

# Аналіз біометрії. Вектори характеристик.

26 листопада 2024 р.

Захаров Дмитро

# План

## 1 Вступ

- Supervised Learning
- Dense Neural Networks
- Convolutional Neural Networks

## 2 Deep Pattern Recognition

- Embedding Neural Network
- Тріплет мережа

## 3 Додаткові розділи біометрії

- Безпека векторів фіч

Вступ

## Deep Pattern Recognition

0000000000000000

## Додаткові розділи біометрії

□-□-□-□-□

## Вступ

# Формулювання задачі

Зазвичай, на вхід подається набір даних виду:

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\},$$

де задача – побудувати функцію  $f$ , що “достатньо точно” відображає  $x_i$  на  $y_i$  (*Supervised Learning*).

## Приклад

Розпізнавання цифри з зображення.  $x_i$  (вхід) – зображення,  $y_i$  (виход) – цифра від 0 до 9.

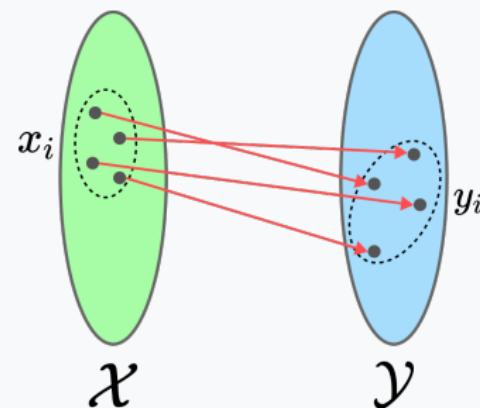
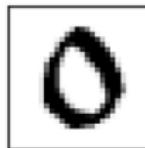


Рис.: Вхідний набір даних  $\mathcal{D}$  – послідовність пар виду  $x_i \mapsto y_i$ .

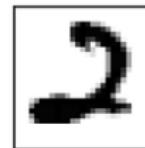
## Приклад: MNIST



Digit: 0



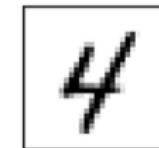
Digit: 1



Digit: 2



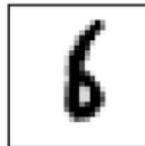
Digit: 3



Digit: 4



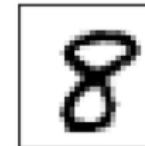
Digit: 5



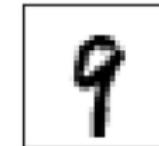
Digit: 6



Digit: 7



Digit: 8



Digit: 9

Рис.: Розпізнавання цифр. Набір даних MNIST – найбільш популярний вибір для написання прототипів (*Proof of Concept*).

# Ефективність відображення

## Питання

Як оцінити, що задана функція  $f$  є “гарною” або “поганою”?

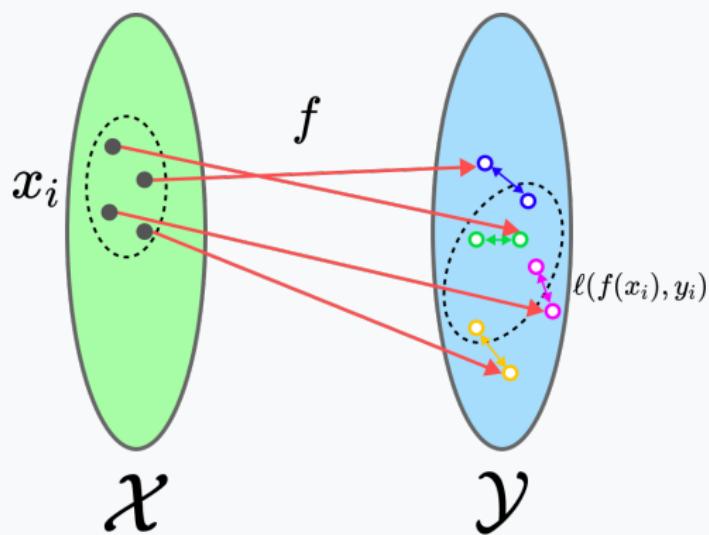


Рис.:  $f(x_i)$  “промахнулось” і дало відхилення від  $y_i$  – що далі?

# Функція втрати

- ✓ Введемо втрату  $\ell$  – міра відстані між передбаченням  $\hat{y} = f(x)$  та фактичним значенням  $y$ .

## Приклад функції втрати

Якщо вихідне значення – вектор, то можна покласти  
 $\ell(\mathbf{y}, \hat{\mathbf{y}}) :=$  відстань( $\mathbf{y}, \hat{\mathbf{y}}$ ).

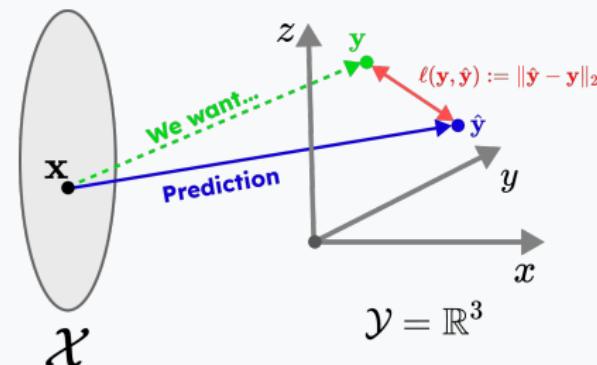


Рис.: Для  $\mathcal{Y} = \mathbb{R}^3$  беремо Евклідову відстань для  $\ell$ .

# Оптимізаційна задача

Нехай функція  $f(\star; \theta)$  параметризована набором параметрів  $\theta$ .

## Приклад параметризації

Наприклад, нехай ми шукаємо

$$f(x) = ax + b,$$

де  $\theta = (a, b)$

Задача – мінімізувати втрату  $\ell(f(x_i), y_i)$ .

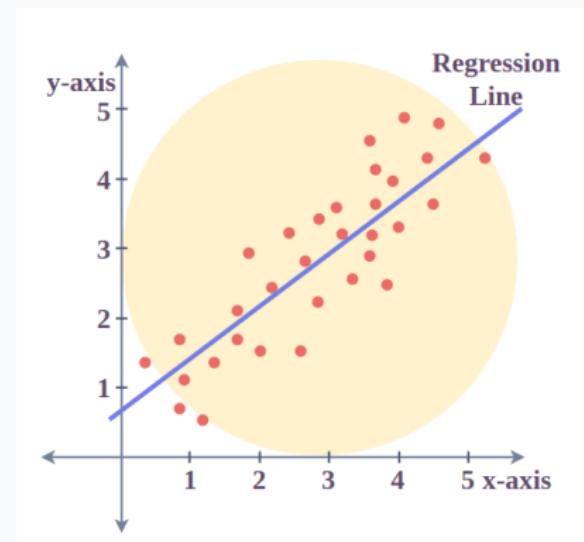


Рис.: Підбираємо пряму, що проходить максимально близько до інших точок.

# Повнозв'язні нейронні мережі (Dense Neural Networks)

Будуємо функцію, що на вхід приймає вектор довжини  $n_{in}$ , на вихід видає вектор довжини  $n_{out}$ , а також параметризована матрицями ваг та зсувами (bias).

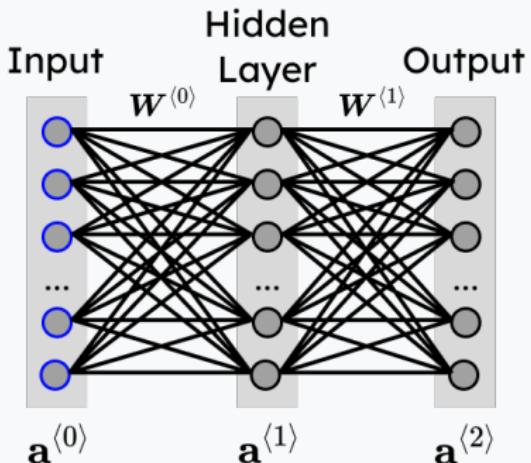


Рис.: Приклад повнозв'язаної нейронної мережі з трьома шарами.

# Вхідні нейрони

## Що таке нейрон?

Нейрон – структурна одиниця нейронної мережі. По своїй суті – вершина, що містить число – активацію.

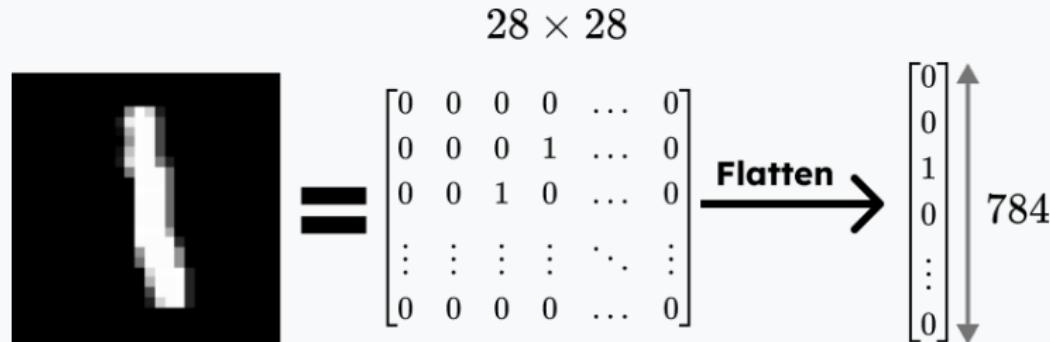
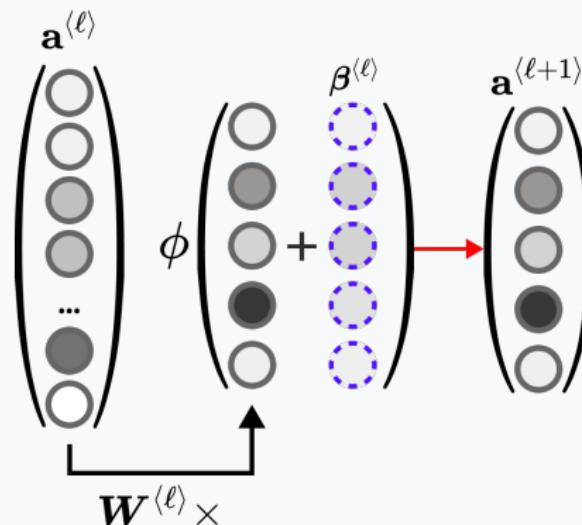


Рис.: Нейрони у першому шарі. Матрицю зображення перетворюємо у плоский вектор.

# Forward Propagation

$$\mathbf{a}^{(\ell+1)} = \phi \left( \mathbf{W}^{(\ell)} \mathbf{a}^{(\ell)} + \boldsymbol{\beta}^{(\ell)} \right).$$



**Рис.:** Пряме поширення. Спочатку знаходимо матричний добуток, додаємо зсув і накладаємо нелінійну функцію.

# Пряме поширення: обрахунок втрати

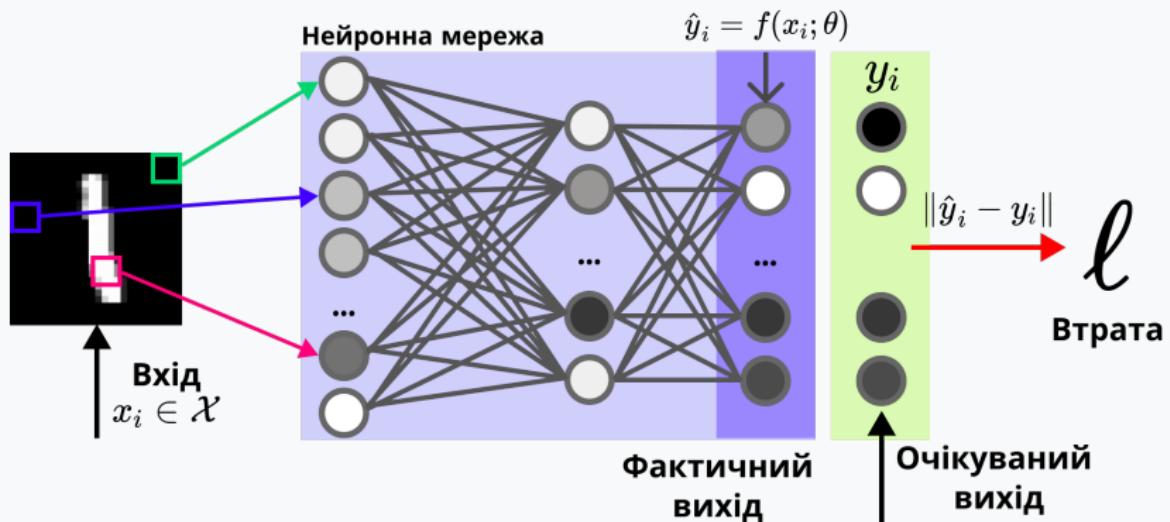


Рис.: Пряме поширення: обрахунок значення втрати

# Резюме

- Нейронна мережа  $f$  – багатопараметризована функція.
- Dense нейронна мережа приймає вектор і “випльовує” також вектор.
- Маючи набір даних, ми намагаємося мінімізувати певну задану функцію  $\ell$ , підбираючи параметри у сімействі функцій.

# Мотивація використання конволюційної нейронної мережі

630



360

$$630 \times 360 = 226,800$$



# Конволюційна операція

$$\begin{array}{|c|c|c|c|c|c|c|c|} \hline
 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ \hline
 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ \hline
 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline
 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 \end{array}
 \quad
 \begin{matrix} * \\ \mathcal{K} \end{matrix}
 \quad
 \begin{array}{|c|c|c|c|c|} \hline
 1 & 0 & 1 & 0 & 1 \\ \hline
 0 & 1 & 0 & 1 & 0 \\ \hline
 1 & 0 & 1 & 0 & 1 \\ \hline
 \end{array}
 \quad
 =
 \quad
 \begin{array}{|c|c|c|c|c|} \hline
 1 & 4 & 3 & 4 & 1 \\ \hline
 1 & 2 & 4 & 3 & 3 \\ \hline
 1 & 2 & 3 & 4 & 1 \\ \hline
 1 & 3 & 3 & 1 & 1 \\ \hline
 3 & 3 & 1 & 1 & 0 \\ \hline
 \end{array}$$

$X$

$\mathcal{K}$

$X * \mathcal{K}$

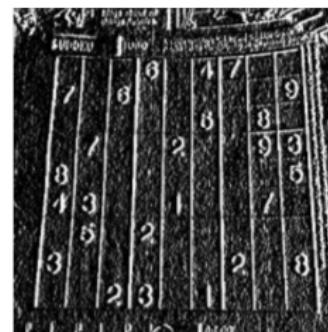
**Рис.:** Механізм знаходження конволюції  $X * \mathcal{K}$  для  $\mathcal{K} \in \mathbb{R}^{3 \times 3}$  та  $X \in \mathbb{R}^{7 \times 7}$ .

# Фільтри Собеля



$$* \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} =$$

*x Sobel kernel*



$$* \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} =$$

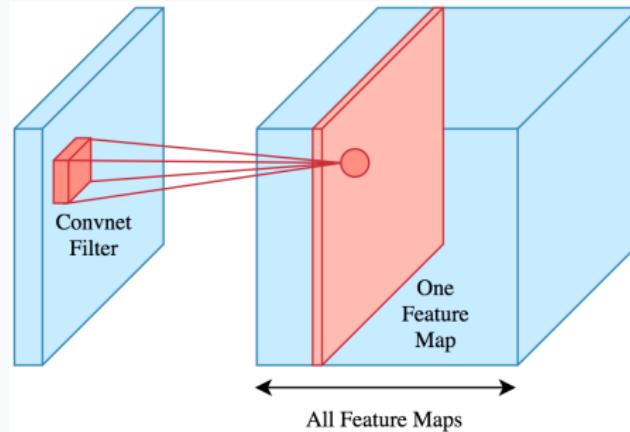
*y Sobel kernel*



Рис.: Демонстрація фільтрів Собеля.

# Конволюційний шар

- Фільтр по зображеню  $W \times H \times n_C$  з  $n_C$  каналами – це набір з  $n_C$  фільтрів.
- Один набір – один шар у вихідному “об’ємі”.  $n_f$  фільтрів дає вихід з  $n_f$  каналами.
- Тренувальні параметри – параметри фільтрів, зсуви (один на кожен фільтр), гіперпараметр – активаційна функція.



# Max Pooling

1. Рухаємо  $2 \times 2 \times n_C$  фільтр по зображеню.
2. Взяти максимальний елемент на кожному каналі і записати у вихідне зображення.

Навіщо це треба? Ми зменьшуємо зображення в 4 рази.

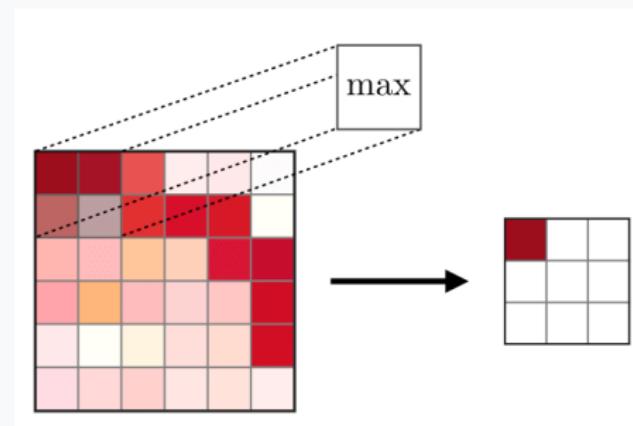


Рис.: Ілюстрація MaxPool шару. Зображення взято з Afshin Amidi, Stanford CS 230 – Deep Learning, CNN Cheatsheet

# Конволюційна нейронна мережа, підсумовуючи

1. Використовуємо конволюційні шари (Conv2D).
2. Зменшуємо розмір зображення за допомогою Conv2D з  $s = 2$  або MaxPool шаром.
3. Повторити, поки об'єм зображення не стане достатньо маленьким.
4. Перевести у повнозв'язану нейронну мережу  $\implies$  вихід.

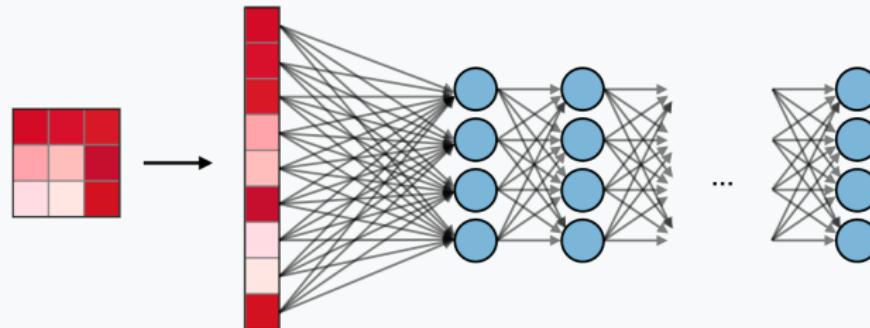
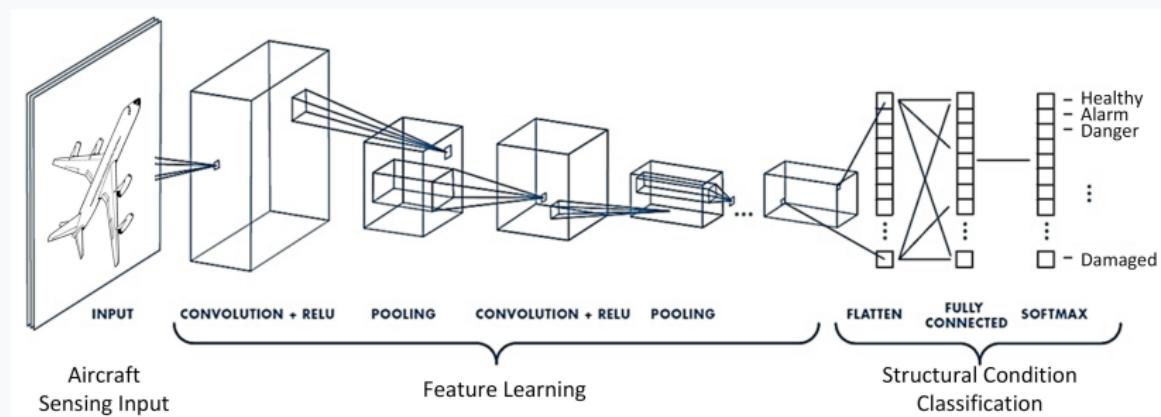


Рис.: Повнозв'язана нейронна мережа в кінці CNN. Зображення взято з Afshin Amidi, Stanford CS 230 – Deep Learning, CNN Cheatsheet

# Конволюційна нейронна мережа, підсумовуючи



**Рис.:** Повний вигляд конволюційної мережі. Зображення взято з роботи Iuliana Tabian et al. 2019. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures.

Вступ

oooooooooooooooooooo

Deep Pattern Recognition

●ooooooooooooooo

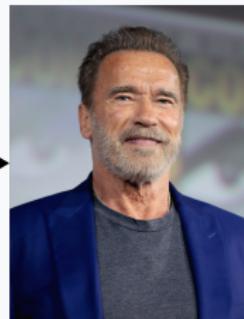
Додаткові розділи біометрії

ooooooooo

# Deep Pattern Recognition

# Постановка задачі

Маючи два зображення  $X, Y$ , сказати чи відповідають вони одній людині чи ні.



# Перша ідея

Нехай кожній людині відповідає номер. Тоді, будуємо класифікаційну нейронну мережу ( $C$  – кількість класів):

$$\mathcal{F} : \text{Image} \rightarrow \{1, \dots, C\}$$



**Рис.:** Даємо номер кожній людині і будуємо мультикласифікаційну модель.

# Чому ні?

- Людей м'яко кажучи багато – близько 8.1 млрд на момент написання цієї презентації :)
- Навіть маючи фіксований набір людей, потрібно більше 50-100 фотографій на людину (One-shot problem).



**Рис.:** Набір класів не є фіксованим, інакше нам доведеться зафіксувати 8+ млрд класів

# Класичні методи: Eigenfaces



Рис.: Test Images.

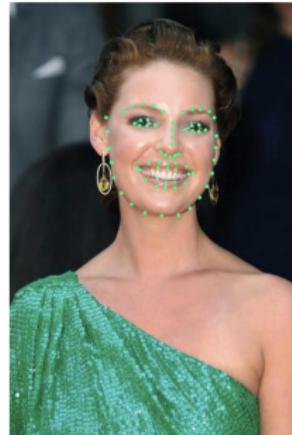


Рис.: Eigenfaces.

$$\text{Image} \approx \lambda_1 \times \text{Eigenface}_1 + \dots + \lambda_N \times \text{Eigenface}_N$$

$$\text{Features} = (\lambda_1, \dots, \lambda_N)$$

# Класичні методи: Facial Feature Points



(a)



(b)



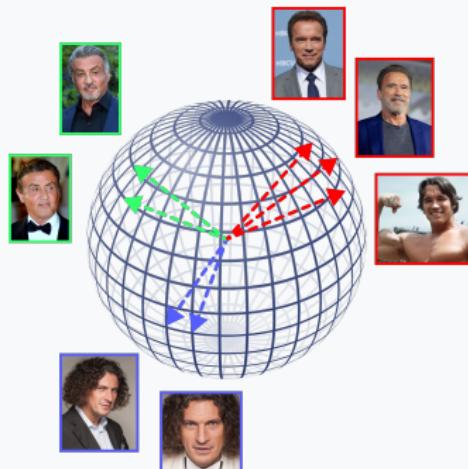
(c)

**Fig. 7** Example faces from (a) LFPW, (b) Helen database and (c) COFW. The dots in deep red indicate corresponding points are occluded.

**Рис.: Facial Feature Points.** Taken from “Facial Feature Point Detection: A Comprehensive Survey” by Nannan Wang et al. 2014.

# Друга ідея: Embedding Neural Network

Будуємо  $\mathcal{F} : \text{Image} \rightarrow S^{m-1}$ . Вперше запропоновано в Florian Schroff et al. "FaceNet: A Unified Embedding for Face Recognition and Clustering". 2015.



**Рис.:** Вихід функції (вектор фіч або "embedding vector") буде давати "характеристику" людини.

# Embedding Neural Network: інтуїція

Приклад вектору фіч – це набір відстаней між ключовими точками.

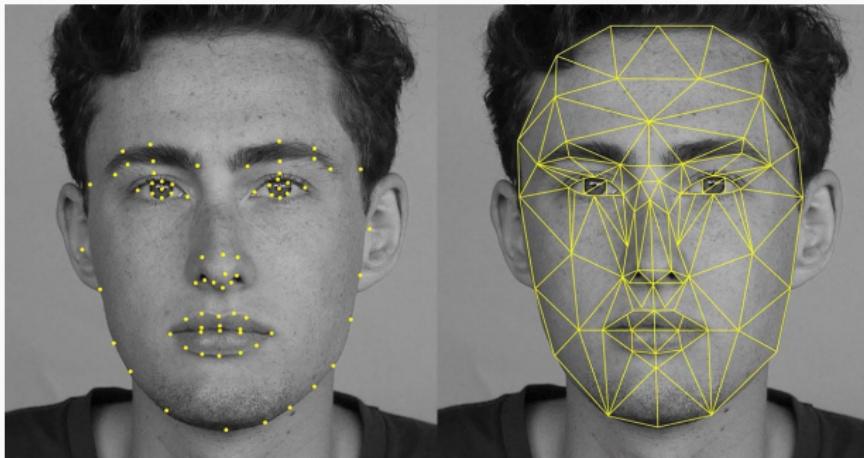


Рис.: Ключові точки на обличчі.

# Ілюстрація роботи Embedding нейронної мережі

## Приклад

Нехай на вхід ми отримали зображення  $X, Y, Z$  і для  $m = 3$  ми отримали наступні вектори фіч:

$$\mathbf{x} = \langle 0.568, 0.568, 0.596 \rangle$$

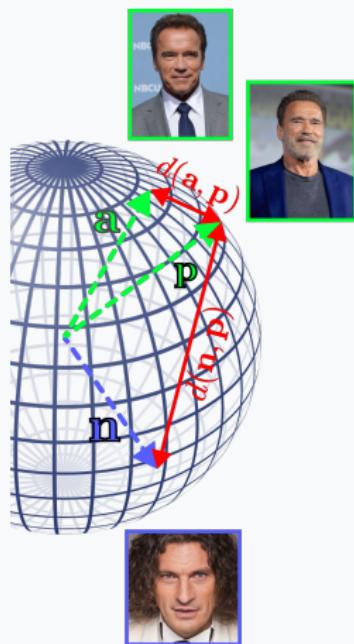
$$\mathbf{y} = \langle 0.613, 0.529, 0.585 \rangle$$

$$\mathbf{z} = \langle -0.408, -0.816, 0.408 \rangle$$

Які два зображення з набору  $X, Y, Z$  належать одній людині, а яка одна до іншої?

Добре видно, що  $X, Y$  належать одній людині. Але як ми це визначили?

# Метрика схожості



Введемо відстань між векторами фіч  $d(\cdot, \star)$ . Найбільш частий вибір – Евклідова метрика.

Умова на одну людину:

$$d(\mathcal{F}(X), \mathcal{F}(Y)) \leq \tau,$$

де  $\tau$  – так званий “поріг” або threshold.

Рис.: Метрика різниці людей – відстань між векторами фіч

# Як реалізувати? Псевдокод

## Реєстрація:

1. Прочитати зображення людини  $X$  зі сканеру.
2. Додати до бази даних  $\mathcal{F}(X)$ .

## Автентифікація:

1. Прочитати зображення людини  $Y$  зі сканеру.
2. Знайти вектор  $y = \mathcal{F}(Y) = (y_1, \dots, y_{128})$ .
3. Для кожного вектору  $z = (z_1, \dots, z_{128})$  з бази даних зробити наступну дію:
  - 3.1. Якщо  $\sum_{i=1}^{128} (y_i - z_i)^2 < \tau$  – впустити людину.
  - 3.2. Якщо ні, то продовжити.

## Резюме

Всі ми – просто набір 128 дійсних чисел на гіперсфері :(

# Як навчати? Головна ідея

Візьмемо три фотографії:

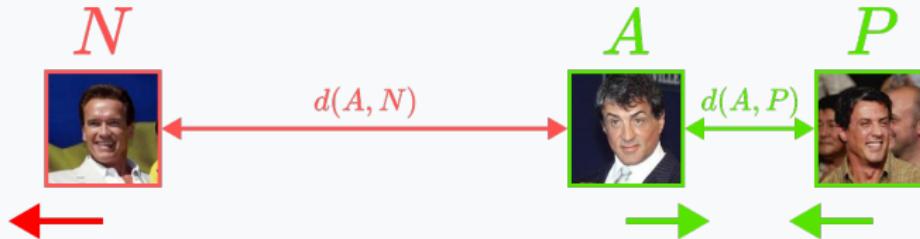
- $A$  (anchor) – зображення людини 1.
- $P$  (positive) – інше зображення людини 1.
- $N$  (negative) – зображення людини 2.

Головне, що ми хочемо:

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

Або, більш строга умова умова:

$$d(A, P) < d(A, N) - \gamma$$

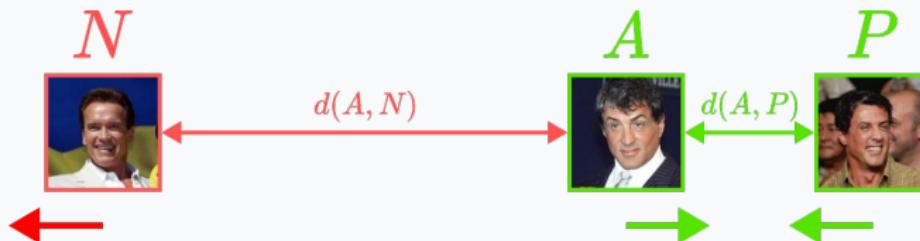


# Як навчати? Робимо метрику!

Як з цієї ідеї зробити конкретну метрику (втрату)?

Використовується наступна функція:

$$\ell(A, P, N) = \max \{d(A, P) - d(A, N) + \gamma, 0\}$$



**Рис.:** Після застосування градієнту, ми хочемо віддалити ( $A, N$ ) і наблизити ( $A, P$ ).

# Як навчати? Triplet Network

Як побудувати нейронну мережу? Triplet Network!

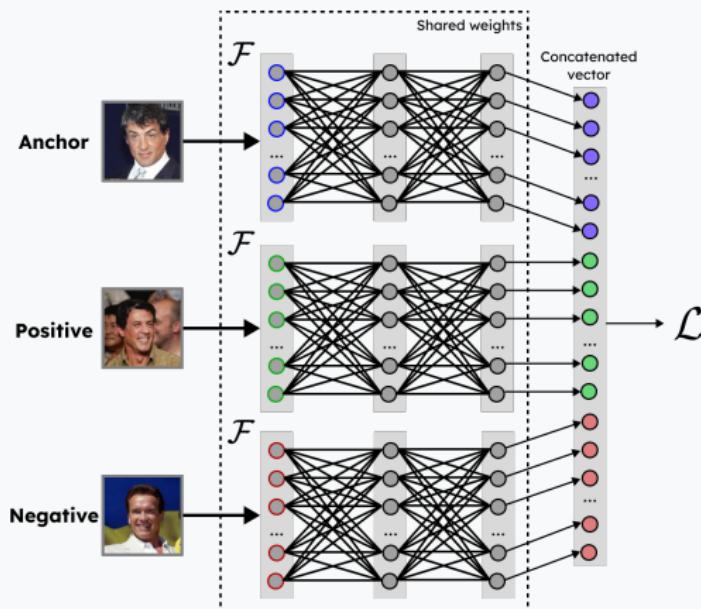


Рис.: Triplet Network архітектура.

# Вектори фіч поза біометрією

Поняття вектору характеристик використовується усюди, бо це, по суті, стискання інформації в одному векторі. Зокрема:

1. Геолокація
2. Текстові дані
3. Рекомендаційні системи
4. Голос людини
5. ...

# Додаткові розділи біометрії

# Безпека векторів фіч

- Зберігати зображення в базі даних небезечно.
- Чи безпечно зберігати вектори фіч? Ні.
- Чи можна створити криптографічний примітив? Дуже активна робота саме в цьому напрямку!

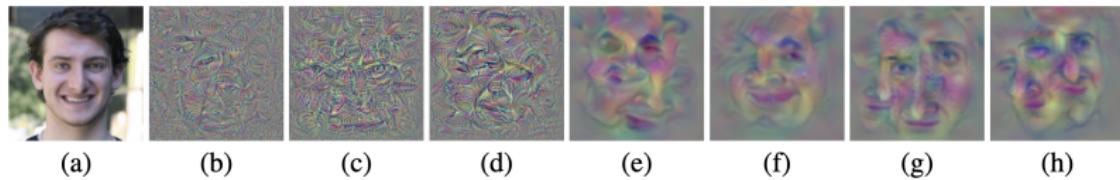


Figure 2: (a) is the desired image whose FaceNet embedding is used. (b) and (c) are generated using just facenet embedding loss. (d), (e), (f) are generated using increasing total variation loss, making an increasingly smoother and more image. (g) and (h) are generated using guiding image loss on an intermediate FaceNet layer. We observe increasing identifiability, but not a convincing result.

**Рис.:** Знаходження обличчя з векторів фіч. Взято з Edward Vendrow et al. 2018. Inverting Facial Embeddings with GANs

# Безпека векторів фіч

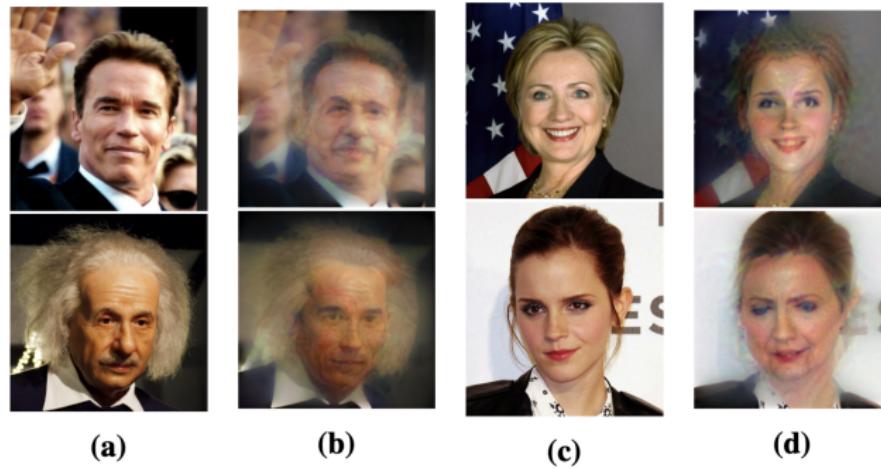


Рис.: Обернення FaceNet, використовуючи базове зображення. Andrey Zhmoginov et al. “Inverting face embeddings with convolutional neural networks”

## Формулювання



**Рис.:** Постановка задачі. Спираючись на зображення  $X \in \mathcal{I}$ , видати ймовірність того, що перед нами нереальна людина.

# Архітектура

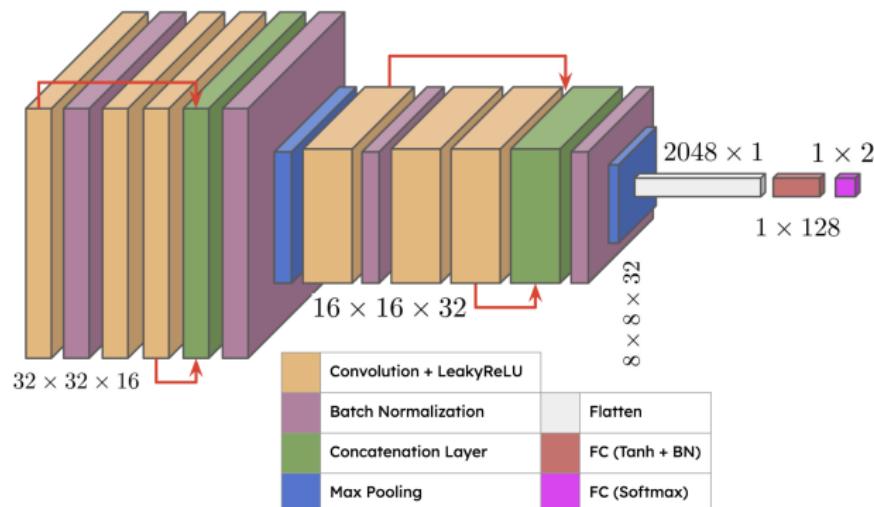


Figure 1: *AttackNet* Architecture [24]

Рис.: Архітектура *AttackNet* v2.2.

# Набори даних

**3DMAD****CSMAD****MSSPoof****Replay-Attack****Our Dataset**

Рис.: Набори даних, на яких проводилося тренування.

# Attention Maps

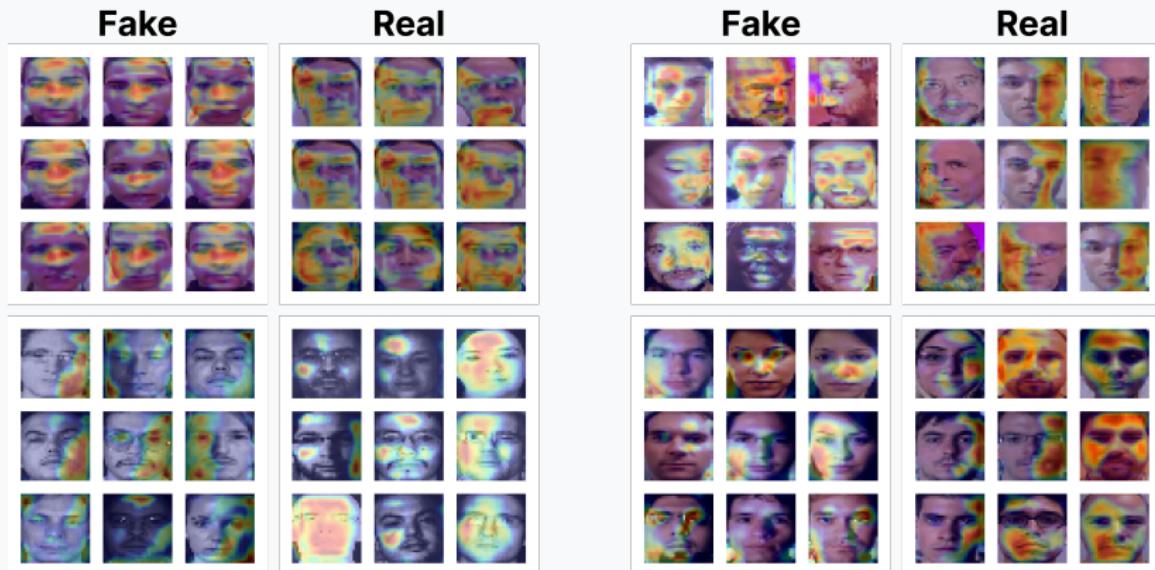


Рис.: Які зони облич нейронна мережа вважає важливими?

# Найбільш актуальні напрямки досліджень

- Мультибіометрія.
- Формальне поняття безпеки біометрії в контексті криптографії.
- Гомоморфні нейронні мережі.
- Zero-Knowledge Machine Learning.

Дякую за Вашу Увагу!

