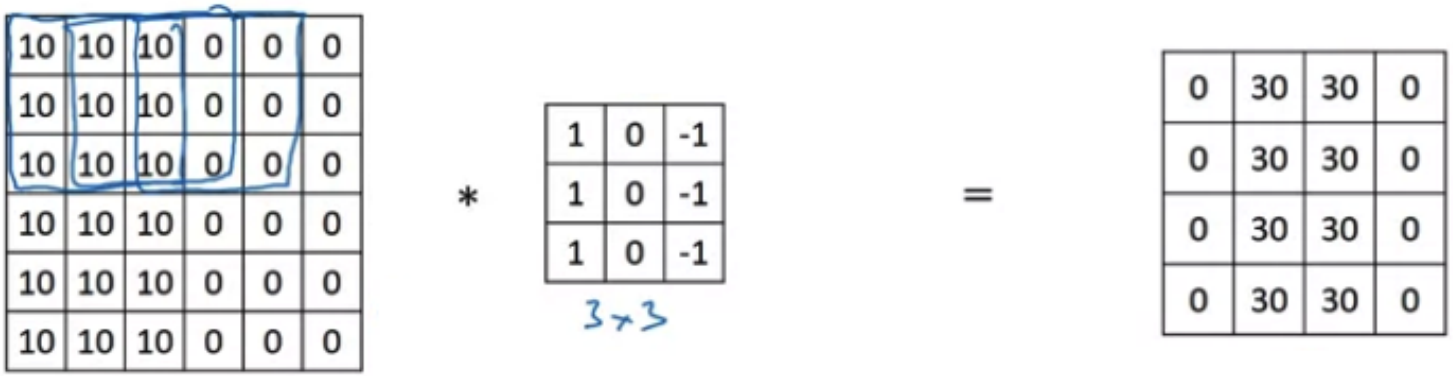
Example of a Convolution Layer: Input → Convolution



- Padding the edges of the input image.
- Apply 6 convolution filters. Each filter contains three channels, one per color, each of 3×3 with potentially different filter weights.
- Each filter yields a 2-d output feature map.
- (Usually) apply a ReLU activation to the convolved image.

What is the number of parameters?

Why convolutions



Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Convolution Layer Summary of Notations

For a convolution layer-l with input volume $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$, need to decide

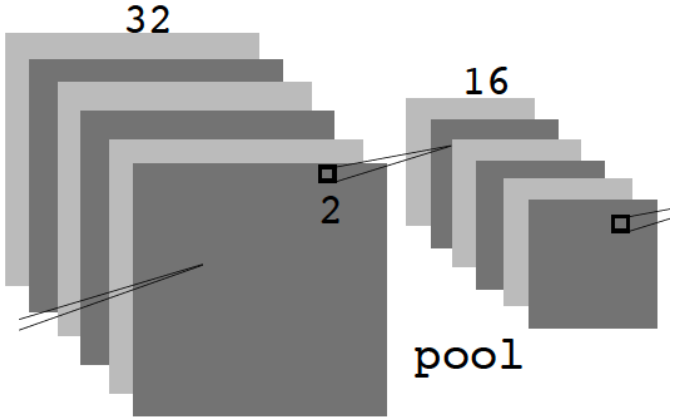
- filter size: $f^{[l]} \times f^{[l]}$ ($n_C^{[l-1]}$ channels)
- padding: $p^{[l]}$
- stride: $s^{[l]}$
- number of filters: $n_C^{[l]}$

Parameters (weights + intercept/bias):

$$f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]} + 1 \times n_C^{[l]}$$

Output: $\left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times n_C^{[l]}$

Pooling Layer: Max Pooling



- A pooling layer condenses a large image into a smaller pooling summary image.
- max pooling operation (with $f = 2, s = 2$): summarizes each non-overlapping 2×2 block of pixels in an image using the maximum value in the block:

$$\text{Max pool} \begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 1 & 3 & 4 \\ 1 & 1 & 2 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 5 \\ 2 & 4 \end{bmatrix}$$

Why Pooling?

- The pooling layer provides some location invariance: as long as there is a large value in one of the four pixels in the block, the whole block registers as a large value in the reduced image.
- Dispose of unnecessary information or features.

The feature after it has been pooled will be detected by the network, despite differences in its appearance between the three images.

Pooling Layer Summary of Notations

Hyper-parameters to decide

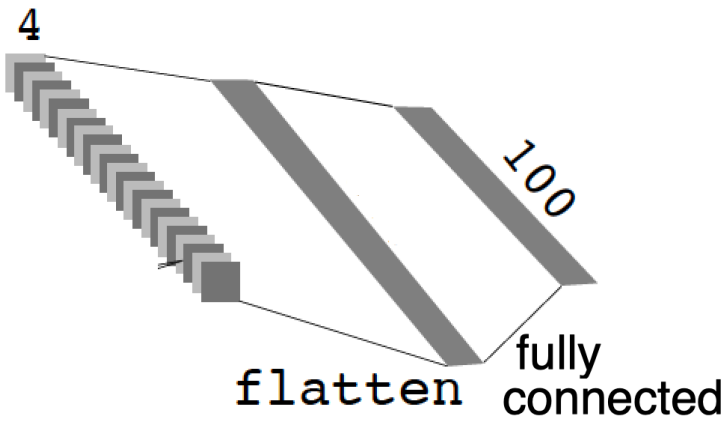
- Filter size $f \times f$ (common choices: 2×2 or 3×3)
- Stride s (common choice: $s = 2$)
- max or average pooling (common choice: max)

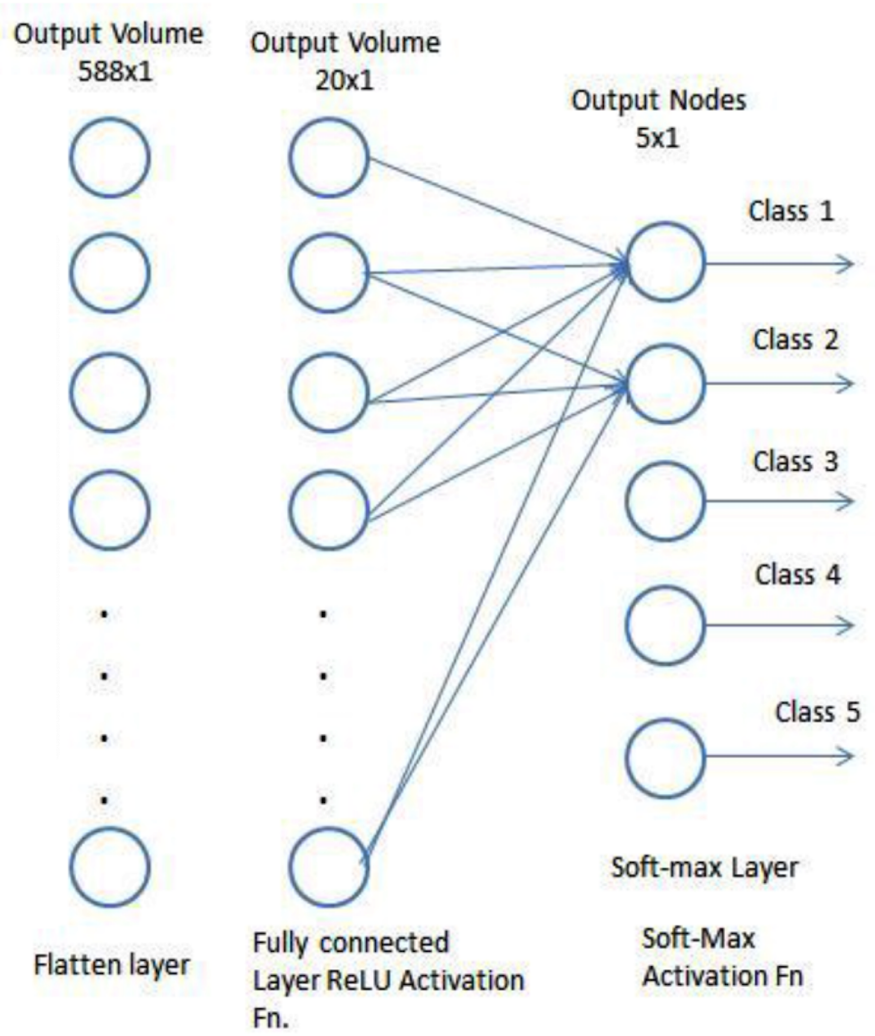
Input $n_H \times n_W \times n_C \rightarrow$ output $\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$

No parameters to learn!

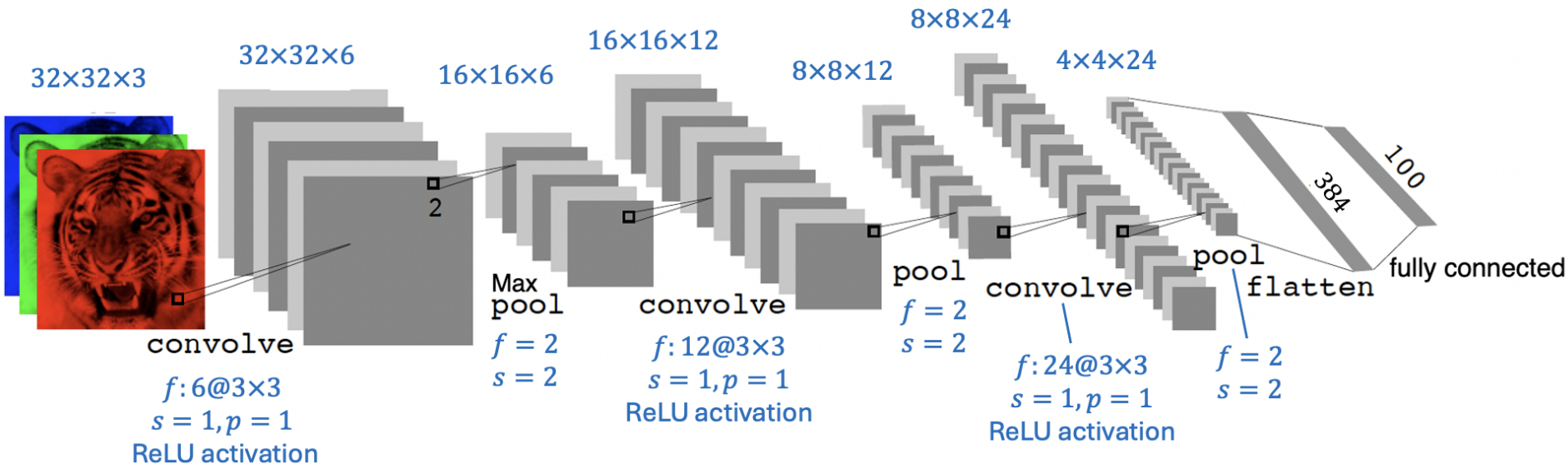
Fully Connected Layer

Flatten and then Feed Forward NN



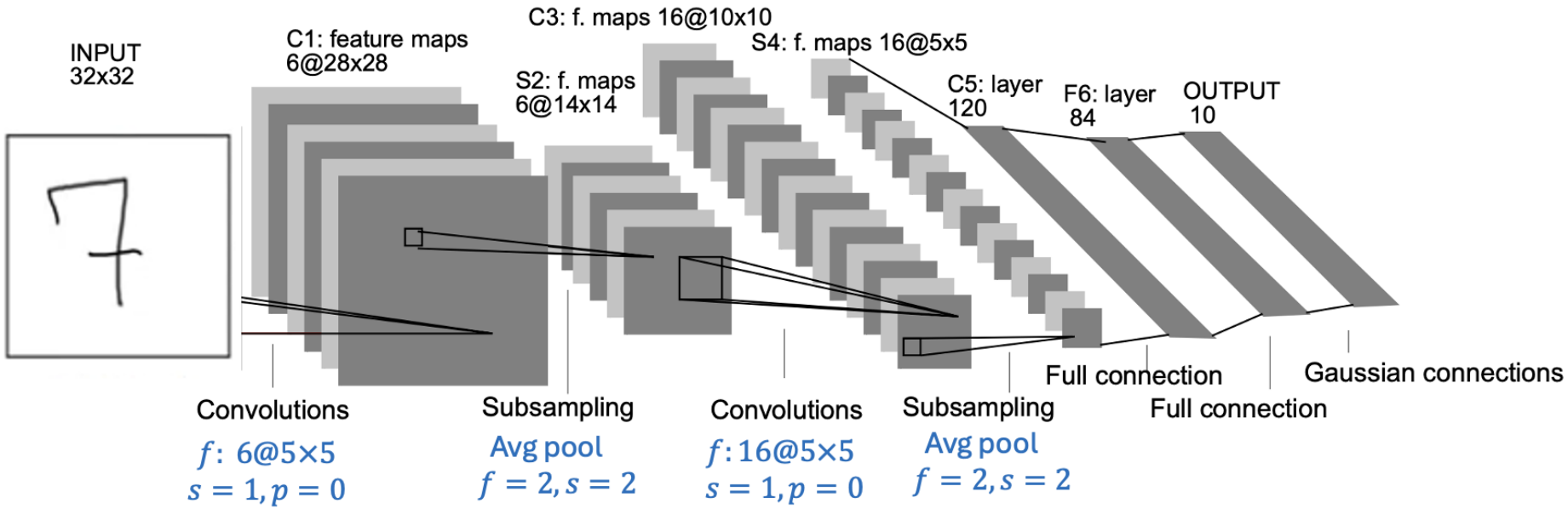


CNN Example


32×32×3
32×32×6
16×16×6
16×16×12
8×8×12
8×8×24
4×4×24
2
100
384
convolve
f: 6@3×3
$s = 1, p = 1$
ReLU activation
Max pool
$f = 2$
$s = 2$
convolve
f: 12@3×3
$s = 1, p = 1$
ReLU activation
pool
$f = 2$
$s = 2$
convolve
f: 24@3×3
$s = 1, p = 1$
ReLU activation
pool
$f = 2$
$s = 2$
flatten
fully connected

LeNet-5 (LeCun et al. 1998)

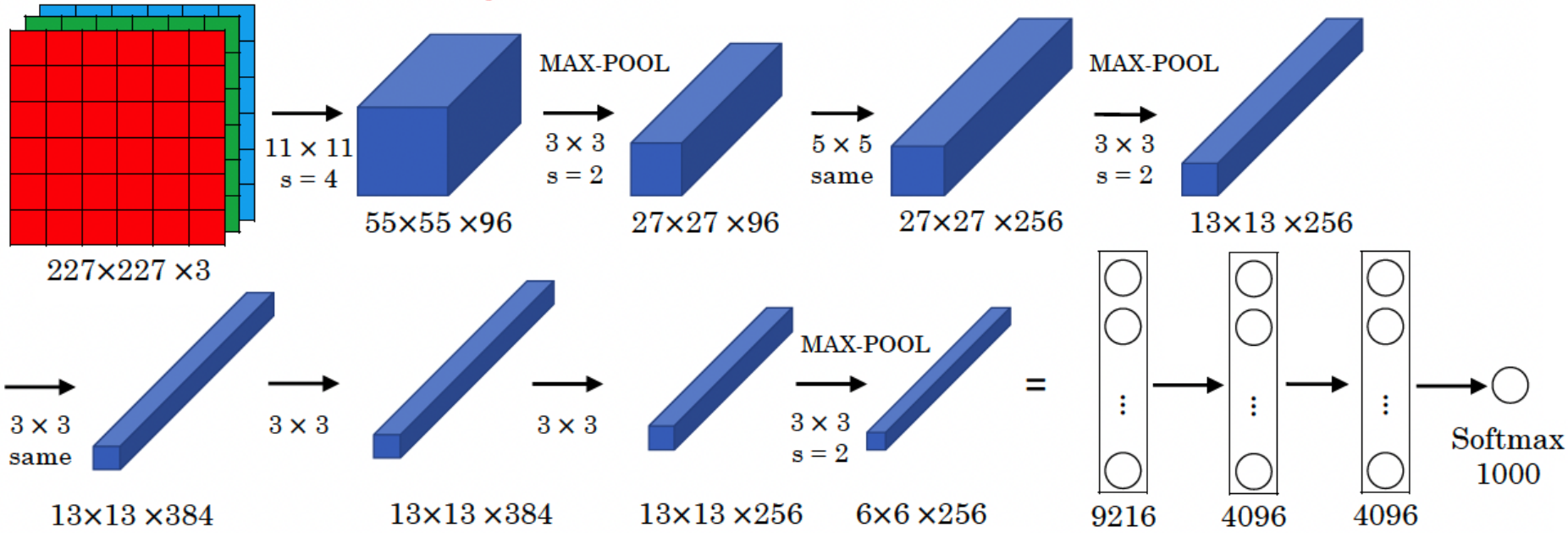
- First CNN architecture
- Goal: handwritten digits recognition



- sigmoid/tanh activation function was used
- $\sim 60K$ parameters

AlexNet (Alex Krizhevsky, et al. 2012)

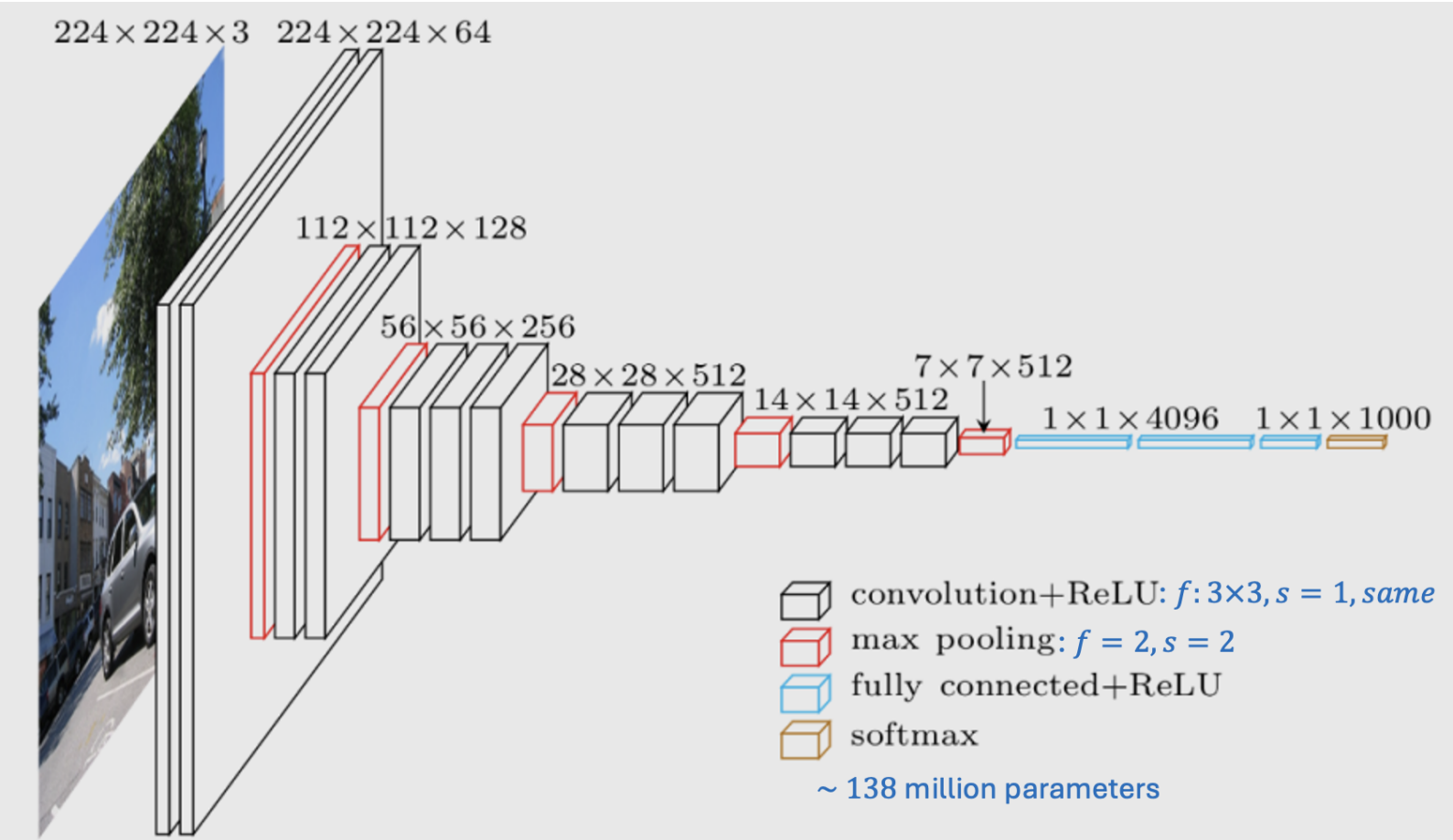
- The first architecture that used GPU.
- ImageNet Challenge: a subset of ImageNet of around 1000 images in each of 1000 categories. Roughly 1.3 million training images, 50,000 validation images and 100,000 testing images.
- Dominated the 2012 ImageNet challenge. DL became popular.



- ReLU activation function was used
- $\sim 60M$ parameters

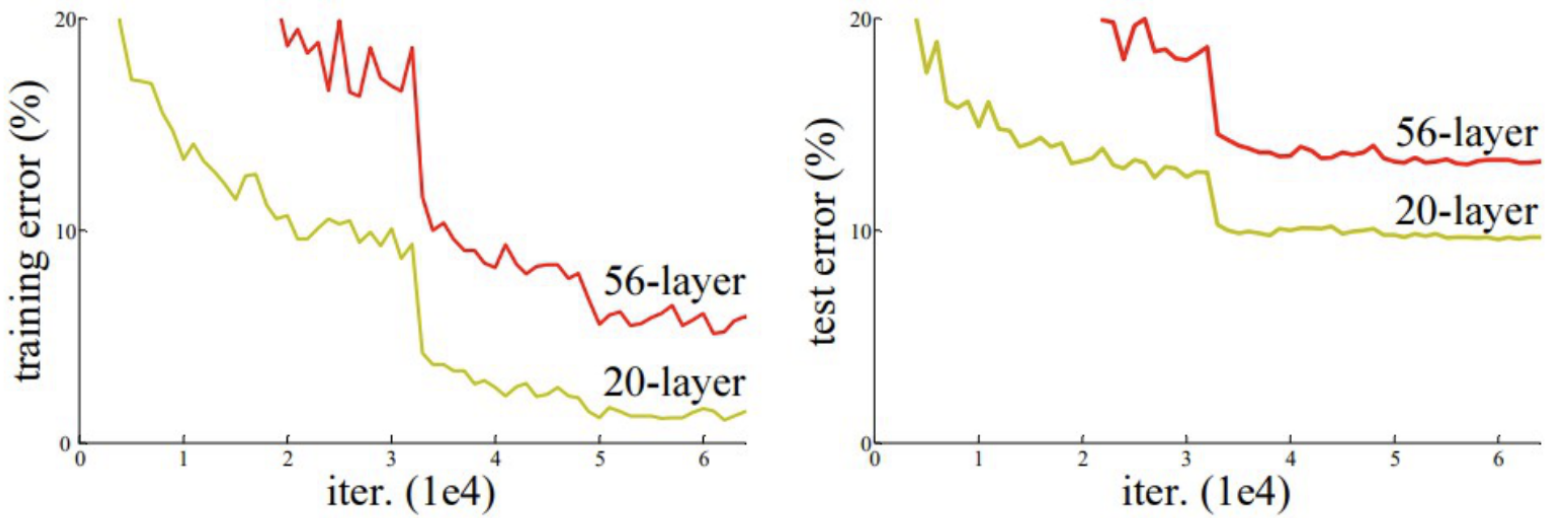
VGG-16 (Simonyan and Zisserman 2015)

- Visual Geometry Group (at Oxford), 16 layers with parameters.
- VGG-16 won the 1st and 2nd place in object detection and classification of 2014 ImageNet Challenge.
- Many other models were built on top of VGGNet or based on the 3×3 conv idea of VGGNet since then.



Problem with Very Deep Networks

- Vanishing/exploding gradient problem in deep learning
- Both training and test errors increase as the number of layers becomes too large

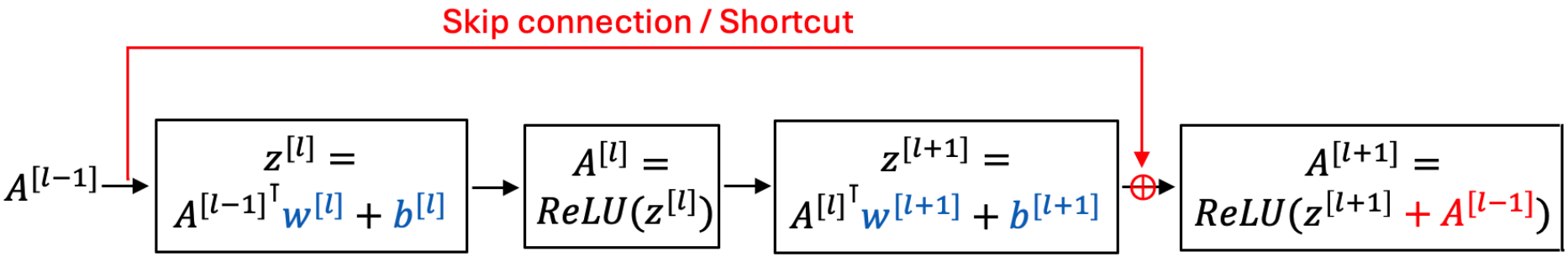


Comparison of 20-layer vs 56-layer architecture

Evidence shows that the best ImageNet models using convolutional and fully-connected layers typically contain between 16 and 30 layers.

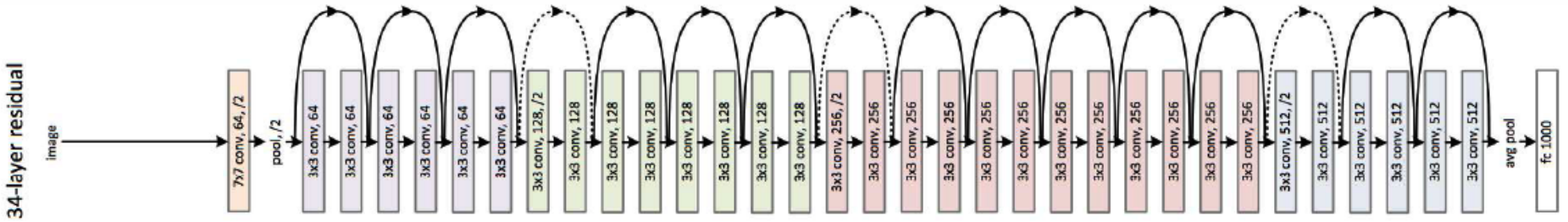
Residual Networks (ResNets, He et al. 2015)

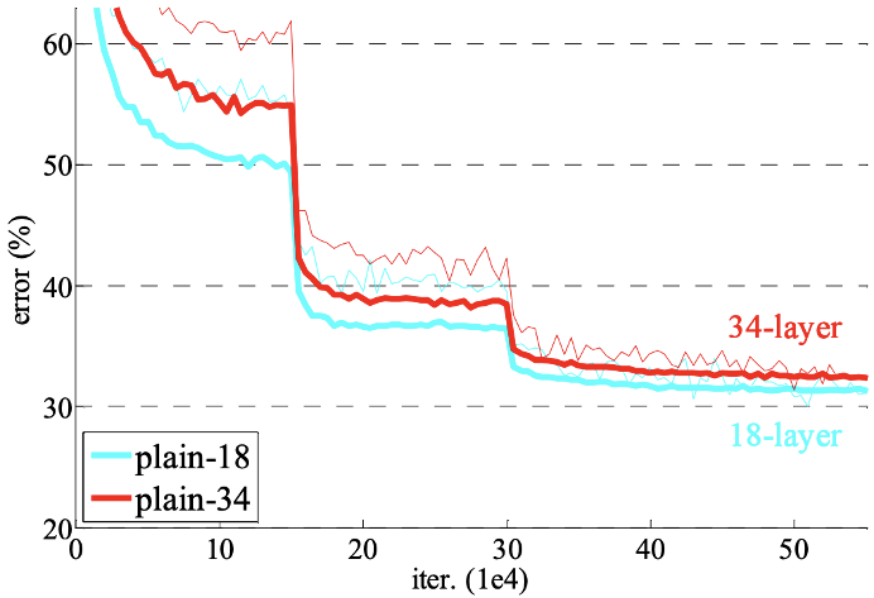
Consider from layer $[l-1] \to [l+1]$:

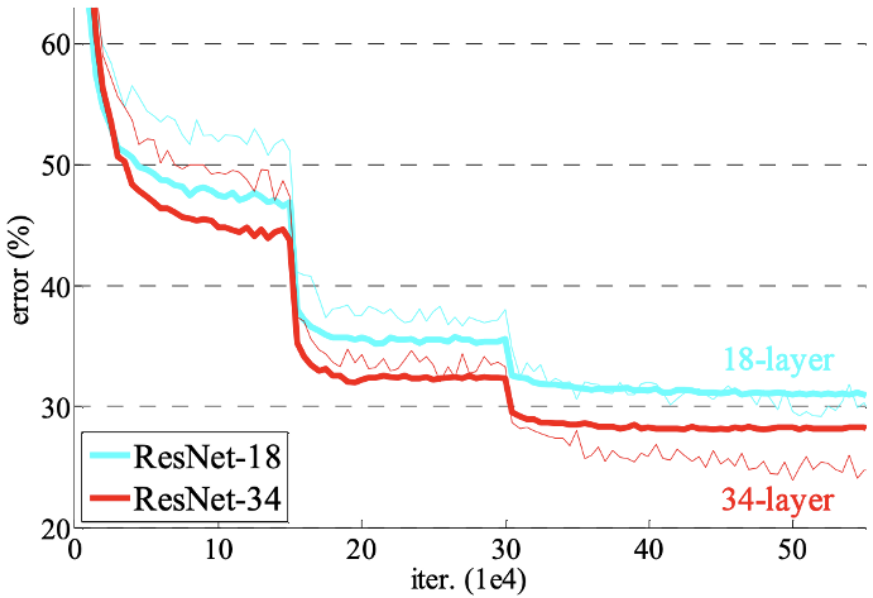


- If $z^{[l+1]} = 0$ (this happens when $w^{[l+1]} = 0, b^{[l+1]} = 0$), then $A^{[l+1]} = ReLU(A^{[l-1]}) = A^{[l-1]}$.
- Adding the two layers $[l]$ and $[l+1]$ can do as well as the simpler NN without.
- Allow training of very deep networks.

ResNets on Image Classification (Inspired by VGG-19)



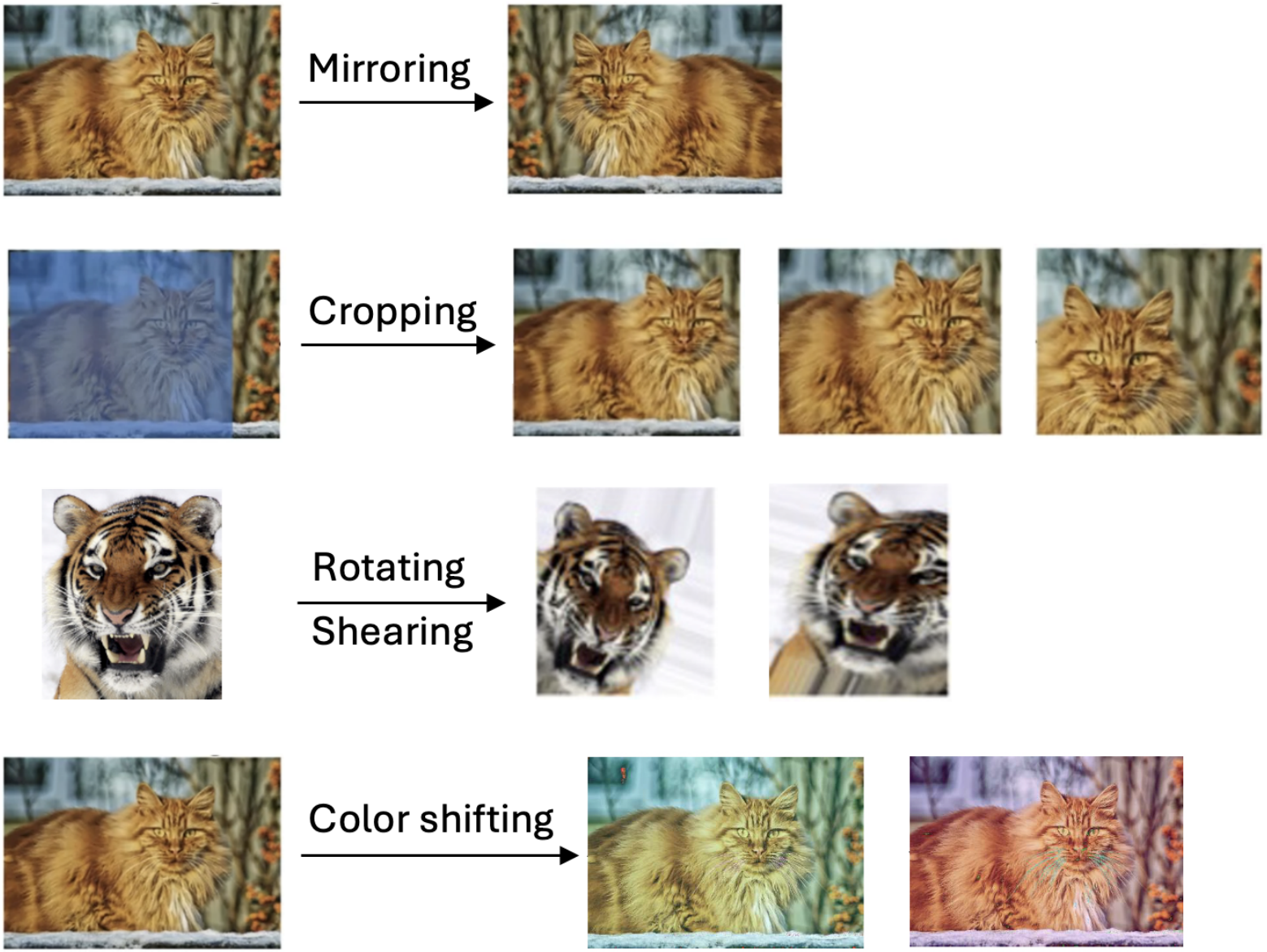




Training on **ImageNet**. Thin curves denote training error, and bold curves denote test error. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers.

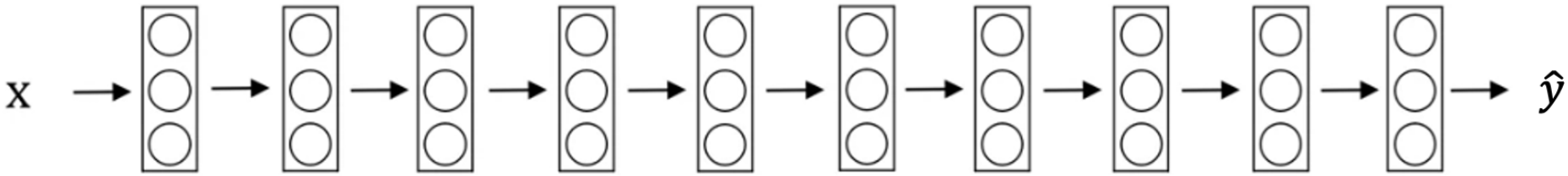
Data Augmentation (to increase sample size)

Distort image to produce different images with the same class label.



Transfer Learning

What to do if you have a small training set / similar task?

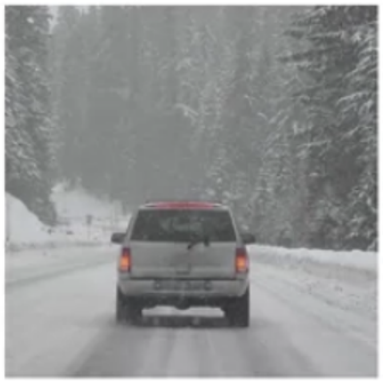


- Use architectures of networks in the literature.
- Use pretrained hidden layers (to reduce # parameters): use convolution filters trained from massive corpora such as imagenet, and just train the last few layers of the network for the new tasks.

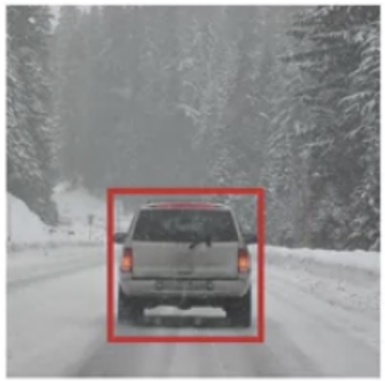
Reminder: Quiz for lecture 7 is due at 9pm on Monday Oct. 21st.

Object Detection

Image classification

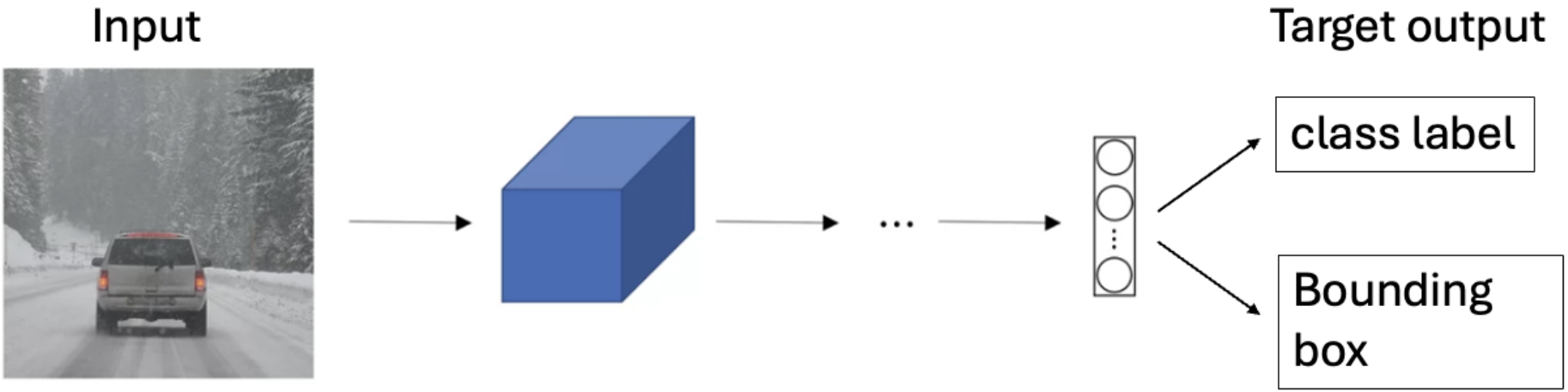


Classification with localization



Detection

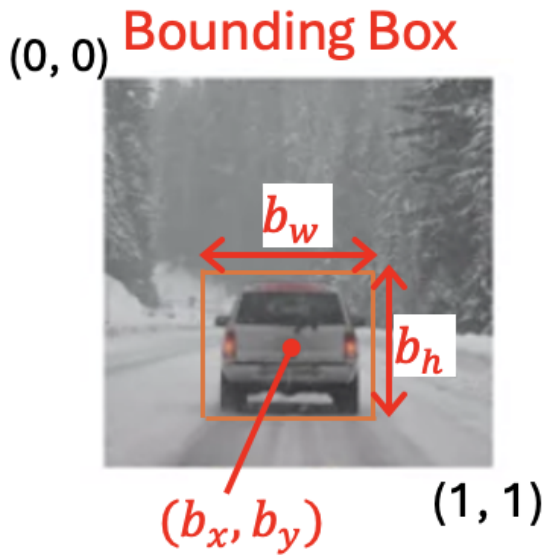
Classification with Localization



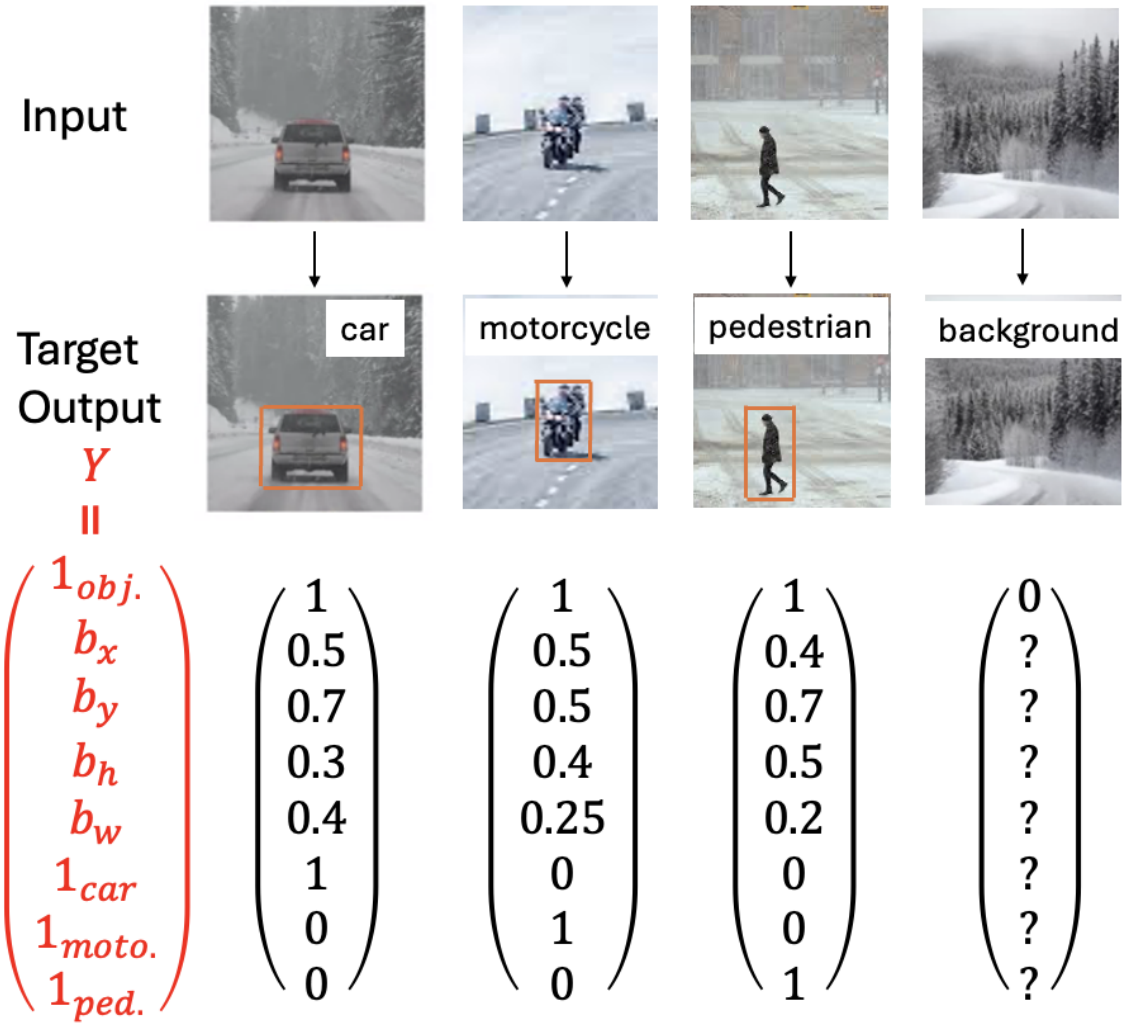
Classification:

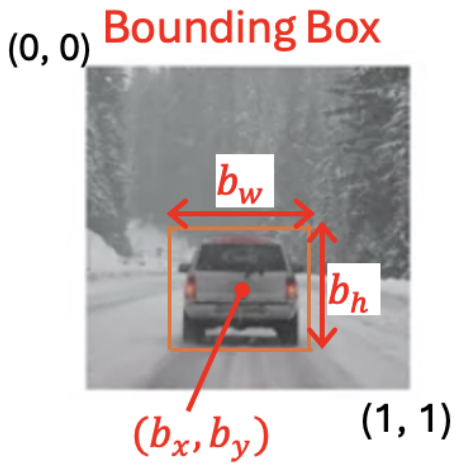
1. Car
2. Motorcycle
3. pedestrian
4. Background (other)

Localization:



Classification with Localization





Loss function:

$$l(Y, \hat{Y}) = \begin{cases} \sum_{m=1}^{8} (\hat{Y}_m - Y_m)^2 & if\ Y_1 = 1 \\ (\hat{Y}_1 - Y_1)^2 & if\ Y_1 = 0 \end{cases}$$

Landmark Detection

Identify important points (landmarks) in an image.
Each landmark is a coordinate in a 2-d image

Input

Output 1

$$Y = \begin{pmatrix} 1_{face} \\ b_{1x}, b_{1y} \\ b_{2x}, b_{2y} \\ b_{3x}, b_{3y} \\ b_{4x}, b_{4y} \end{pmatrix}$$

Output 2

$$Y = \begin{pmatrix} 1_{face} \\ b_{1x}, b_{1y} \\ b_{2x}, b_{2y} \\ \vdots \\ b_{21x}, b_{21y} \\ b_{22x}, b_{22y} \end{pmatrix}$$

Object Detection

Target:

Input

Output

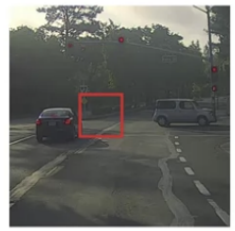
Training set:

X (closely cropped image)	Y
	1
	1
	1
	0
	0

Obtain a trained ConvNet

Sliding windows detection:

Apply the trained ConvNet to each little red box, sliding over the entire image

...  ... 

- Try different window size/stride
- Computationally expensive
- Solution: implementing sliding windows convolutionally.

Face verification vs. face recognition

Verification

- Input image, name/ID
- Output whether the input image is that of the claimed person

Recognition

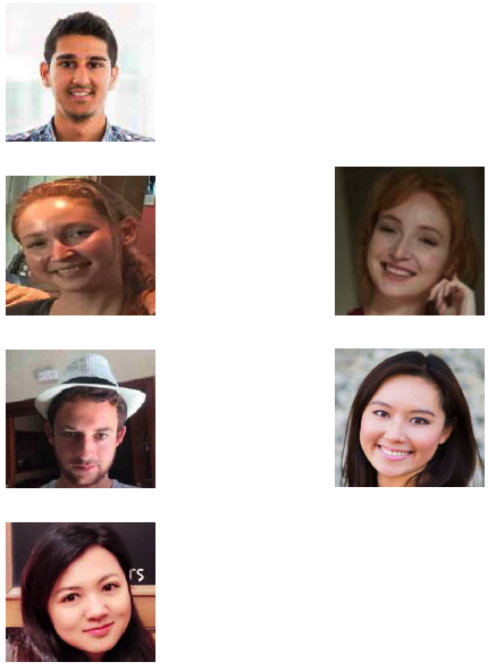
- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or "not recognized")

Andrew Ng

Task:

- Build a highly accurate face verification system.
- Use it in a recognition system.

One-shot learning



Learning from one example to recognize the person again

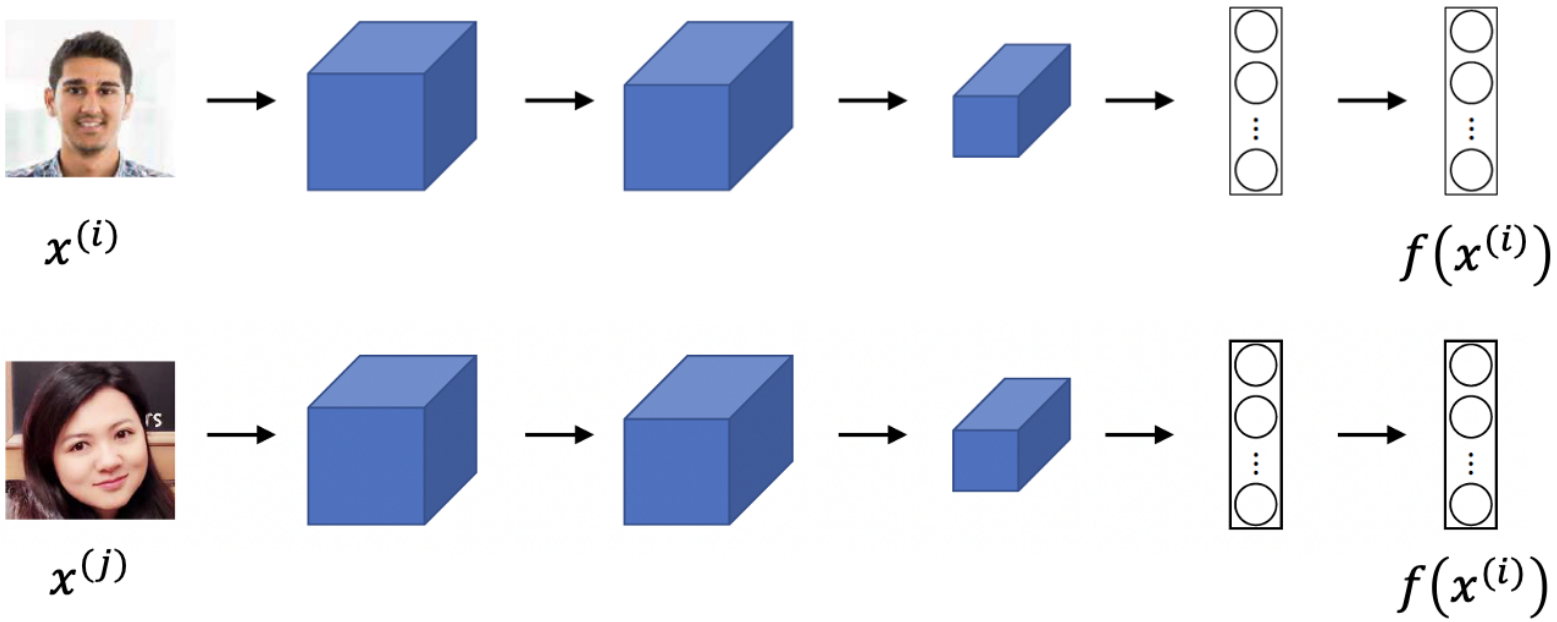
Andrew Ng

Learning a Similarity Function:

- d(img1, img2): dis-similarity measure between images
- Output: "same" if d(img1, img2)$\leq \tau$, and "different" otherwise.

Siamese Network

A NN that uses the same weights (parameters) while working on two different inputs to compute comparable output vectors.



Output of NN, $f(x^{(i)})$, define an encoding of image $x^{(i)}$. Learn parameters so that

- $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is small if $x^{(i)}$ and $x^{(j)}$ are the same person.
- $\|f(x^{(i)}) - f(x^{(j)})\|^2$ is large if $x^{(i)}$ and $x^{(j)}$ are different persons.

Train Network with Triplet Loss

One training data point contain three images with label A, P, N.

Anchor Positive Negative



Want: $\|f(A) - f(P)\|^2 + \alpha < \|f(A) - f(N)\|^2$.

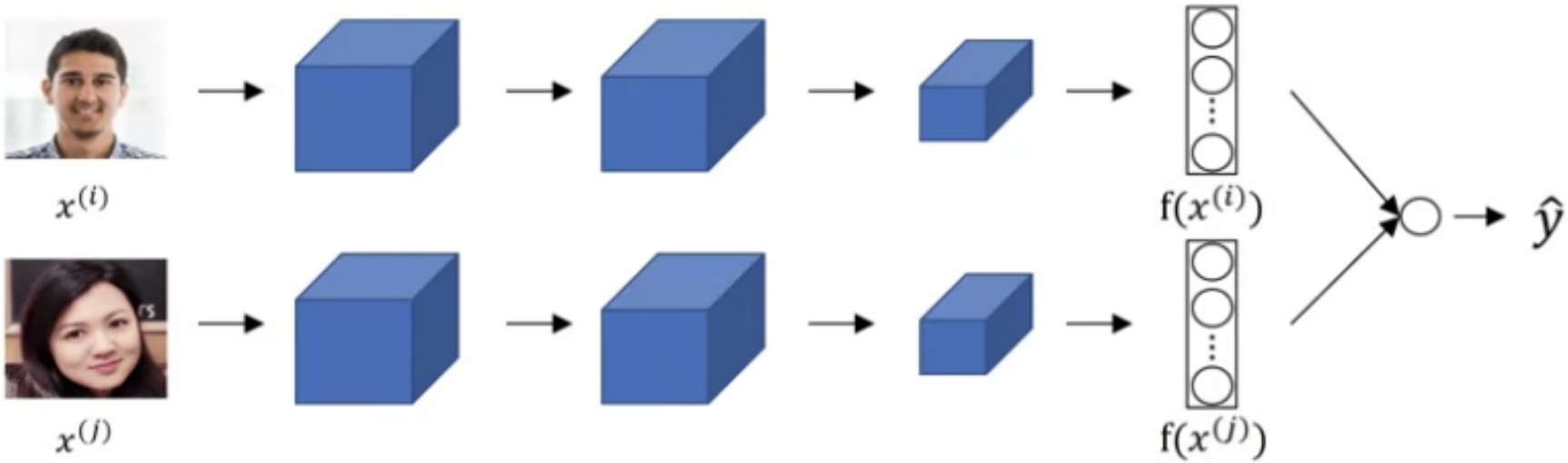
Loss: $l(A, P, N) = \max\left\{\|f(A) - f(P)\|^2 - \|f(A) - f(N)\}^2 + \alpha, 0\right\}$.

Construct training Set:

- A pool of images (e.g., 10k pictures of 1k persons)
- For each person, select a triplet (A, P, N) that are "hard" to train on (i.e., A and P look very different, and A and N look similar).

Others Methods instead of Triplet Loss

One training data point contain a pair of images with label "same" ($Y = 1$) or "different" ($Y = 0$).



- Binomial deviance loss $l(Y, \widehat{Y})$ with $\widehat{Y} = \text{sigmoid}\Big(\sum_{k=1}^{128} w_k \Big| f(x^{(i)})_k - f(x^{(j)})_k \Big| + b \Big)$.
- Hinge loss with $\widehat{Y} = \sum_{k=1}^{128} w_k \frac{[f(x^{(i)})_k - f(x^{(j)})_k]^2}{f(x^{(i)})_k + f(x^{(j)})_k}$.