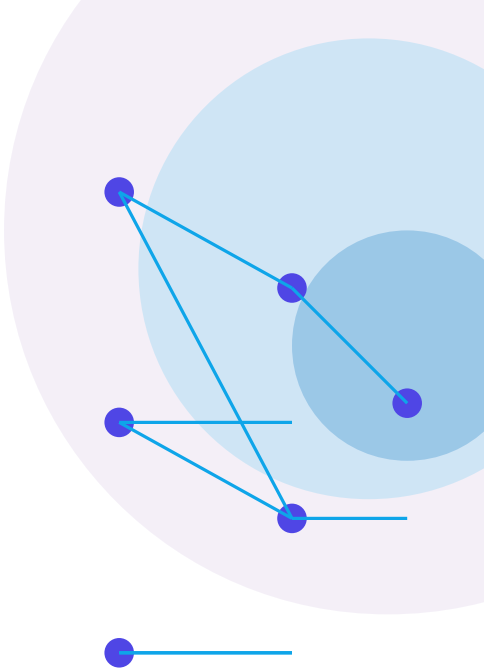
ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ



НЕЙРОННІ МЕРЕЖІ

Від нейрона до глибоких архітектур: MLP · CNN · RNN

ЗМІСТ ПРЕЗЕНТАЦІЇ

01

НЕЙРОН І ПРОСТІ МЕРЕЖІ

- Біологічний та штучний нейрон
- Функції активації
- Перцептрон · MLP
- Backpropagation
- Функції втрат · Оптимізатори · Регуляризація

02

ЗГОРТКОВІ МЕРЕЖІ (CNN)

- Операція згортки та ядра
- Шари пулінгу
- ResNet, VGG, GoogLeNet
- Transfer Learning

03

РЕКУРЕНТНІ МЕРЕЖІ (RNN)

- Класичний RNN · BPTT
- LSTM та GRU
- Seq2Seq · Attention
- Трансформери (огляд)

01

НЕЙРОН І ПРОСТІ МЕРЕЖІ

Від біологічного нейрона до багатошарового перцептрона

БІОЛОГІЧНИЙ НЕЙРОН

Дендрити

Отримують вхідні сигнали від інших нейронів. Аналог: входи $x_1, x_2, \ldots, x_n$

Тіло клітини (сома)

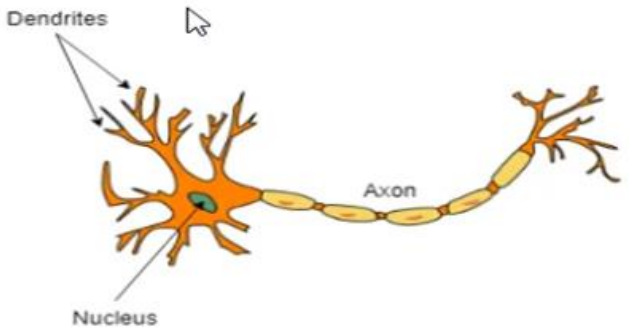
Підсумовує вхідні сигнали. Аналог: зважена сума $\Sigma w_i x_i + b$

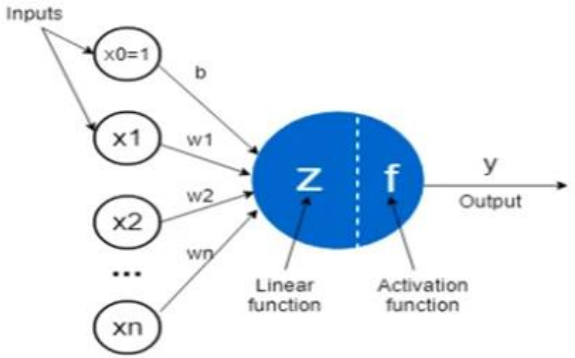
Аксон

Передає вихідний сигнал іншим нейронам. Аналог: вихід y = f(net)

Синапс

З'єднання між нейронами, що має певну «силу». Аналог: вага w_i





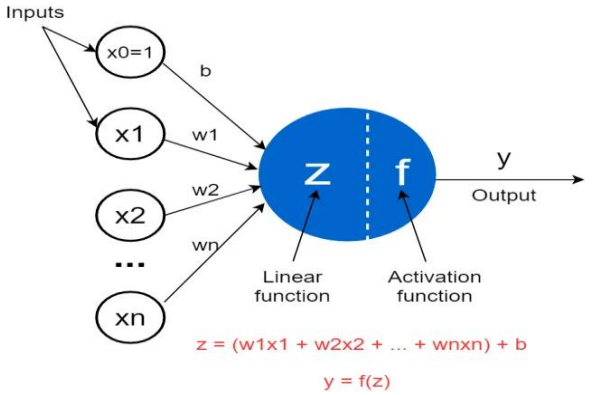
Штучний нейрон моделює основні функції біологічного: отримує сигнали, зважує їх і передає через функцію активації.

МАТЕМАТИЧНА МОДЕЛЬ НЕЙРОНА

$$net = \Sigma_i w_i \cdot x_i + b = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b$$

$$y = f(net) \quad \text{де } f - \text{функція активації}$$

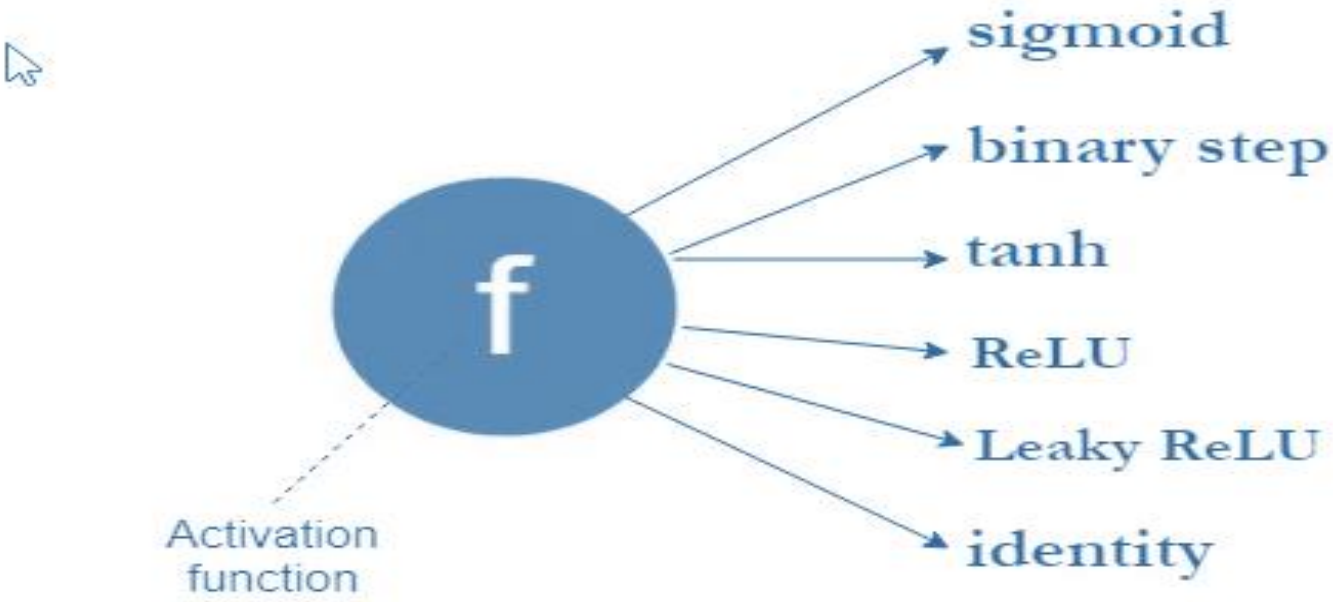
$$\text{Матрична форма: } y = f(W^Tx + b)$$



Позначення

x_i	Вхідні ознаки (features)
net	Зважена сума — лінійна комбінація
w_i	Ваги — навчувані параметри
$f(\cdot)$	Функція активації — нелінійність
b	Зміщення (bias) — зсув порогу
y	Вихід нейрона

ФУНКЦІЯ АКТИВАЦІЇ



ФУНКЦІЇ АКТИВАЦІЇ

Sigmoid

$$\sigma(x) = 1 / (1 + e^{-x})$$

Діапазон: (0, 1)

+ Гладка, імовірнісна

− Vanishing gradient

Tanh

$$\tanh(x) = (e^{x}-e^{-x})/(e^{x}+e^{-x})$$

Діапазон: (−1, 1)

+ Центрована навколо 0

− Vanishing gradient

ReLU

$$\mathrm{ReLU}(x) = \max(0, x)$$

Діапазон: [0, +∞)

+ Швидка, без насичення

− Dying ReLU проблема

Leaky ReLU

$$f(x) = \max(\alpha x, x), \quad \alpha \approx 0.01$$

Діапазон: (−∞, +∞)

+ Вирішує dying ReLU

− Гіперпараметр α

Softmax

$$\sigma_i = e^{x_i} / \Sigma_j e^{x_j}$$

Діапазон: (0,1), сума=1

+ Вихідний шар (класиф.)

− Лише фінальний шар

GELU

$$\mathrm{GELU}(x) = x \cdot \Phi(x)$$

Діапазон: ≈(−0.17, +∞)

+ Transformer default

− Обчислювально складна

ПЕРЦЕПТРОН (Rosenblatt, 1958)

Визначення

Найпростіший нейрон з бінарним виходом. Класифікує лінійно роздільні набори даних.

$$\hat{y} = 1 \quad \text{якщо} \quad \Sigma w_i x_i + b \geq 0$$

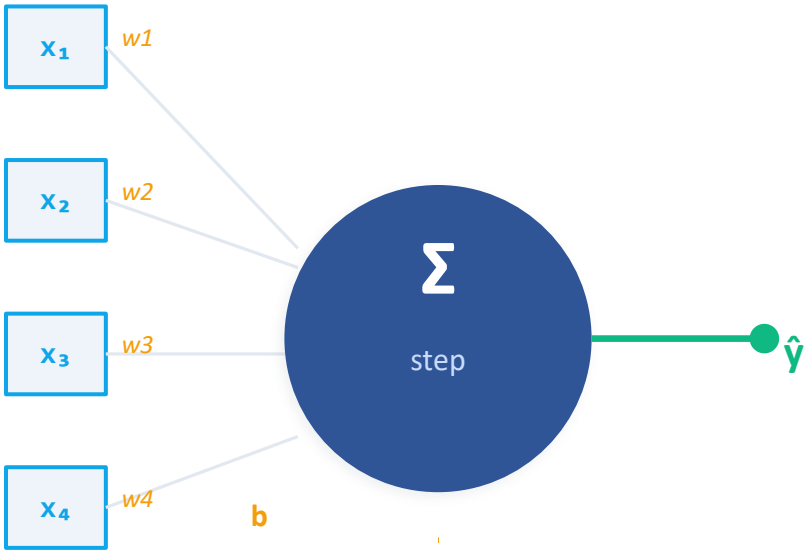
$$\hat{y} = 0 \quad \text{якщо} \quad \Sigma w_i x_i + b < 0$$

Правило навчання (delta rule):

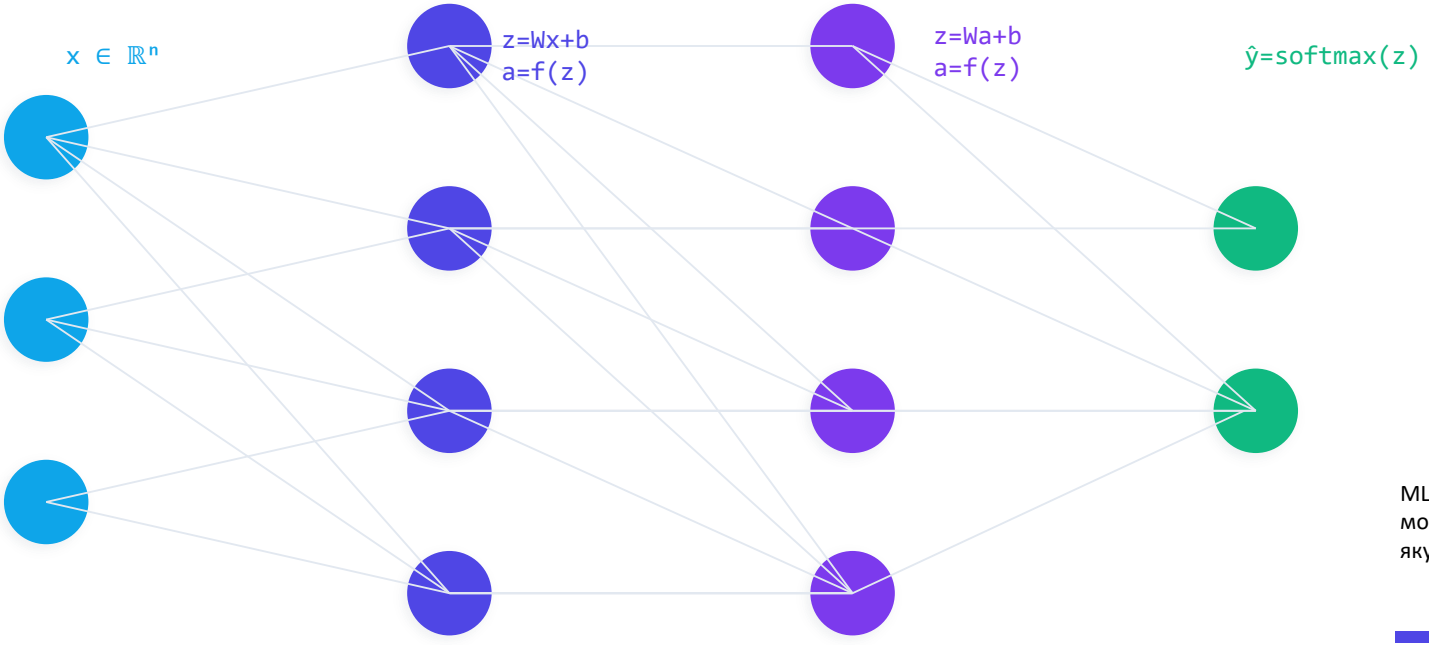
$$w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$$

η — швидкість навчання (learning rate)

Обмеження: не може вирішити нелінійно нероздільні задачі (XOR).



БАГАТОШАРОВИЙ ПЕРЦЕПТРОН (MLP)



MLP з ≥1 прихованим шаром може апроксимувати будь-яку неперервну функцію.

ЗВОРОТНЕ ПОШИРЕННЯ ПОМИЛКИ (BACKPROPAGATION)

1

Forward Pass

Подаємо x, обчислюємо активації a^l по всіх шарах

$$a^l = f(W^l a^{l-1} + b^l)$$

2

Функція втрат

Порівнюємо вихід із правильною відповіддю

$$L = (1/m) \Sigma \, loss(\hat{y}_i, y_i)$$

3

Backward — вихідний шар δ^L

Похідна втрат по активаціях (delta-правило)

$$\delta^L = \nabla_a L \odot f'(z^L)$$

4

Backward — прихований шар δ^l

Поширюємо помилку назад через chain rule

$$\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot f'(z^l)$$

5

Оновлення параметрів

Рухаємось у напрямку спадання градієнта

$$W^l \leftarrow W^l - \eta \cdot \delta^l (a^{l-1})^T$$

Chain rule: $\partial L/\partial W^l = \delta^l \cdot (a^{l-1})^T$ $\partial L/\partial b^l = \delta^l$

ФУНКЦІЇ ВТРАТ (LOSS FUNCTIONS)

MSE
Mean Squared Error

$$(1/n)\Sigma(y_i-\hat{y}_i)^2$$

Задача: Регресія

Карає великі помилки

MAE
Mean Absolute Error

$$(1/n)\Sigma|y_i-\hat{y}_i|$$

Задача: Регресія

Стійка до викидів

Binary CE
Binary Cross-Entropy

$$-(y\cdot\log\hat{y}+(1-y)\log(1-\hat{y}))$$

Задача: Бінарна кл.

Вихід: sigmoid

Categ. CE
Categorical Cross-Entropy

$$-\Sigma_i\ y_i\cdot\log(\hat{y}_i)$$

Задача: Мульти-кл.

Вихід: softmax

Hinge
Hinge Loss

$$\Sigma\ \max(0,\ 1\ -\ y_i\cdot\hat{y}_i)$$

Задача: SVM / класиф.

Великий відступ

KL Div.
Kullback–Leibler Divergence

$$\Sigma P(x)\cdot\log(P(x)/Q(x))$$

Задача: VAE · генеративні

Розбіжність розподілів

АЛГОРИТМИ ОПТИМІЗАЦІЇ

SGD

$$w \leftarrow w - \eta \cdot \nabla L$$

Стохастичний GD. Простий але нестабільний. Потребує ручного підбору η.

Momentum

$$v \leftarrow \gamma v + \eta \cdot \nabla L$$
$$w \leftarrow w - v$$

Накопичує імпульс руху (γ≈0.9). Прискорює збіжність у стабільних напрямах.

RMSProp

$$E[g^2] \leftarrow \rho E[g^2] + (1-\rho) g^2$$
$$w \leftarrow w - \eta / \sqrt{(E[g^2] + \varepsilon)}$$

Адаптивний η. Ефективний в RNN та онлайн-задачах.

Adam

$$m \leftarrow \beta_1 m + (1-\beta_1) g$$
$$v \leftarrow \beta_2 v + (1-\beta_2) g^2$$
$$w \leftarrow w - \eta \cdot \hat{m} / \sqrt{(\hat{v} + \varepsilon)}$$

Momentum + RMSProp. β_1=0.9, β_2=0.999. Найпопулярніший оптимізатор.

✓ Default

Рекомендовано: Adam із η=0.001, β_1=0.9, β_2=0.999, ε=1e-8 як стартова точка для більшості задач.

РЕГУЛЯРИЗАЦІЯ ТА НОРМАЛІЗАЦІЯ

L2 (Weight Decay)

$$L_{total} = L + \lambda \cdot \Sigma w^2$$

Штрафує великі ваги. Запобігає overfitting.

L1 (Lasso)

$$L_{total} = L + \lambda \cdot \Sigma |w|$$

Обнуляє частину ваг → sparse мережа.

Dropout

$$p(\text{вимкнено}) = p \approx 0.3\text{–}0.5$$

Випадково вимикає нейрони під час навчання.

Early Stopping

Стоп якщо val_loss ↑ k епох

Зупиняємо при погіршенні val-метрики.

Batch Norm (BN)

$$\hat{x} = (x - \mu_B) / \sqrt{(\sigma^2_B + \varepsilon)} \rightarrow \gamma\hat{x} + \beta$$

Нормалізує активації батчу. Прискорює навчання.

Layer Norm (LN)

$$\hat{x} = (x - \mu_L) / \sqrt{(\sigma^2_L + \varepsilon)}$$

По ознаках (не батчу). Стандарт у Transformer.

ПРИКЛАД

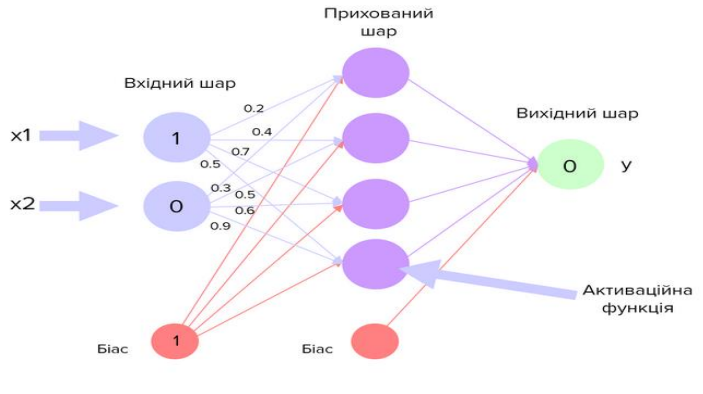
У нейронній мережі прямого розповсюдження кожен вихід нейрону стає входом для наступного шару нейронів.

1) Вхід до прихованого шару обчислюється за формулою:

$$Z_{np} = \sum_{i=1}^{N} x_i w_i + b_1$$

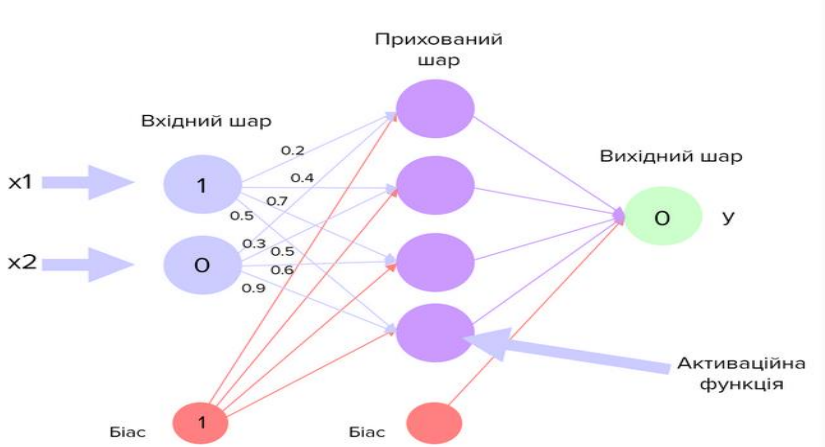
де *N* - кількість нейронів у вхідному шарі (дорівнює кількості ознак)
xi - вхідні дані
wi - ваги
*b*1 - біас

Нехай ваги та біас мають наступні значення:
*w*1=0.2, *w*5=0.3, *b*1=1.
*w*2=0.4, *w*6=0.5,
*w*3=0.7, *w*7=0.6,
*w*4=0.5, *w*8=0.9,



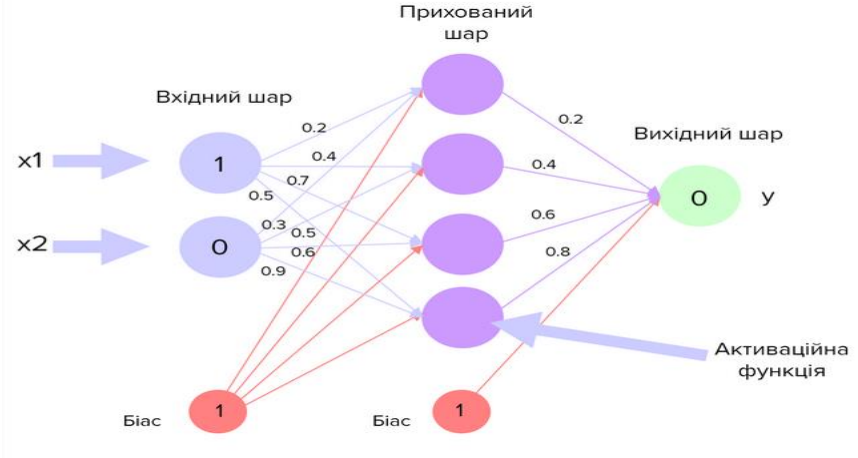
ПРИКЛАД

Кожен нейрон прихованого шару містить активаційну функцію. Вона може бути будь-якою, але будемо використовувати сігмоїд-функцію



Ця функція набуває значень на проміжку [0;1].

$$y_{np} = \frac{1}{1 + e^{-z_{np}}}$$



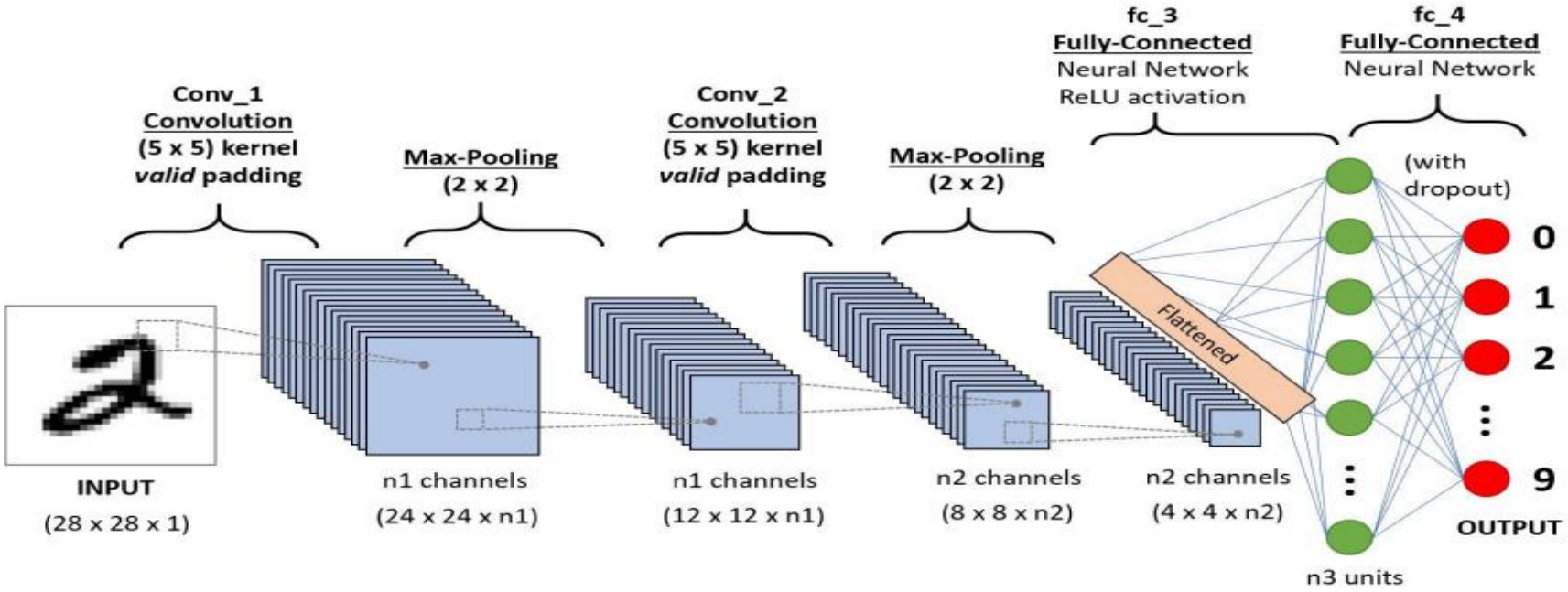
02

ЗГОРТКОВІ МЕРЕЖІ (CNN)

Просторова обробка даних: зображення, відео, сигнали

ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ (CNN)

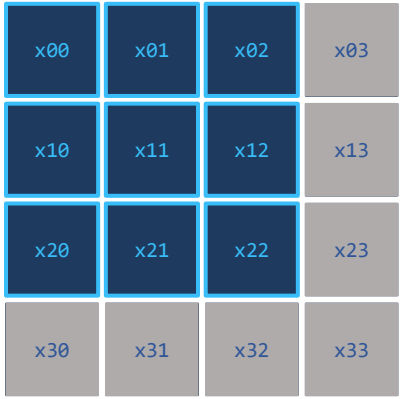
CNN — спеціалізована нейронна мережа для обробки даних з просторовою структурою (зображення, звук, текст). Використовує операцію згортки для локального видобутку ознак замість повно-з'єднаних шарів.



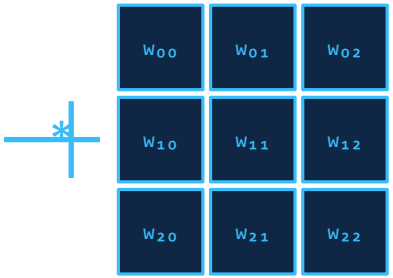
ЗАГАЛЬНИЙ ПОТІК: Вхід → [Conv → BN → ReLU]×N → Pooling×M → Flatten → FC → Softmax → Клас

Шар Згортки

Convolutional Layer

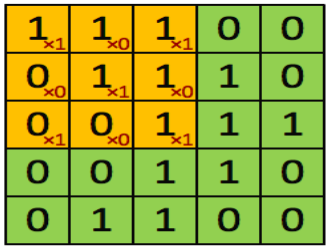


Вхід (4×4)

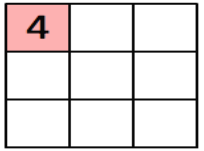


Фільтр (3×3)

Карта ознак (2×2)



Image



Convolved Feature

Формула

$$(I * K)[i,j] = \Sigma\Sigma\, I[i+m,j+n,c] \cdot K[m,n,c] + b$$

Розмір виходу

$$W_{out} = \lfloor (W - F + 2P) / S \rfloor + 1$$

Параметри (3×3×C×K)

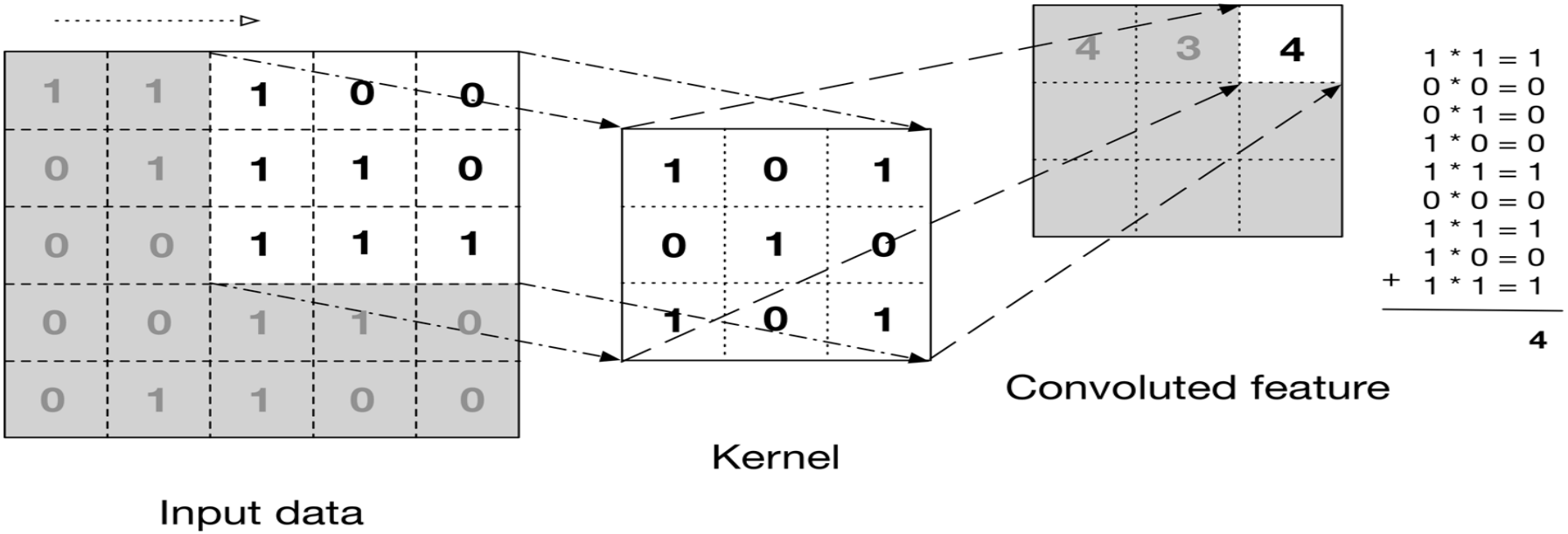
$$F \times F \times C_{in} \times K + K$$

Приклад (3×3, 32 фільтри, 3 канали)

$$3 \times 3 \times 3 \times 32 + 32 = 896 \text{ параметрів}$$

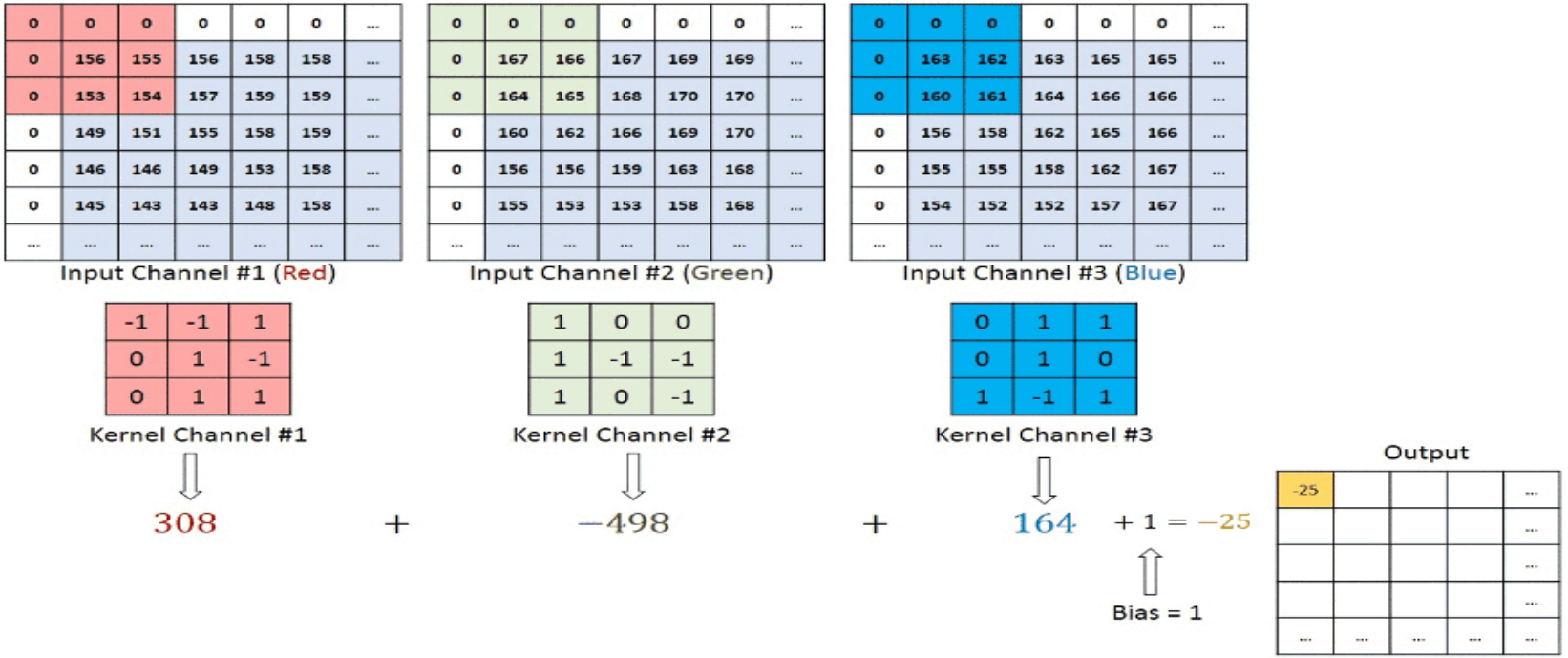
Шар Згортки

Convolutional Layer



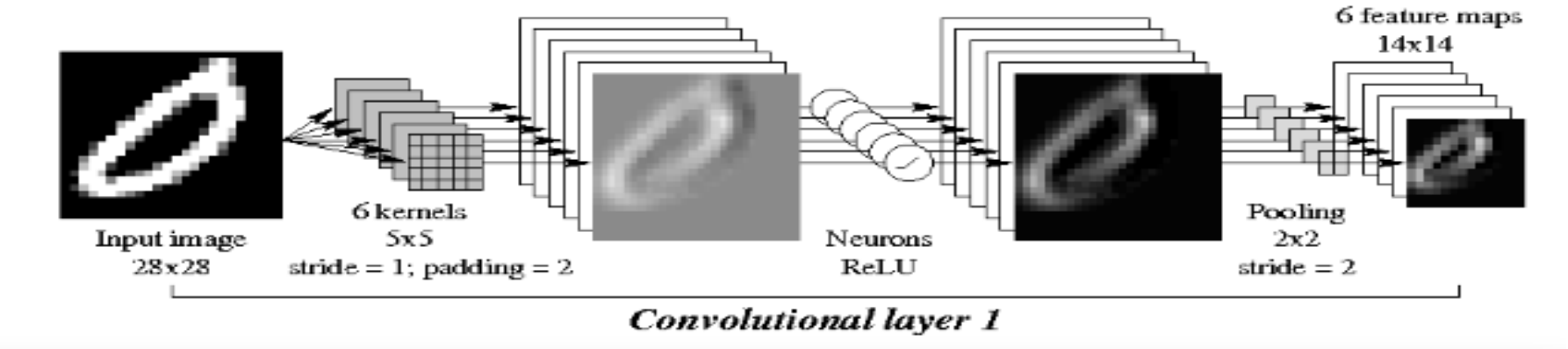
Шар Згортки

Convolutional Layer



Шар Згортки

Convolutional Layer



Згорткова нейронна мережа для задачі MNIST: детальний вигляд першого згорткового шару.

ШАРИ ПУЛІНГУ ТА АРХІТЕКТУРНИЙ ПАЙПЛАЙН CNN

Max Pooling

```
Out[i,j] = max(вікно 2×2)
```

Вибирає максимум. Зберігає найяскравіші ознаки. Найпоширеніший.

Average Pooling

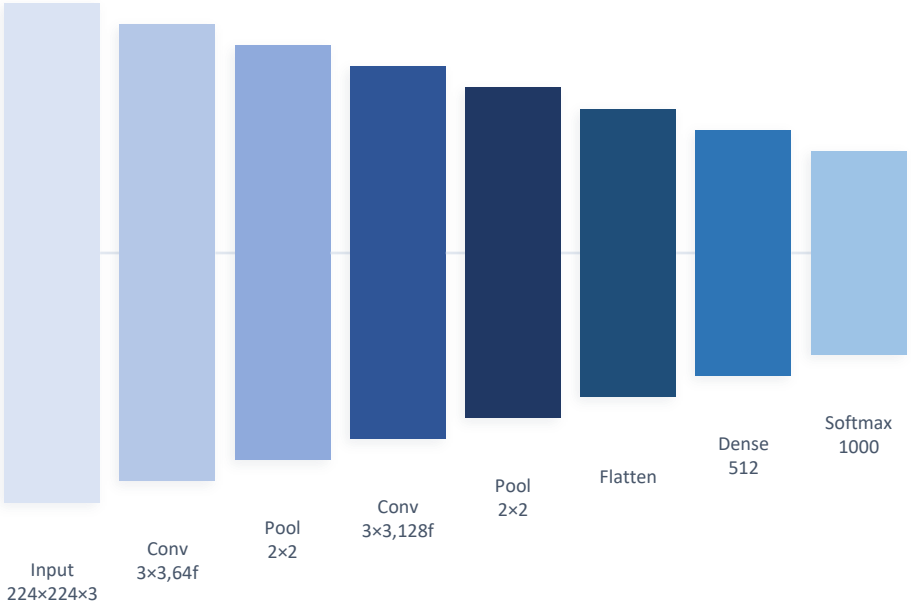
```
Out[i,j] = mean(вікно 2×2)
```

Середнє вікна. М'якша версія, зберігає більше контексту.

Global Avg Pool

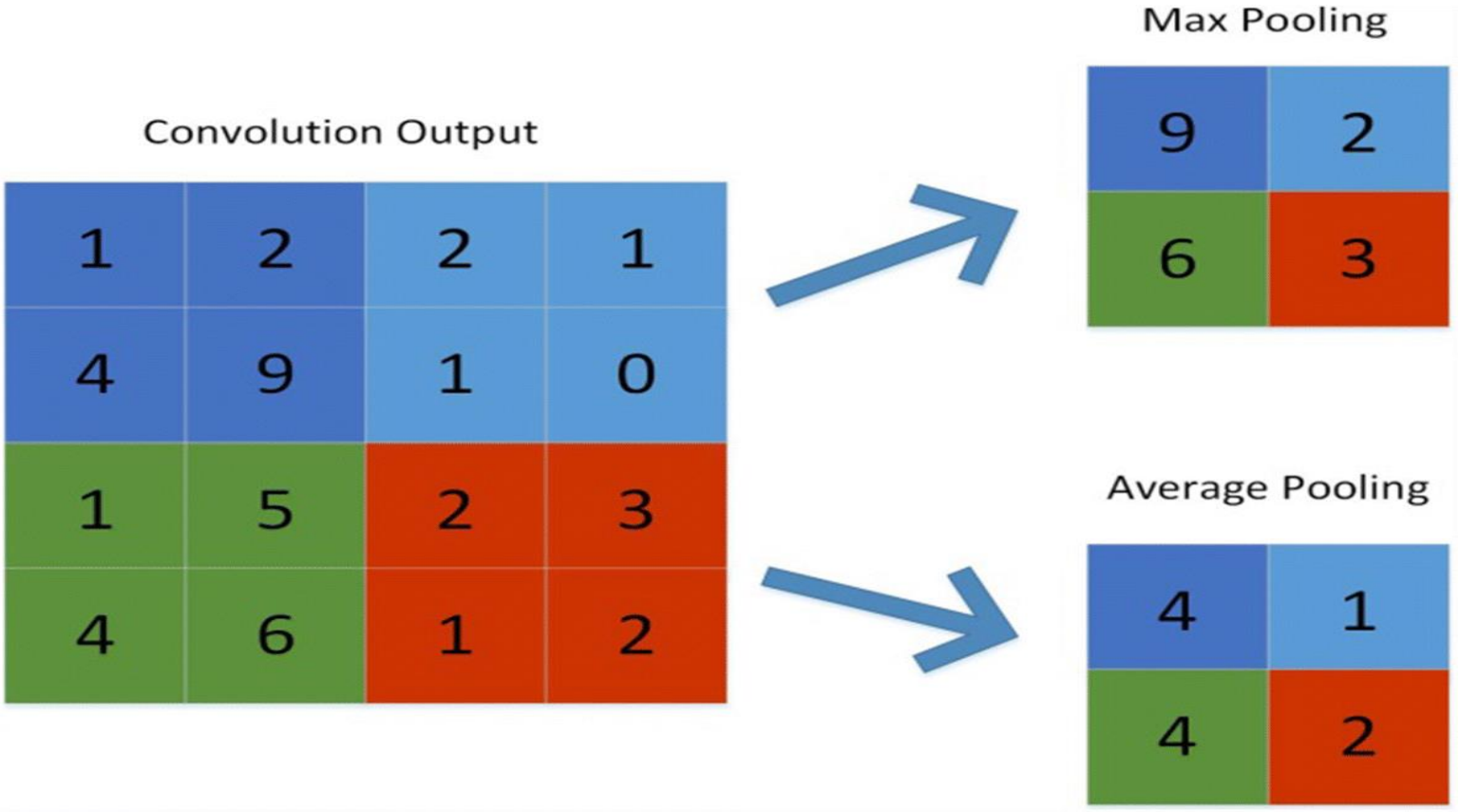
```
Out = mean(карта H×W)
```

Замінює Dense-шар. Зменшує кількість параметрів у кінці CNN.

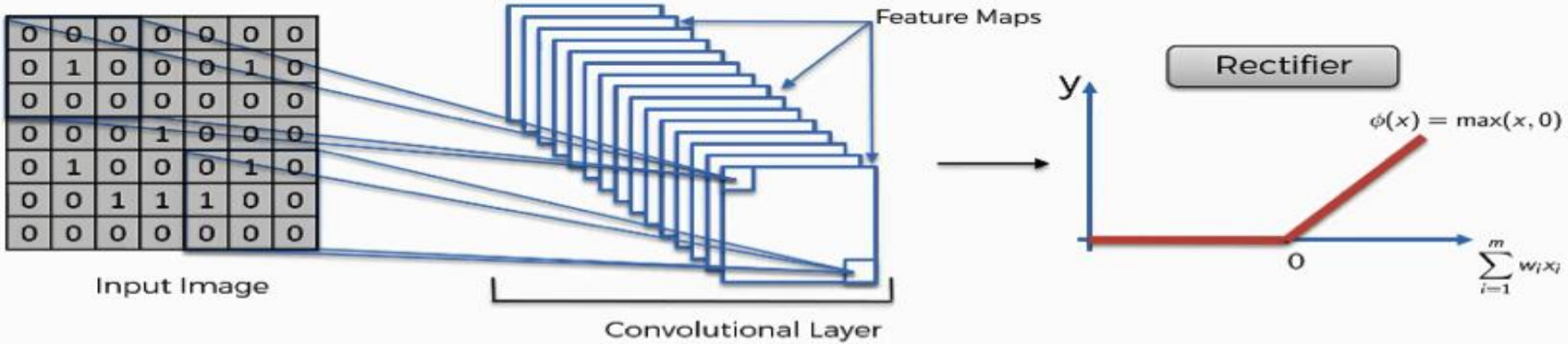


Принцип: Conv нарощують C_{out} (глибина), Pool зменшують H×W (простір). Dense робить класифікацію.

ШАРИ ПУЛІНГУ ТА АРХІТЕКТУРНИЙ ПАЙПЛАЙН CNN



ШАРИ ПУЛІНГУ ТА АРХІТЕКТУРНИЙ ПАЙПЛАЙН CNN



Математичне визначення функції активації ReLU - $f(x) = \max(0.0, x)$

Dropout · Flatten · FC · Softmax

Dropout

Регуляризація

Під час навчання кожен нейрон деактивується з ймовірністю p (0.2–0.5). Запобігає перенавчанню.
f: $y_i = x_i \cdot B_i/(1-p)$, $B_i \sim$ Bernoulli

Flatten

Розгортання

3D тензор → 1D вектор для FC-шару.
7×7×512 = 25,088 значень
АБО: GAP: H×W×C → C (512)

Fully Connected

Повнозв'язний

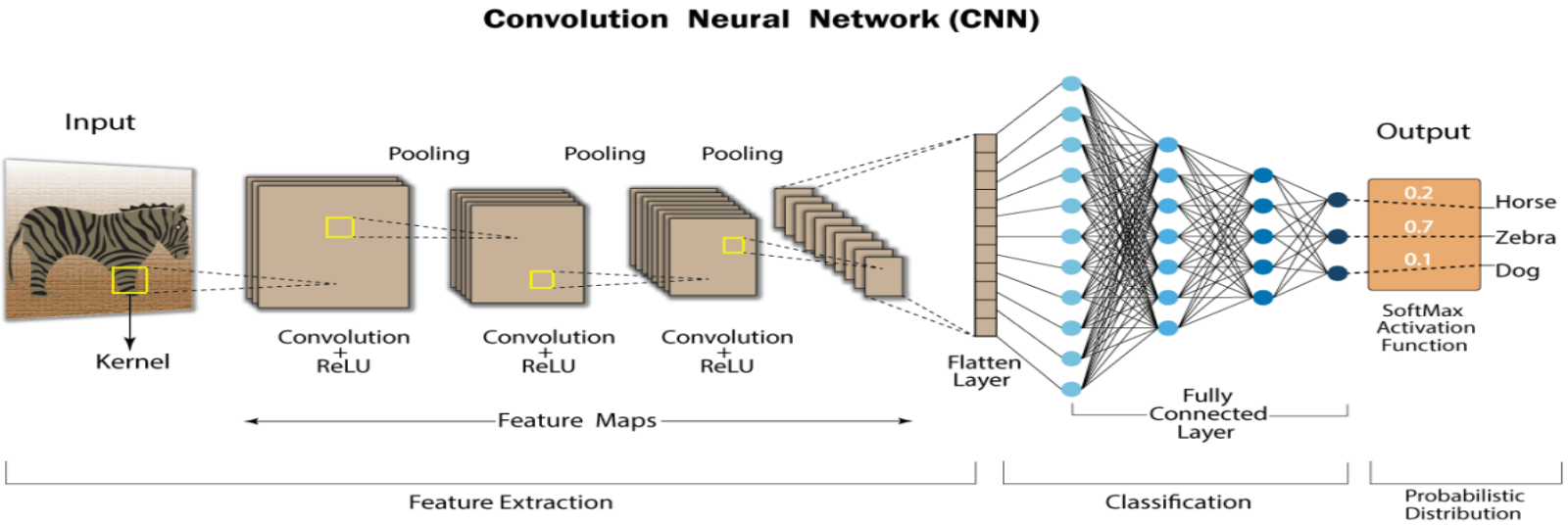
$y = f(Wx + b)$, $W \in \mathbb{R}^{(out \times in)}$
Параметри: (in+1)×out
25088×512 = 12,845,056 ваг!

Softmax

Вихідний шар

$\sigma(z)_i = e^{z_i} / \Sigma_j e^{z_j}$
$\Sigma\sigma(z)_i = 1$ | Loss: $-\Sigma y_i \log(\hat{y}_i)$
1000 класів (ImageNet)

ШАРИ ПУЛІНГУ ТА АРХІТЕКТУРНИЙ ПАЙПЛАЙН CNN



Ієрархія ознак:

Краї та градієнти	Кути та текстури	Форми та частини	Об'єкти та концепції

ЕВОЛЮЦІЯ АРХІТЕКТУР CNN

1989

LeNet-5

Точність: ~99% MNIST

Парам: 60K

Перша успішна CNN (LeCun). Conv+Pool+Dense.

2012

AlexNet

Точність: Top-5: 15.3%

Парам: 60M

Виграш ImageNet. ReLU, Dropout, GPU навчання.

2014

VGG-16

Точність: Top-5: 7.3%

Парам: 138M

Лише 3×3 фільтри, глибина 16–19 шарів.

2014

GoogLeNet

Точність: Top-5: 6.7%

Парам: 7M

Inception-модулі. Паралельні conv різних розмірів.

2015

ResNet-50

Точність: Top-5: 3.6%

Парам: 25M

Skip connections: F(x)+x. 152 шари.

2017

DenseNet

Точність: SOTA

Парам: 8M

Кожен шар з'єднаний з усіма попередніми.

2019

EfficientNet

Точність: Top-1: 84%+

Парам: 5–66M

Compound scaling: ширина + глибина + роздільн.

2021

Vision Transformer

Точність: 87%+

Парам: 86M+

Self-Attention замість Conv. Patches як токени.

ЗАСТОСУВАННЯ CNN

Класифікація зображень

ImageNet, медична діагностика (МРТ, рентген), контроль якості на виробництві

`SOTA: Top-1 ~90%+`

Object Detection

YOLO, SSD, Faster R-CNN: знаходить і локалізує об'єкти з bounding boxes

`SOTA: COCO mAP ~60%`

Semantic Segmentation

U-Net, DeepLab: класифікація кожного пікселя. Медицина.

`SOTA: mIoU ~85%`

Face Recognition

FaceNet, ArcFace: ідентифікація та верифікація осіб (FaceID на телефонах)

`SOTA: LFW ~99.8%`

OCR / Document AI

Розпізнавання тексту з зображень, аналіз документів, перевірка підписів

`SOTA: ~99% (друк)`

Ігри та Reinforcement L.

Atari, AlphaGo — CNN обробляє стан поля, передає в policy network

`SOTA: superhuman`

TRANSFER LEARNING У CNN

Ідея: взяти модель, навчену на ImageNet (1.2M зображень), і донавчити на власних даних

1

Feature Extraction

Заморозити всі conv-шари. Замінити лише Dense-шар. Навчати тільки новий класифікатор.

Підходить: мало даних (100–1000 прикладів).
η = 0.001

2

Fine-tuning (частковий)

Заморозити перші шари (загальні ознаки). Розморозити останні conv-блоки. Навчати з малим η.

Підходить: середня кількість даних.
η = 1e-5

3

Full Fine-tuning

Розморозити всю мережу. Навчати з дуже малим learning rate. Потрібно багато даних.

Підходить: великий датасет (10K+).
η = 1e-6

Популярні базові моделі: ResNet50/101, EfficientNetB0–B7, VGG16, InceptionV3, MobileNetV2

03

РЕКУРЕНТНІ МЕРЕЖІ (RNN)

Обробка послідовностей: текст, мовлення, часові ряди

РЕКУРЕНТНА НЕЙРОННА МЕРЕЖА (RNN)

Математична модель

RNN обробляє послідовності $x_t \in \mathbb{R}^n$ по одному кроку t=1,…,T:

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

$$\hat{y}_t = \mathrm{softmax}(W_y \cdot h_t + b_y)$$

Позначення:

x_t — вхідний вектор на кроці t

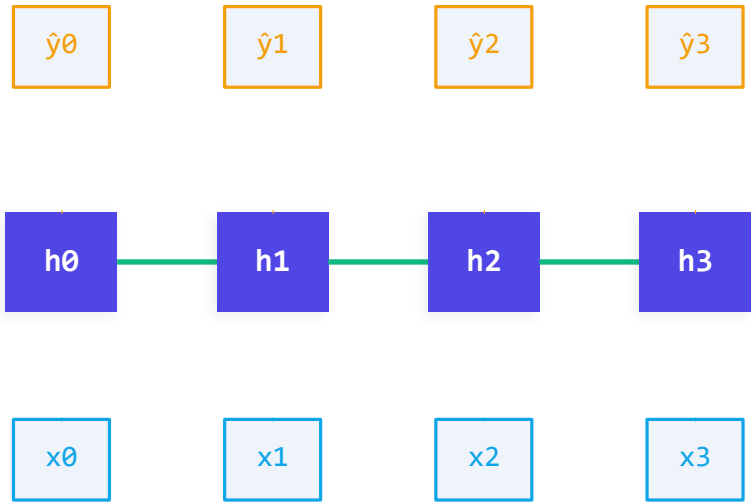
h_t — прихований стан — пам'ять

h_{t-1} — попередній прихований стан

W_h, W_x — спільні ваги для всіх t

$\hat{y}_t$ — вихід на кроці t

Розгорнута RNN у часі (навч.: BPTT)



⚠ Vanishing gradient при T>20. Рішення: LSTM та GRU.

LONG SHORT-TERM MEMORY (LSTM, Hochreiter 1997)

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \qquad h_t = o_t \odot \tanh(c_t)$$

f_t Forget Gate

$$\sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Вирішує, яку частину c_{t-1} зберегти (0=забути, 1=залишити)

i_t Input Gate

$$\sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Скільки нової інформації записати до пам'яті

$\tilde{c}_t$ Cell Update

$$\tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Новий кандидат у пам'ять (масштабується на i_t)

o_t Output Gate

$$\sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Скільки пам'яті видати як прихований стан h_t

Ключ LSTM: cell state c_t — «транспортна стрічка» — передає довгострокову пам'ять без перемноження градієнтів.

GATED RECURRENT UNIT (GRU, Cho et al. 2014)

Рівняння GRU

Update Gate

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

Reset Gate

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

Candidate State

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$

Output

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

Update gate z_t: скільки попередн. стану зберегти. Reset gate r_t: скільки пам'яті враховувати.

LSTM vs GRU

	LSTM	GRU
Гейти	4	2
Стани	є	тільки h_t
Параметри	~4×H(H+D)	~3×H(H+D)
Пам'ять	Довша	Коротша
Швидкість	Повільніше	Швидше
Якість	Кращ. довгі	Кращ. короткі
Рік	1997	2014

BPTT ТА ПРОБЛЕМИ НАВЧАННЯ RNN

$$\text{BPTT: } \partial L/\partial W_h = \Sigma_t \Sigma_{k \le t} \left(\Pi_{j=k+1}^{t} \partial h_j/\partial h_{j-1}\right) \cdot \partial h_k/\partial W_h$$

Vanishing Gradient

$$\|\partial h_t/\partial h_1\| = \|\Pi_t \partial h_t/\partial h_{t-1}\| \to 0$$

Добуток малих чисел (<1) → градієнт зникає. Мережа не навчається на далеких залежностях.

Рішення

Рішення: LSTM, GRU; Transformer; gradient clipping

Exploding Gradient

$$\|\partial h_t/\partial h_1\| \to \infty$$

Добуток великих значень → NaN у градієнтах. Параметри розходяться під час навчання.

Рішення: Gradient Clipping: $g \leftarrow g \cdot max\_norm/\|g\|$

Послідовне навчання (складність)

$$O(T \cdot H^2) \text{ — немає паралелізму}$$

RNN не можна навчати паралельно (залежність від попереднього кроку). Довго на довгих послідовн.

Рішення: Truncated BPTT; Transformer Architecture

ЗАСТОСУВАННЯ RNN / LSTM / GRU

Машинний переклад

Seq2Seq LSTM перекладає речення (Google Translate до 2017). Many-to-Many.

Генерація тексту/музики

CharRNN передбачає наступний символ. Шекспір, стиль Баха.

Часові ряди

Прогноз акцій, погоди, енергоспоживання. Many-to-One або Many-to-Many.

Аналіз тональності

Bi-LSTM: 'Мені сподобалось' → Positive. IMDB, Twitter.

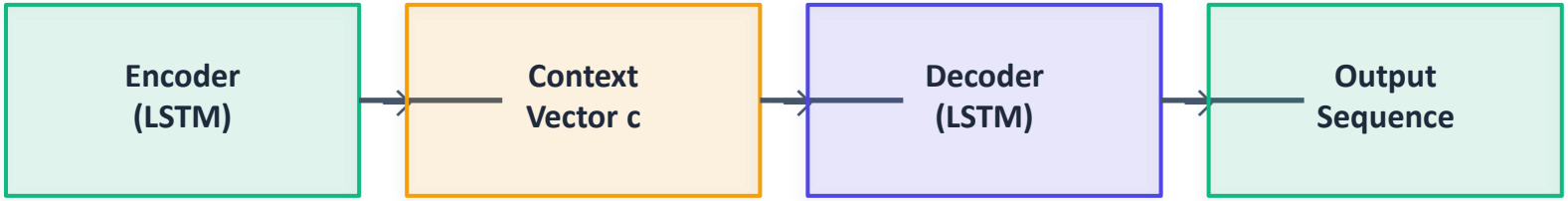
Розпізнавання мовлення

CTC+LSTM → акустична модель аудіо→текст (DeepSpeech, Whisper).

NER / POS tagging

Bi-LSTM-CRF: Named Entity Recognition — Особа, Організація, Місце.

SEQ2SEQ ТА МЕХАНІЗМ УВАГИ (ATTENTION)



Bahdanau Attention: $e_{ts}=score(s_t,h_s) \rightarrow \alpha_{ts}=softmax(e_{ts}) \rightarrow c_t=\Sigma_s\ \alpha_{ts}\cdot h_s$

Функції score (схожість запиту та ключа):

Dot-Product

$score(s,h) = s^T h$

Scaled Dot

$score = s^T h/\sqrt{d_k}$

Additive

$score = v^T\cdot\tanh(W_s s+W_h h)$

Cosine

$score = s^T h/(\|s\|\cdot\|h\|)$

Multi-Head: $MH(Q,K,V) = Concat(head_i)\cdot W^O, \quad head_i = Attention(QW_iQ,\ KW_iK,\ VW_iV)$

Self-Attention: $A = softmax(QK^T/\sqrt{d_k})\cdot V$ (Scaled Dot-Product)

TRANSFORMER: ATTENTION IS ALL YOU NEED (Vaswani 2017)

Self-Attention

Кожен токен зважує всі інші. Паралельне навчання без рекурентності.

$$A = \text{softmax}(QK^T/\sqrt{d_k})V$$

Positional Encoding

Додається до ембедінгів: sin/cos функції від позиції.

$$PE(pos,2i)=\sin(pos/10000^{(2i/d)})$$

Feed-Forward Block

FFN застосовується до кожного токену незалежно: 2 шари + ReLU.

$$FFN(x) = \text{ReLU}(xW_1+b_1)W_2+b_2$$

Add & Layer Norm

Residual + нормалізація: стабілізує навчання глибоких стеків.

$$\text{Output} = \text{LayerNorm}(x + \text{Sublayer}(x))$$

Покоління Transformer

2017	Transformer	65M
2018	BERT	340M
2018	GPT-1	117M
2019	GPT-2	1.5B
2020	GPT-3	175B
2021	DALL·E	12B
2022	ChatGPT	~175B
2023	GPT-4/Gemini	?T

ПІДСУМКИ ЛЕКЦІЇ

Нейрон та MLP

- net = $\Sigma w_i x_i + b$, вихід $y = f(net)$
- Активації: ReLU (default), sigmoid, tanh, GELU
- MLP: $a^l = f(W^l a^{l-1} + b^l)$ — Universal Approximation Theorem
- Backprop + Chain Rule + Adam optimizer

CNN (Згорткові)

- Згортка: $(I*K)[i,j] = \Sigma\Sigma I[i+m, j+n] \cdot K[m,n]$
- Conv→BN→ReLU→Pool→Dense пайплайн
- ResNet: skip connections F(x)+x — без vanish.grad
- Застосування: CV, медицина, OCR, сегментація

RNN / LSTM / GRU

- RNN: $h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$ — послідовності
- LSTM: 4 гейти + cell state c_t — довга пам'ять
- GRU: 2 гейти (z,r) — ефективний аналог LSTM
- Transformer: Self-Attention замінив рекуренцію