

COS30082

Applied Machine Learning



Lecture 8

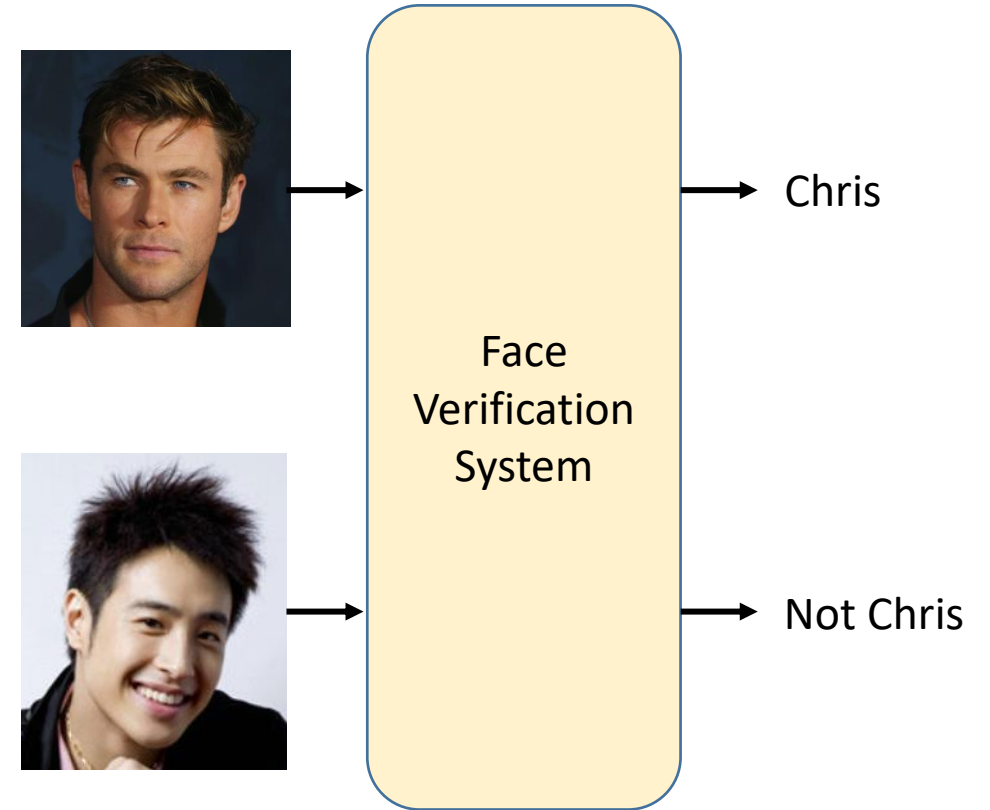
Face Recognition

What is Face Recognition?

- A set of two tasks:
 - **Face identification:** Given a face image that belongs to a person in a database, tell whose image it is
 - **Face verification:** Given a face image that might not belong to the database, verify whether it is from the person it is claimed to be in the database.

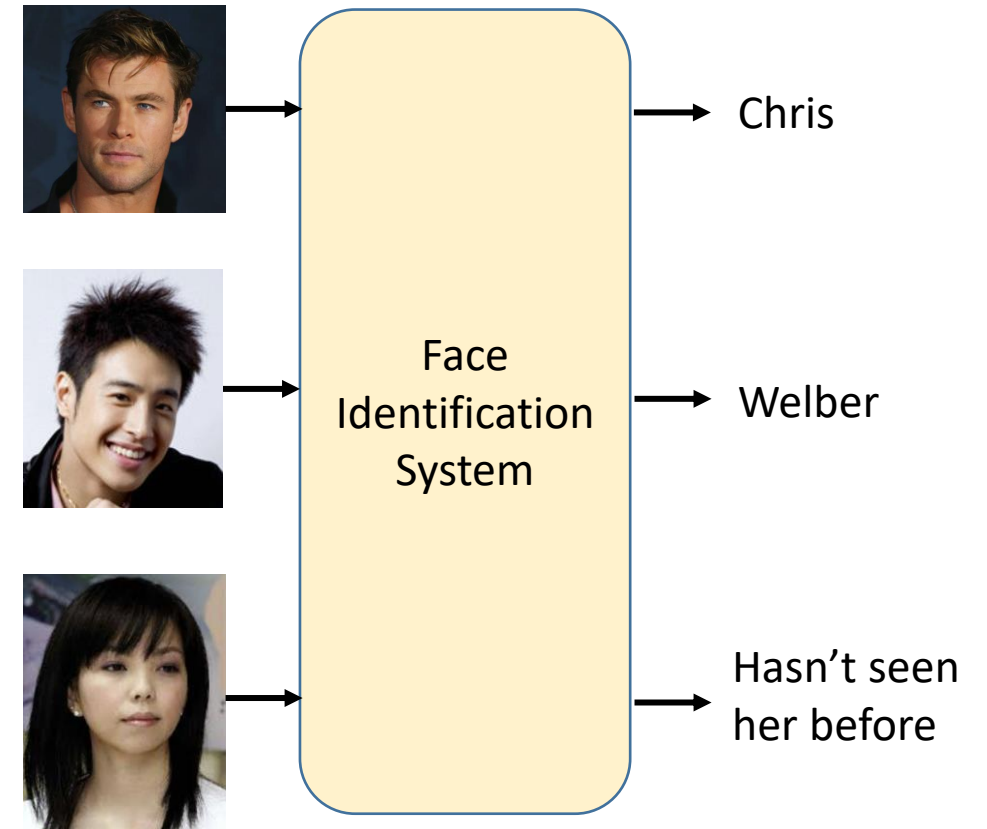
Face Verification

- Verification
 - Input: Image of a person, name/ID
 - Output: Output whether the input image is that of the claimed person.



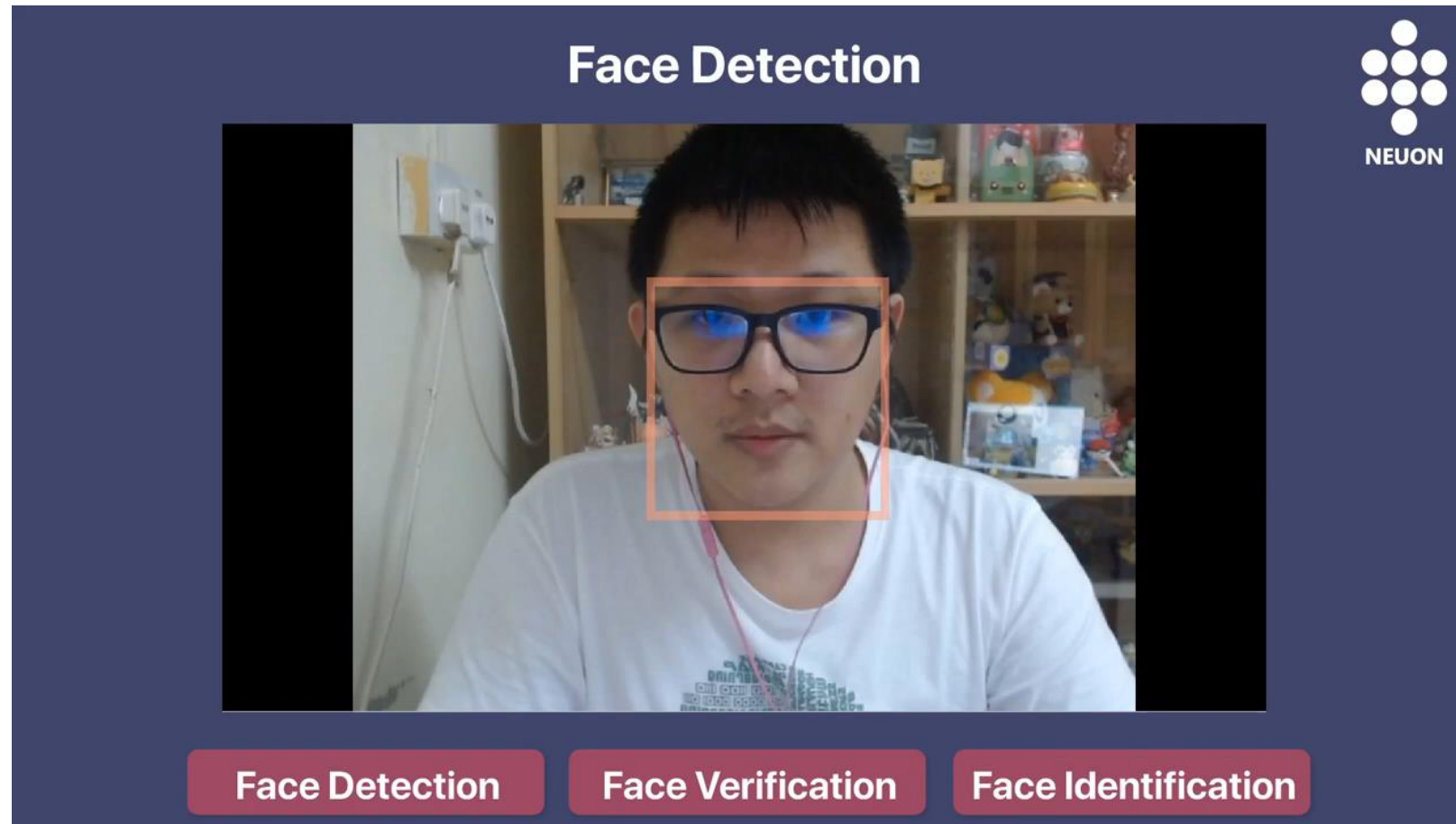
Face Identification

- Identification
 - Has a database of K persons
 - Get an input image
 - Output: Output ID if the image is any of the K persons or “not recognized”



Face recognition overview

[Courtesy of [NeuonAI](#)]

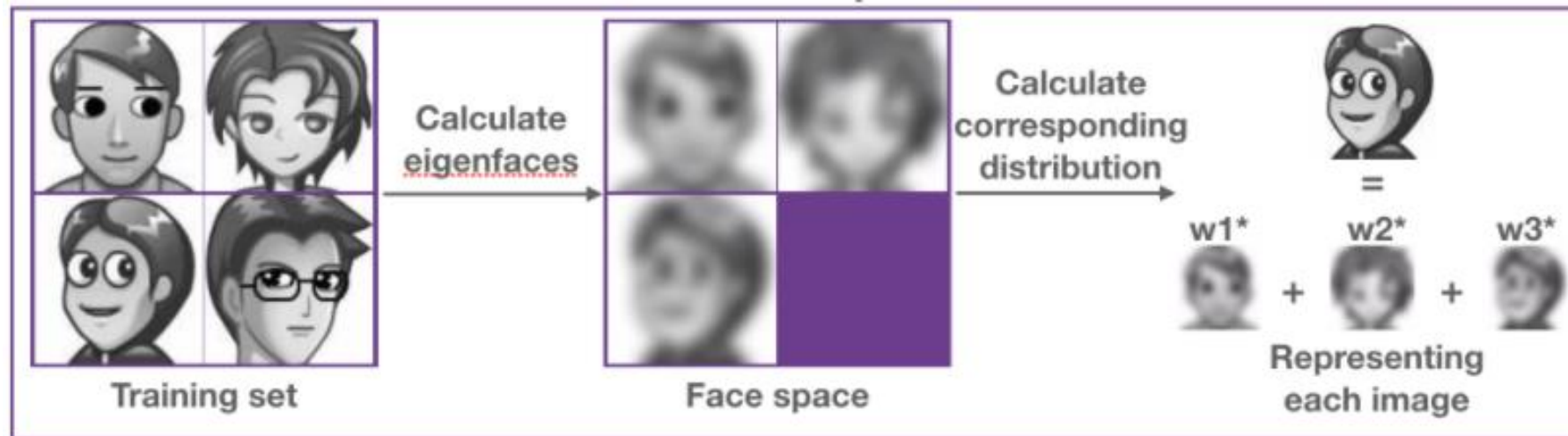


Importance of Face Detection

- The forefront module for any automatic face recognition system.
- The first process in many Human Computer Interactive system
 - Expression recognition
 - Cognitive state/Emotional state recognition
 - Biometric security solution
- First step in many surveillance system
 - Tracking
 - Automatic Target recognition or generic object detection algorithm
 - Human behaviour analysis

Classical face recognition algorithms

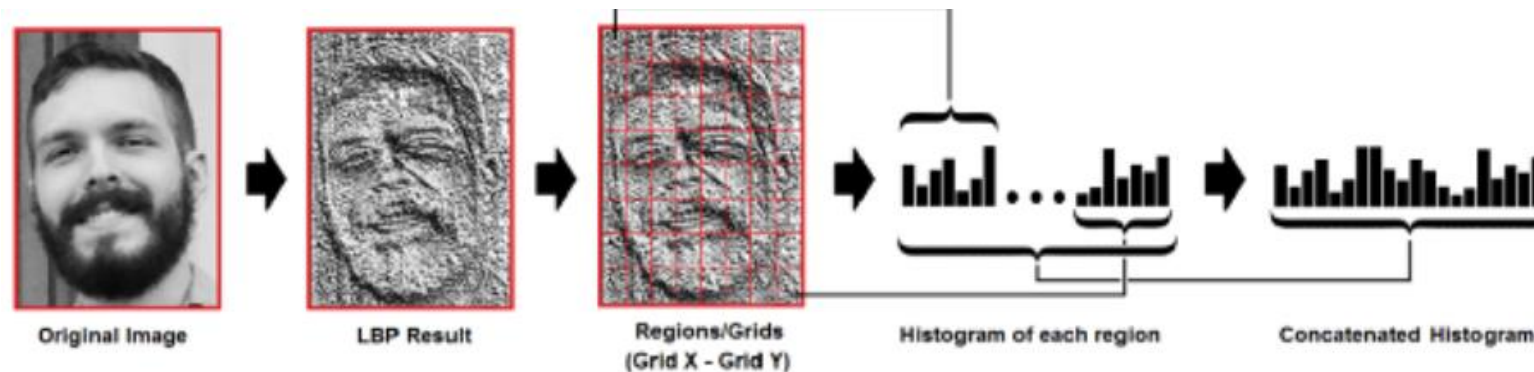
- Two categories of feature representation:
 - **Holistic features:** eigenfaces (PCA)
 - Advantage: Easy to implement, no knowledge (such as facial features) required except face ID.
 - Disadvantage: Algorithm sensitive to lighting, shadows and scale of the face in the image.



eigenfaces

Classical face recognition algorithms

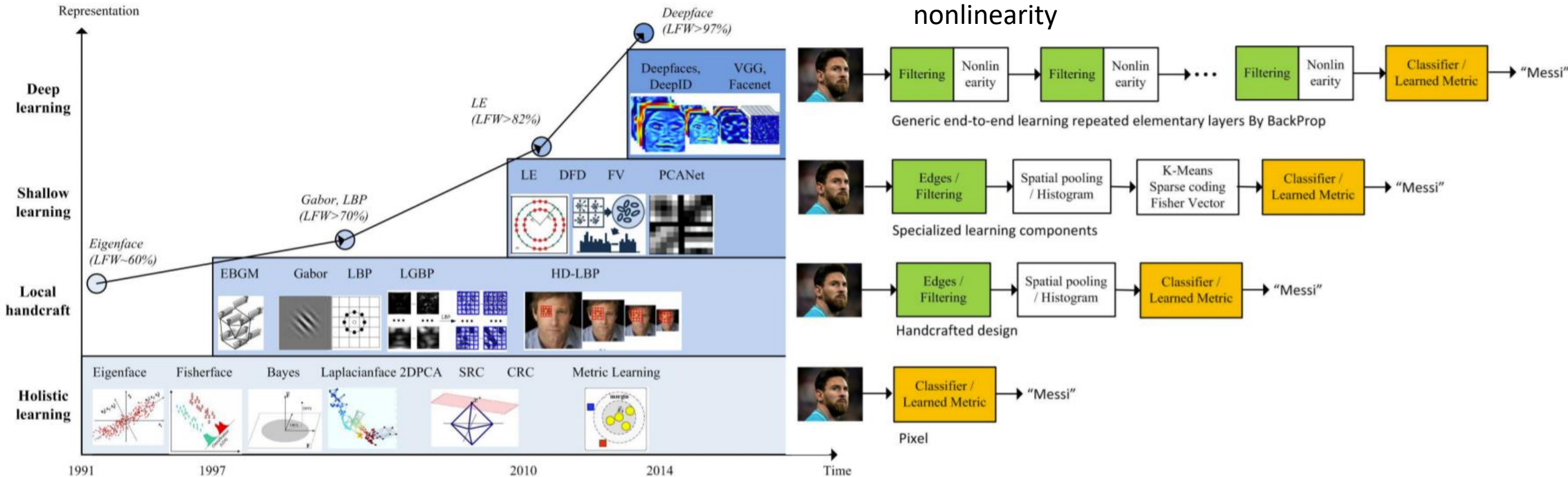
- Two categories of feature representation:
 - **Holistic features:** eigenfaces (PCA)
 - Advantage: Easy to implement, no knowledge (such as facial features) required except face ID.
 - Disadvantage: Algorithm sensitive to lighting, shadows and scale of the face in the image.
 - **Local based features:** Local binary pattern (LBP), SIFT
 - Advantage: Robust features in frontal face. Sensitive to noise, and invariant to translations and rotations.



LBP in face recognition

- Classical approaches would not perform well in very large datasets of faces and limit the learning of rich and robust representations of faces.
- The recent advent of affordable, **powerful GPUs** and the creation of **huge face databases** has drawn research focus primarily on the development of increasingly deep neural networks designed for all aspects of face recognition tasks, ranging from detection and preprocessing to feature representation and classification in verification and identification solutions.

Milestones of Face Representation



Masi, I., Wu, Y., Hassner, T. and Natarajan, P., 2018, October. Deep face recognition: A survey. In *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)* (pp. 471-478). IEEE.

One shot learning

- One-shot learning is a classification task that is used to form a model with **only one example** from each class, which in turn will be used to make predictions about many unknown examples in the future.

One shot learning and Face recognition

- Face recognition tasks provide examples of one-shot learning.
- Specifically, in the case of face identification, a model or system may only have one or a few examples of a given person's face and must correctly identify the person from new photographs with changes to expression, hairstyle, lighting, accessories, and more.
- In the case of face verification, a model or system may only have one example of a person's face on record and must correctly verify new images/photos of that person, perhaps each day.
- As such, face recognition is a common example of one-shot learning

Can CNN's usual learning process yield good results?

- Train a CNN model using one example from each class.

Known data

Face ID: 1



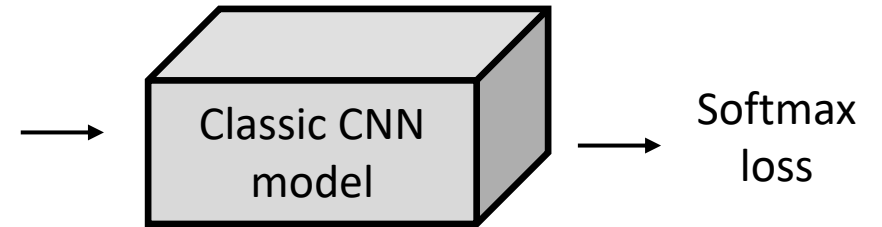
Face ID: 2



Face ID: 3



Face ID: 4



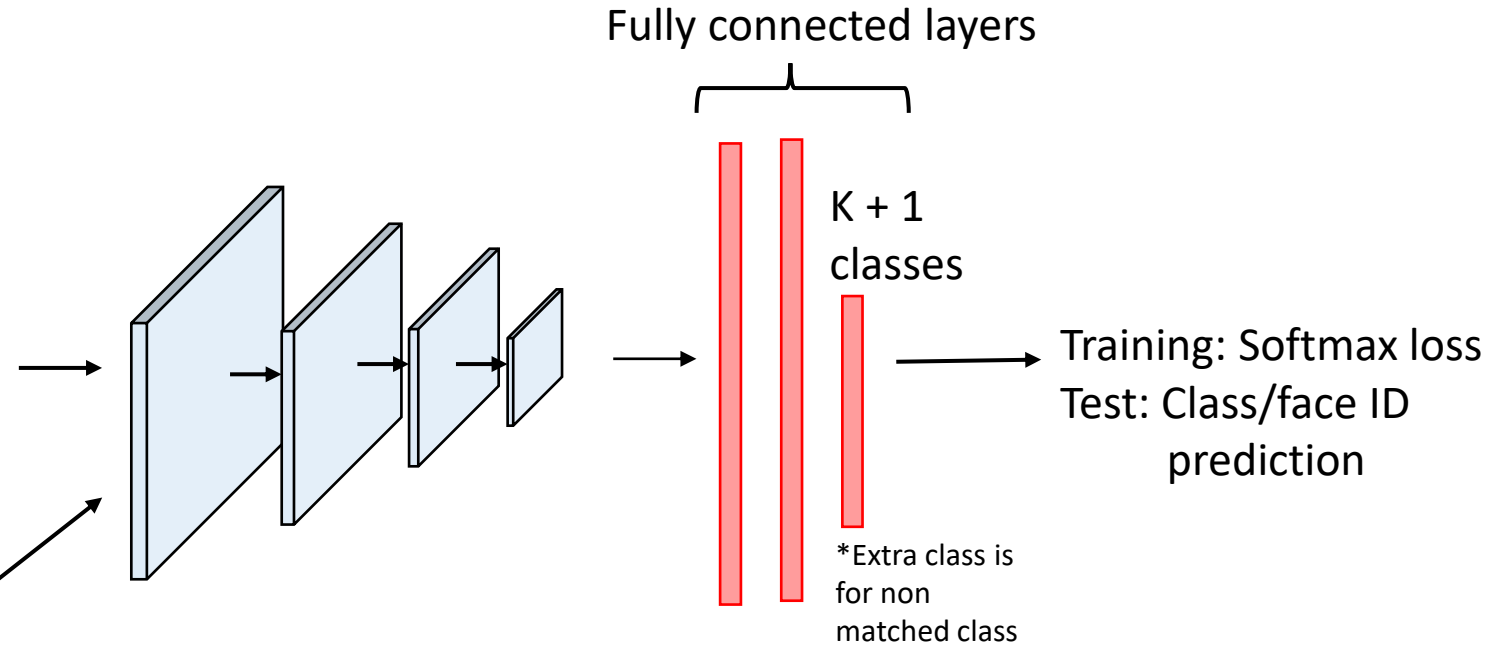
Can CNN's usual learning process yield good results?

- Answer is no. Why?

Training with known data

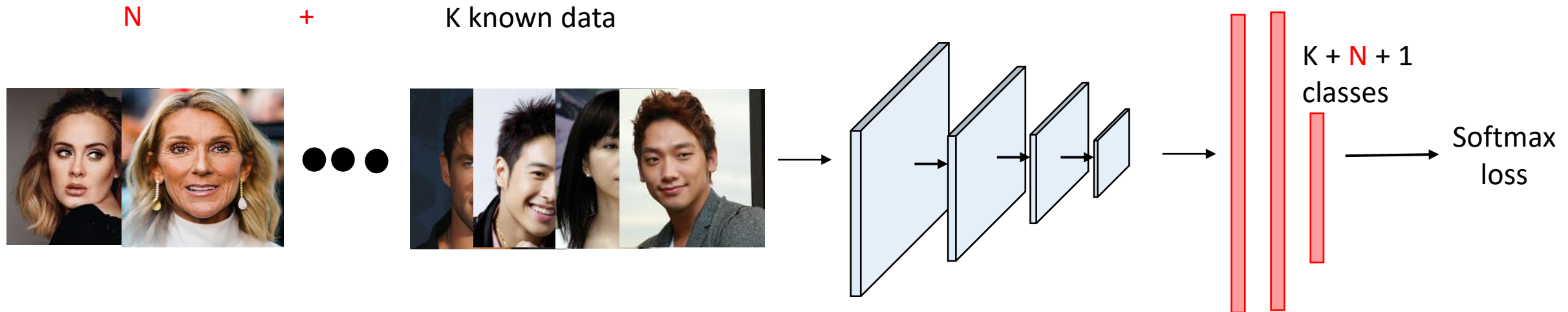


Testing with unknown data



Can CNN's usual learning process yield good results?

- Answer is no. Why?
 - Training data is too small to form a robust CNN model.
 - Not practical due to intractable changes in the output classes / face IDs.
 - For example, the increase of N classes due to newly registered actors leads to classification layer to be redesigned and re-trained.



Repurposing CNNs for One shot learning

- Instead of treating the task as a classification problem, one-shot learning turns it into a **difference-evaluation problem**.
- Learning a “similarity” function.

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

$$\begin{array}{l} \text{If } d(img1, img2) \leq \tau \rightarrow \text{same} \\ \phantom{\text{If }} > \tau \rightarrow \text{different} \end{array} \quad \left. \vphantom{\begin{array}{l} \text{If } d(img1, img2) \leq \tau \rightarrow \text{same} \\ \phantom{\text{If }} > \tau \rightarrow \text{different} \end{array}} \right\} \begin{array}{l} \text{face} \\ \text{verification} \end{array}$$

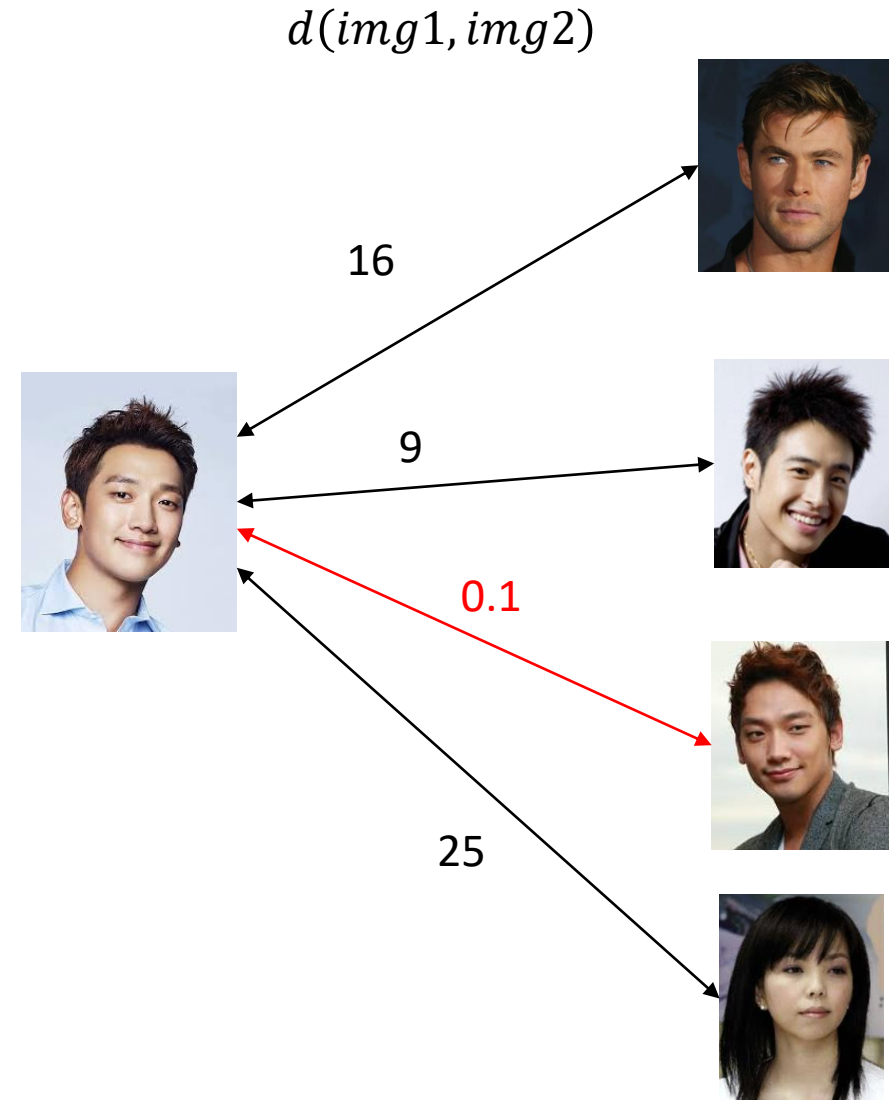


One shot
learning model

Not similar









Repurposing CNNs for One shot learning

- For face recognition, similarity function is used to check for every pairs of the images.
- If the images contain the same face, the neural network returns a value that is smaller than a specific threshold (say, 0.2) and if they're not the same face, it will be higher than the threshold.
- So, how we train a deep neural network to learn this function d ?



Siamese network

- The key to one-shot learning is an architecture called the “Siamese neural network.”
- Siamese networks were used more recently, where deep convolutional neural networks were used in parallel image inputs in a 2015 paper by Gregory Koch, et al. titled “Siamese Neural Networks for One-Shot Image Recognition.”

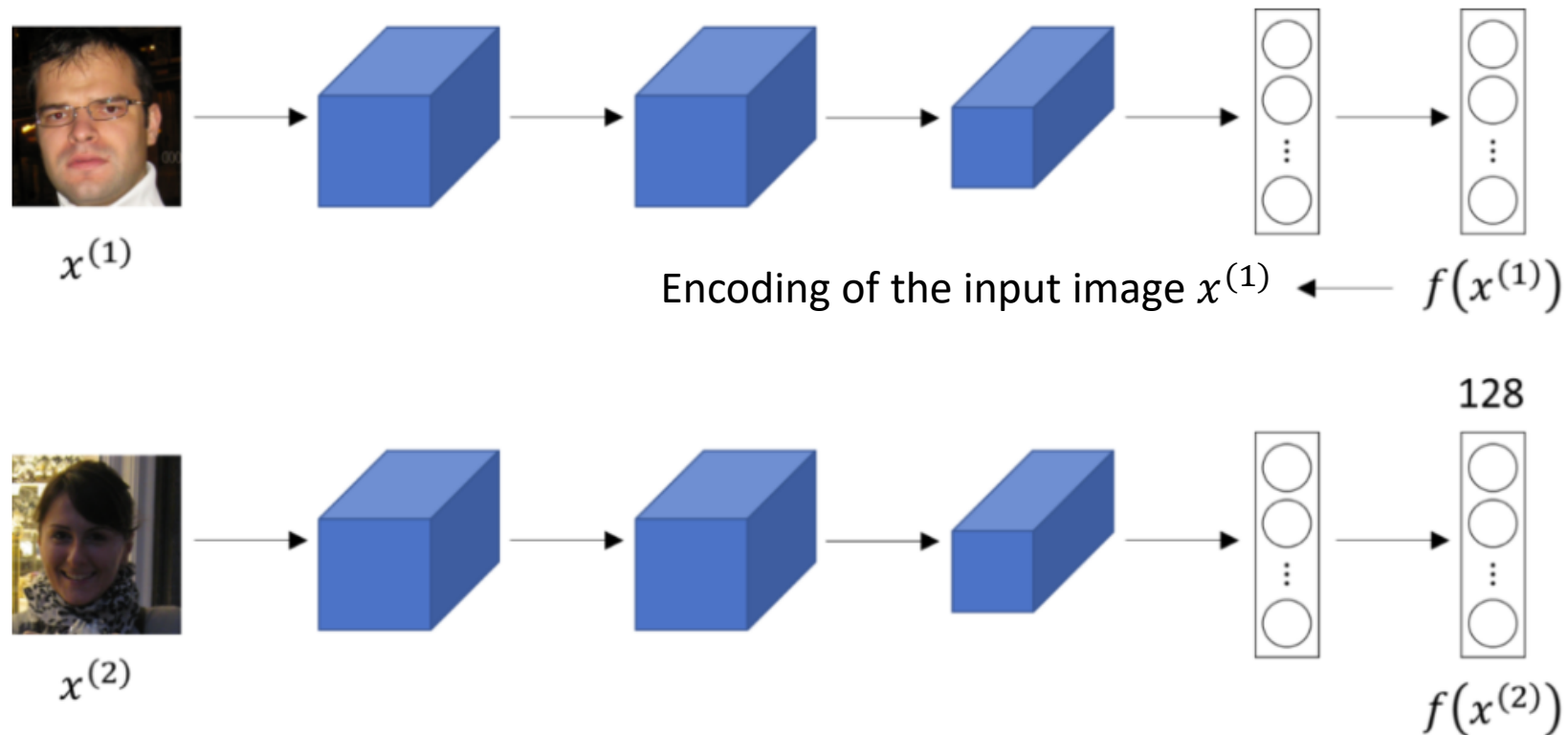
		same	"cow" (speaker #1)	"cow" (speaker #2)	same
		different	"cow" (speaker #1)	"cat" (speaker #2)	different
		same	"can" (speaker #1)	"can" (speaker #2)	same
		different	"can" (speaker #1)	"cab" (speaker #2)	different

Verification tasks (training)

Example of Image Verification Used to Train a Siamese Network.
Taken from: Siamese Neural Networks for One-Shot Image Recognition.

Siamese network

- Siamese neural network is not much different from other convolutional neural nets. It takes images as input and encodes their features into a set of numbers. The difference comes in the output processing.



Goal of learning

- During the training phase, classic CNNs tune their parameters so that they can associate each image to its proper class. The Siamese neural network, on the other hand, trains to be able to measure the distance $\|f(x^{(1)}) - f(x^{(2)})\|_2^2$ between the features in two input images.
- Parameters of neural network define an encoding $f(x^{(1)})$. The goal is to learn parameters so that:

If $x^{(i)}, x^{(j)}$ are the same person, $\|f(x^{(i)}) - f(x^{(j)})\|_2^2$ is small.

If $x^{(i)}, x^{(j)}$ are the different person, $\|f(x^{(i)}) - f(x^{(j)})\|_2^2$ is large.

So, how can we define an objective function to make sure this condition satisfied???

Triple loss

- To achieve this goal, we use a function called “triplet loss”.
- Basically, the triplet loss trains the neural network by giving it three images: an **anchor** image, a **positive** image, and a **negative** image.



Anchor

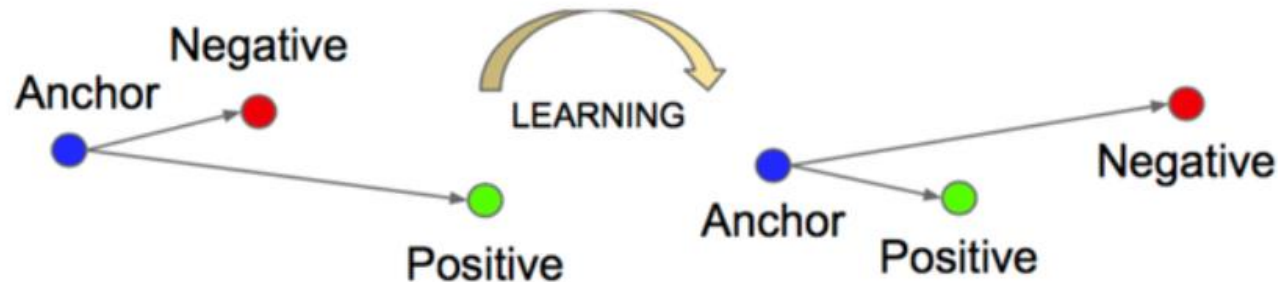


Positive



Negative

- After the training, the positive examples will be closer to the anchor while the negative examples will be farther from it.



Learning Objective



Anchor



Positive



Anchor



Negative

Goal: $d(A, P) = ||f(A) - f(P)||^2 \leq d(A, N) = ||f(A) - f(N)||^2$

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 \leq 0$$

- To prevent deep model output 0 for all the encodings or learn the similar encoding for every images, we add a margin.

$$||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \text{margin} \leq 0$$

- Margin defines how far away the dissimilarities should be in order to distinguish the two images better.

Loss function formulation

- Given 3 images A , P , and N :

$$L(A, P, N) = \max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha, 0)$$

- Notice that, to train the model, we need some pairs of A and P for training, meaning pairs of pictures of the same person.
- After having the trained model, you can apply in the one-shot learning problem in the face recognition, where maybe you have only single photo of the person you want to recognize.

Choosing the Triplets A,P,N

- During training, if A , P , and N are chosen randomly,
 $d(A, P) + \alpha \leq d(A, N)$ is easily satisfied
- We have to choose triplets that are “hard” to train in order to increase the learning efficiency.
 - Choose the samples where the $d(A, P)$ close to $d(A, N)$. [Refer to FaceNet]

- Based on the definition of the loss, there are three categories of triplets:

- Easy triplets: triplets which have a loss of 0, because:

$$d(A, P) + \textit{margin} < d(A, N)$$

- Hard triplets: triplets where the negative is closer to the anchor than the positive:

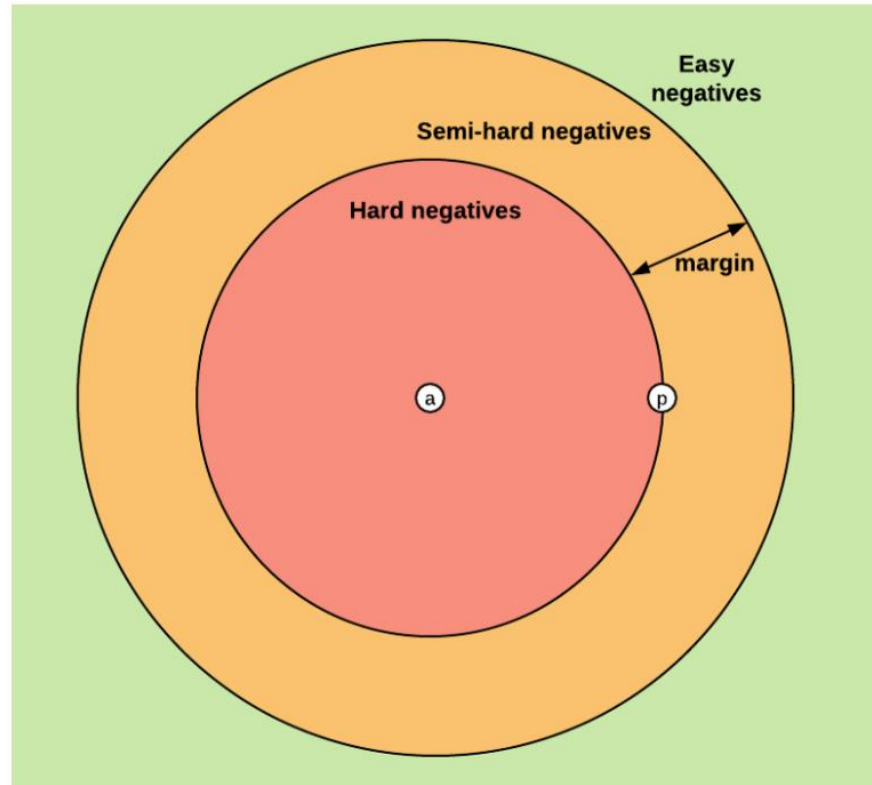
$$d(A, N) < d(A, P)$$

- Semi-hard triplets: triplets where the negative is not closer to the anchor than the positive, but which still have positive loss:

$$d(A, P) < d(A, N) < d(A, P) + \textit{margin}$$

Triplet Mining

- Each of these definitions depend on where the negative is, relatively to the anchor and positive. We can therefore extend these three categories to categories of negatives: **hard negatives**, **semi-hard negatives** or **easy negatives**.

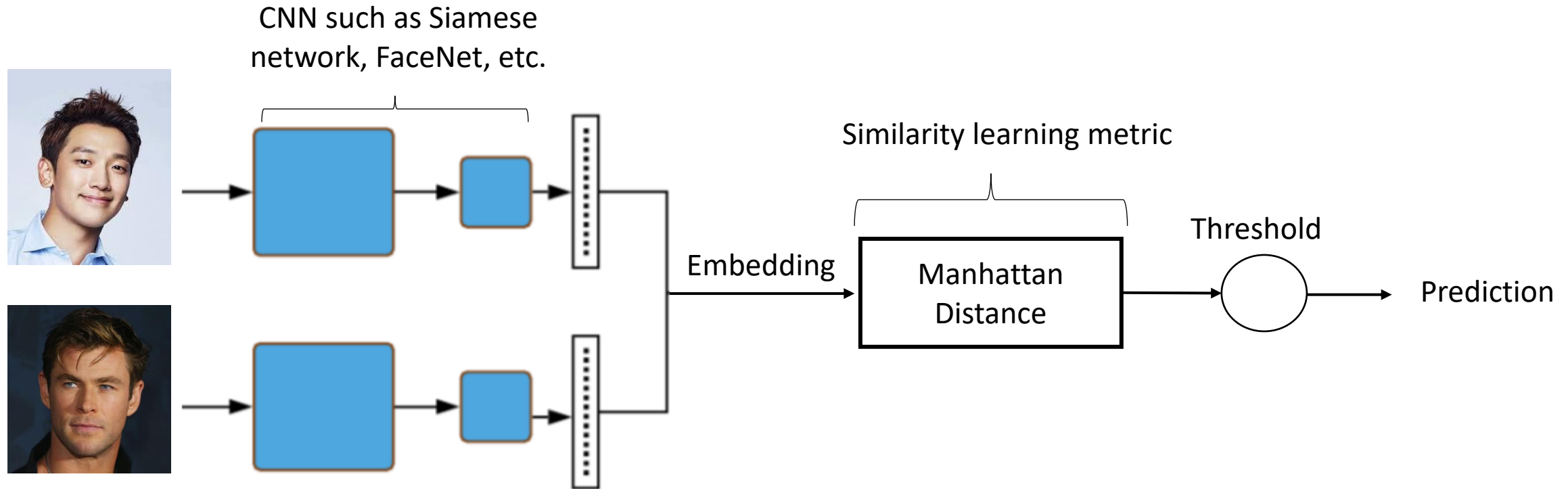


Regions of embedding space for negatives.

Triplets Mining for training

- An important decision of a training with Triplet Loss is negatives selection or triplet mining. The strategy chosen will have a high impact on the training efficiency and final performance. An obvious appreciation is that training with Easy Triplets should be avoided, since their resulting loss will be 0.
- A model can be trained on triplets by using either offline or online triplet mining.
 - **Offline Triplet Mining:** In this approach, we first generate the triplets manually and then fit the data to the network.
 - **Online Triplet Mining:** In this approach, we feed a batch of training data, generate triplets using all examples in the batch and calculate the loss on it. This approach allows us to randomize the triplets and increase the chance to find triplets with high losses — this will help train the model faster. For batch size of N , we can generate at most N^3 triplets.

Deploy ConvNet as an embedding generator



- The embedding can be used to compare faces in three ways:
 - **Face verification** considers two faces and it decides whether they are similar or not. Face verification can be done by computing the distance metric.
 - **Face recognition** is a classification problem for labelling a face with a name. The embedding vector can be used for training the final labels.
 - **Face clustering** groups similar faces together just like how photo applications cluster photos of the same person together. A clustering algorithm such as K-means is used to group faces.

Types of similarity learning metrics

- **Cardinality based similarity metrics** leverage on the union and intersection of sets in comparison. The Jaccard similarity metric uses cardinality to determine the relation between two vectors under experiment.

$$d(x^{(i)}, x^{(j)}) = \frac{|x^{(i)} \cap x^{(j)}|}{|x^{(i)} \cup x^{(j)}|}$$

- It is the size of the overlap between the two sets (what they have in common) divided by the size of the combined sets (everything they have without duplicates).
- A higher value means the sets are more similar.

Types of similarity learning metrics

- **Orientation based similarity metrics** leverage on the angle between two vectors in their respective vector spaces. An example is the Cosine similarity metric, it calculates similarity by measuring the cosine of angle between two vectors.

$$d(x^{(i)}, x^{(j)}) = \cos(\theta) = \frac{\sum_{k=1}^n |x_k^{(i)} \cdot x_k^{(j)}|}{\sqrt{\sum_{k=1}^n x_k^{(i)2}} \sqrt{\sum_{k=1}^n x_k^{(j)2}}}$$

- It measures similarity by calculating the cosine of the angle between two vectors in a multidimensional space. It's like you have two arrows pointing in different directions, and you're checking how closely aligned they are.
- If the Cosine similarity is close to 1, it means the vectors are pointing in roughly the same direction, indicating that the items you're comparing are similar.

Types of similarity learning metrics

- **Distance-based similarity metrics** are the Euclidean distance, Manhattan distance, Minkowski distance, and so on. The basic idea common to these metrics is that they leverage the average distance between the elements of two vectors under experiment.

- Euclidean distance

$$d(x^{(i)}, x^{(j)}) = \sqrt{(x_1^{(i)} - x_1^{(j)})^2 + (x_2^{(i)} - x_2^{(j)})^2 + \dots + (x_n^{(i)} - x_n^{(j)})^2}$$

- Manhattan distance

$$d(x^{(i)}, x^{(j)}) = |x_1^{(i)} - x_1^{(j)}| + |x_2^{(i)} - x_2^{(j)}| + \dots + |x_n^{(i)} - x_n^{(j)}|$$

- Minkowski distance

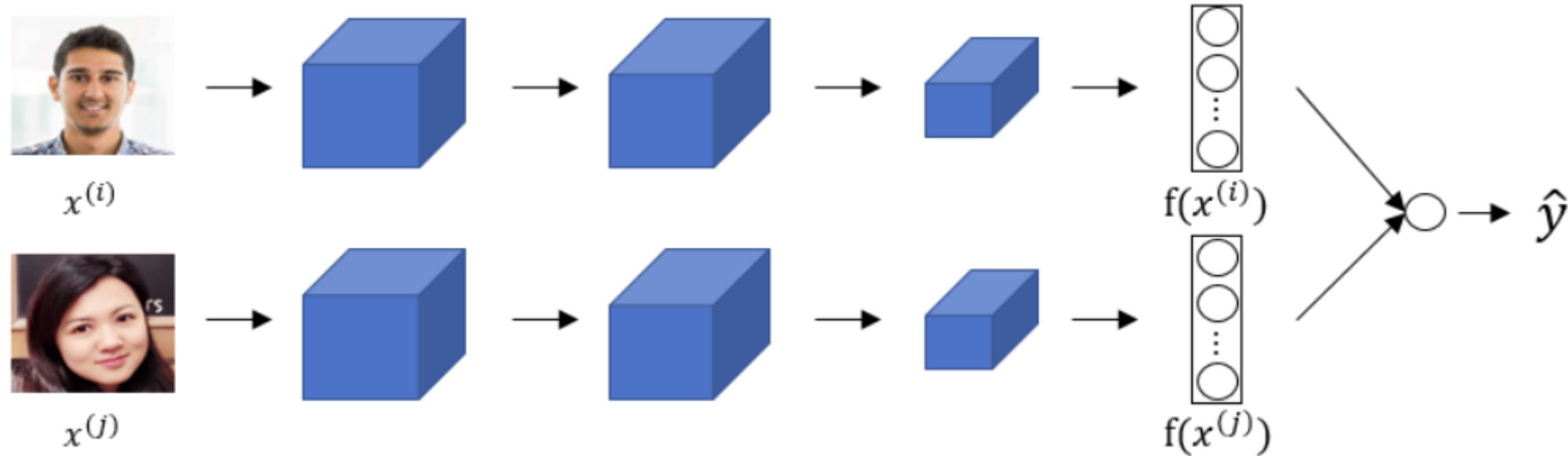
$$d(x^{(i)}, x^{(j)}) = \left(\sum_{k=1}^n |x_k^{(i)} - x_k^{(j)}|^p \right)^{\frac{1}{p}}$$

If you set $p = 2$, you get the Euclidean distance. If you set $p = 1$, you get the Manhattan distance. You can also set p to other infinite numbers.

Face verification and Binary Classification

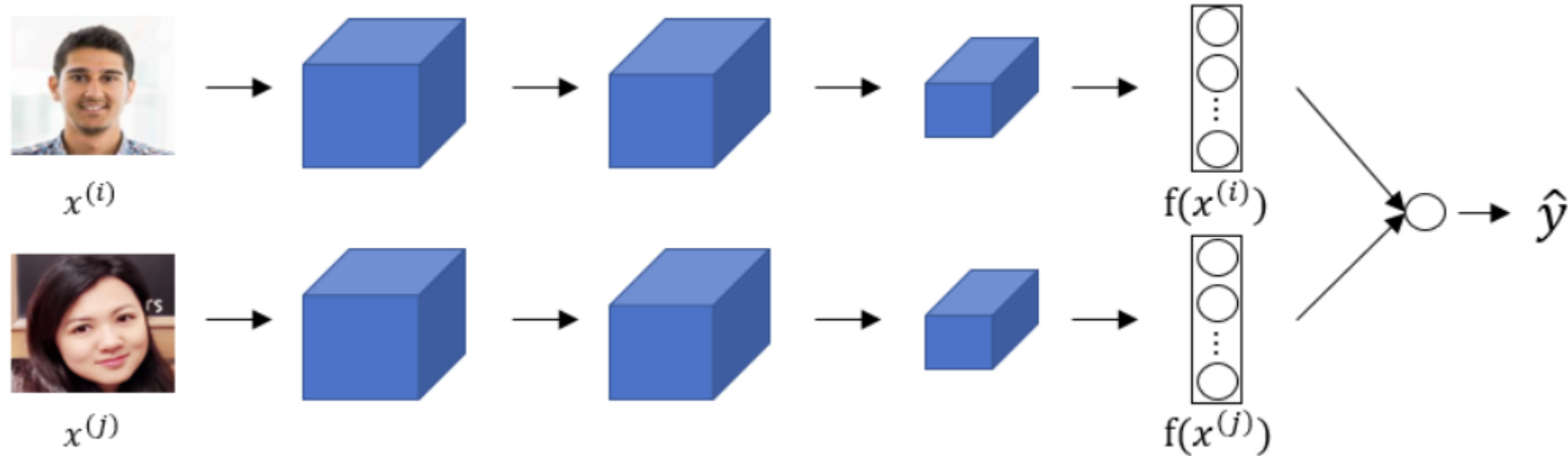
- Triplet loss is one way to learn the parameters of a neural network for face recognition. There's another way to learn these parameters as a straight **binary classification problem**.
- To do that, we have to select a pair of neural network, says the Siamese network. We simultaneously compute the embedding of image pairs, such as 128-dimensional embedding, or more, which are then fed into the logistic regression unit, and then move on to the next step to make predictions.

Face verification and Binary Classification









- The final layer is a sigmoid layer.
- $Y' = w^{(i)} * \text{Sigmoid} (f(x^{(i)}) - f(x^{(j)})) + b$
where the subtraction is the Manhattan distance between $f(x^{(i)}) - f(x^{(j)})$
- Some other similarities can be Euclidean and chi square similarity.
- The neural network here is Siamese means the top and bottom convolutions has the same parameters.

Face verification and Binary Classification



- A good performance/deployment trick:
 - Pre-compute all the images that you are using as a comparison to the vector $f(x^{(j)})$
 - When a new image that needs to be compared, get its vector $f(x^{(i)})$ then put it with all the pre computed vectors and pass it to the sigmoid function.

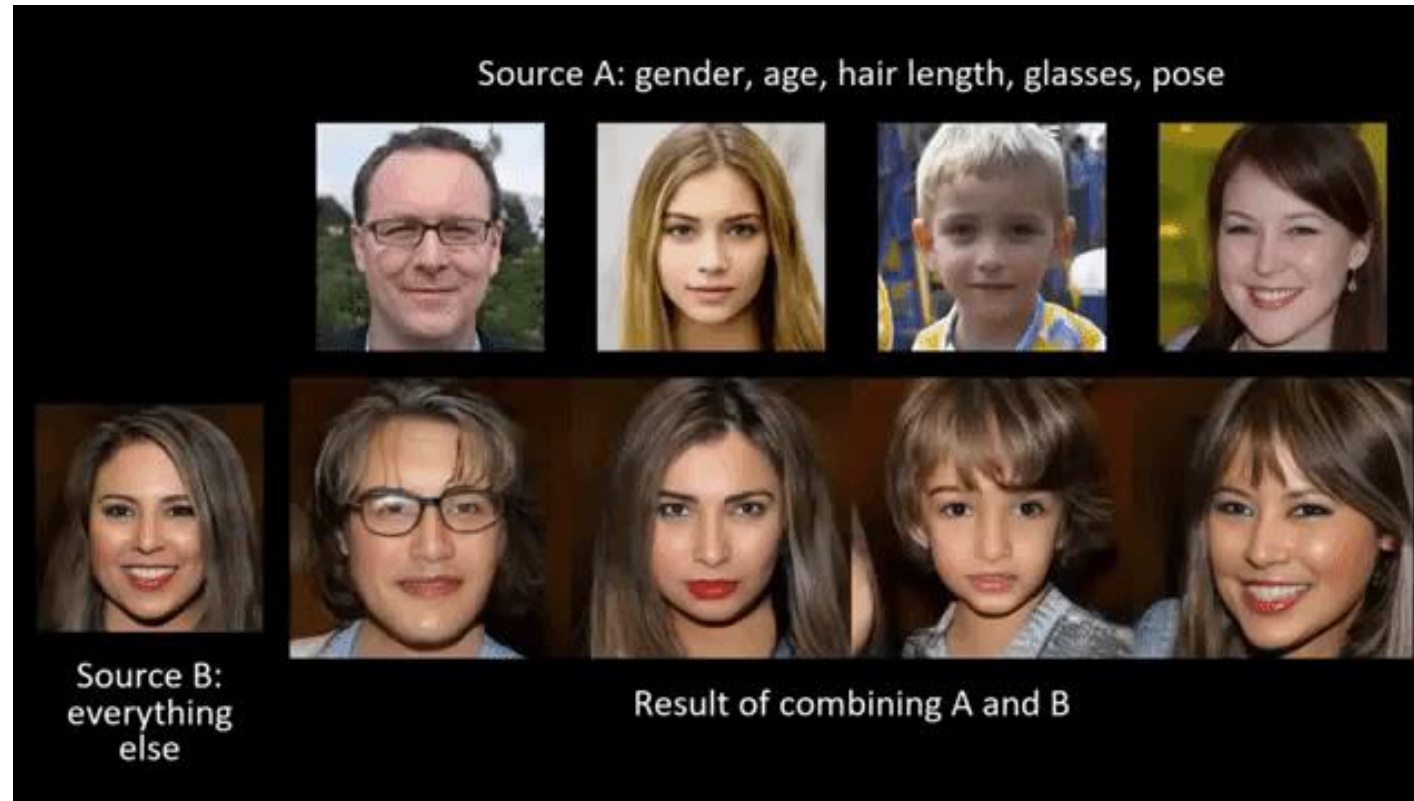
Face verification and Binary Classification

x		y	
		0	“Different”
		0	“Different”
		1	“Same”

- There are two different approaches for face recognition:
 - either you minimize the loss of triplets by training on the triplets A, P, N , or
 - you use a training set containing two images with only targets of 1 or 0 and trained by logistic regression.

Next lecture

❖ Generative Deep Learning



A Generative Adversarial Network AI That Creates Portraits of Human Faces That Don't Exist in Real Life [\[NVIDIA: StyleGan\]](#)