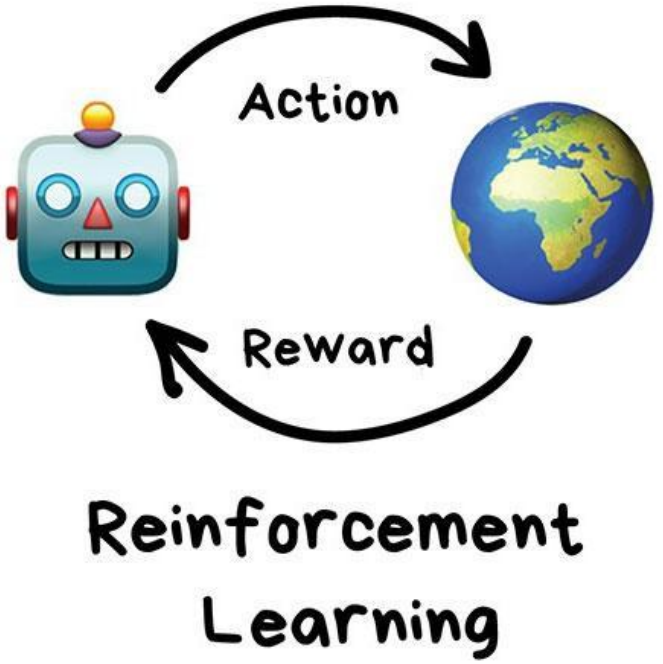
**Частина 2. Навчання з підкріпленням**

Сьогодні використовують для:

- Самокерованих автомобілів.
- Роботів-порохотягів.
- Ігор.
- Автоматичної торгівлі.
- Управління ресурсами підприємств.

Популярні алгоритми: Q-Learning, SARSA, DQN, A3C, Генетичний Алгоритм




Навчання з підкріпленням використовують там, де завданням є не аналіз даних, а виживання в реальному середовищі.

Середовищем може бути навіть відеогра, наприклад, роботи, які грають в Маріо. Середовищем може бути реальний світ. Як приклад - автопілот Тесли, який вчиться не збивати пішоходів, або роботи-порохотяги, головним завдання яких є налякати кота з максимальною ефективністю.

Знання про навколишній світ для такого робота можуть бути корисними, але чисто для довідки. Не важливо скільки даних він збере, в нього все одно не вийде передбачити всі ситуації. Тому його метою є **мінімізувати помилки, а не розрахувати всі ходи**. Робот вчиться виживати в просторі з максимальною вигодою: зібраними монетками в Маріо або часом поїздки в Теслі.

Ідеєю навчання з підкріпленням є виживання в середовищі, наприклад, помістити робота в реальне життя, штрафувати його за помилки і нагороджувати за правильні вчинки.

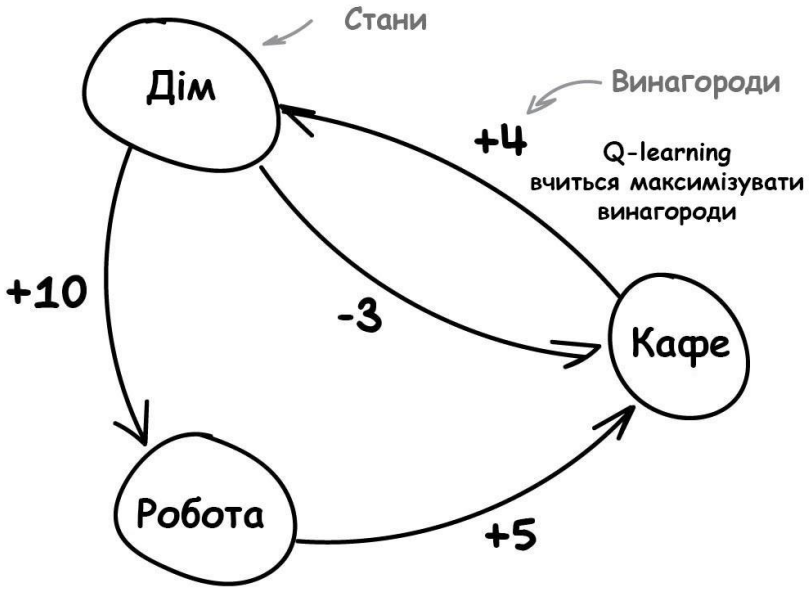
Розумні моделі роботів-порохотягів і самокеровані автомобілі навчаються саме так: для них створюють віртуальне місто (часто на основі карт справжніх міст), населяють випадковими пішоходами і відправляють вчитися нікого там не збивати. Коли робот починає добре себе почувати в штучному світі, його тестують на реальних вулицях.

Запам'ятовувати саме місто машині не потрібно - такий підхід називається Model-Free. Звичайно, тут є і класичний Model-Based, але в ньому машині довелося б запам'ятовувати модель всієї планети, всі можливі ситуації на всіх перехрестях світу. Таке просто не працює.

У навчанні з підкріпленням **машина не запам'ятовує кожен рух, а намагається узагальнити ситуації, щоб виходити з них з максимальною вигодою**.


Кілька років назад машина обіграла людину в Го, хоча незадовго до цього було доведено, що число комбінацій фізично неможливо прорахувати, адже воно перевищує кількість атомів у Всесвіті. Тобто, якщо в шахах машина реально прораховувала всі майбутні комбінації і перемагала, з Го так не можна. Тому, машина просто вибирала найкращий вихід з кожної ситуації і робила це досить точно, щоб обіграти людину.

Ця ідея лежить в основі алгоритму Q-learning і його похідних (SARSA і DQN). Буква Q в назві означає слово Quality, тобто робот вчиться поводитися найбільш якісно в будь-якій ситуації, а всі ситуації він запам'ятовує як простий марківський процес.




Рутинний **Марківський процес**

Машина проганяє мільйони симуляцій в середовищі, запам'ятовуючи всі сформовані ситуації і виходи з них, які принесли максимальну винагороду. Але як зрозуміти, коли склалася відома ситуація, а коли абсолютно нова? Ось самокерований автомобіль стоїть біля перехрестя і загоряється зелений - значить можна їхати? А якщо справа мчить швидка допомога з мигалками?

Відповідь - ніяк, дослідники постійно цим займаються, винаходячи свої підходи. Одні прописують всі ситуації руками, що дозволяє їм обробляти виняткові випадки. Інші віддають цю роботу до нейромереж, нехай самі все знайдуть. Так замість Q-learning'а з'являється Deep Q-Network (DQN).

Reinforcement Learning для пересічного користувача виглядає як справжній інтелект, оскільки машина сама приймає рішення в реальних ситуаціях! Популярні генетичні алгоритми теж відносяться до навчання з підкріпленням.

***Частина 3. Ансамблеві методи***

Сьогодні використовують для:

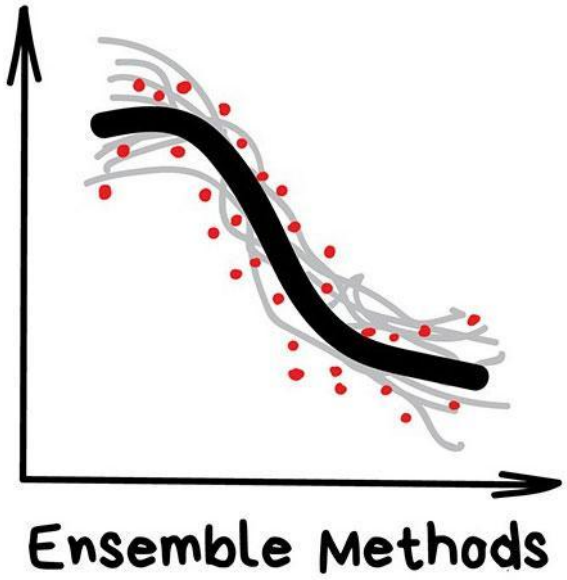
Всього, де підходять класичні алгоритми (але працюють точніше).

Пошукові системи.

Комп'ютерний зір.

Розпізнавання об'єктів.

Популярні алгоритми: Random Forest, Gradient Boosting.




Ансамблі та нейромережі – головні підходи на шляху до неминучого прогресу. Сьогодні вони надають найточніші результати і використовуються всіма великими компаніями.

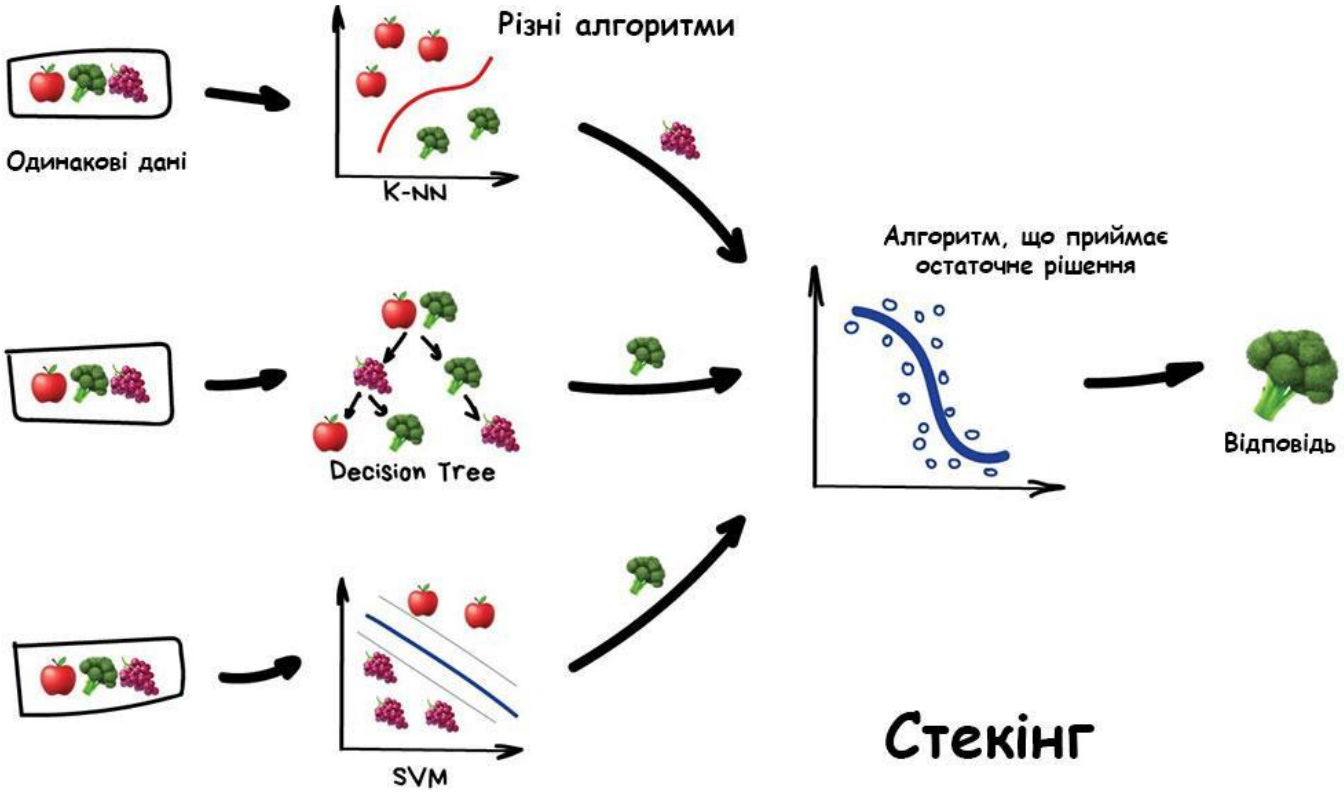
При всій їх ефективності, ідея є доволі простою. Виявляється, якщо взяти декілька не дуже ефективних методів навчання і навчити виправляти помилки один одного, якість такої системи буде значно вищою, ніж кожного з методів окремо.

Причому, навіть краще, коли обрані алгоритми є максимально нестабільними і сильно залежними від вхідних даних. Тому, частіше беруть Регресію і Дерева Рішень, яким достатньо однієї сильної аномалії в даних, щоб розбалансувалася вся модель. А ось Байес і K-NN не беруть ніколи - вони хоч і мають прості обчислення, але є дуже стабільними.

Ансамбль можна зібрати як завгодно, але є три перевірених способи робити ансамблі.

**Стекінг (Stacking)**

Стекінг - один з найпопулярніших способів використання декількох алгоритмів (ансамбля) для вирішення однієї задачі машинного навчання. Навчаємо кілька різних алгоритмів і передаємо їх результати на вхід останньому, який приймає остаточне рішення, зазвичай беруть регресію.

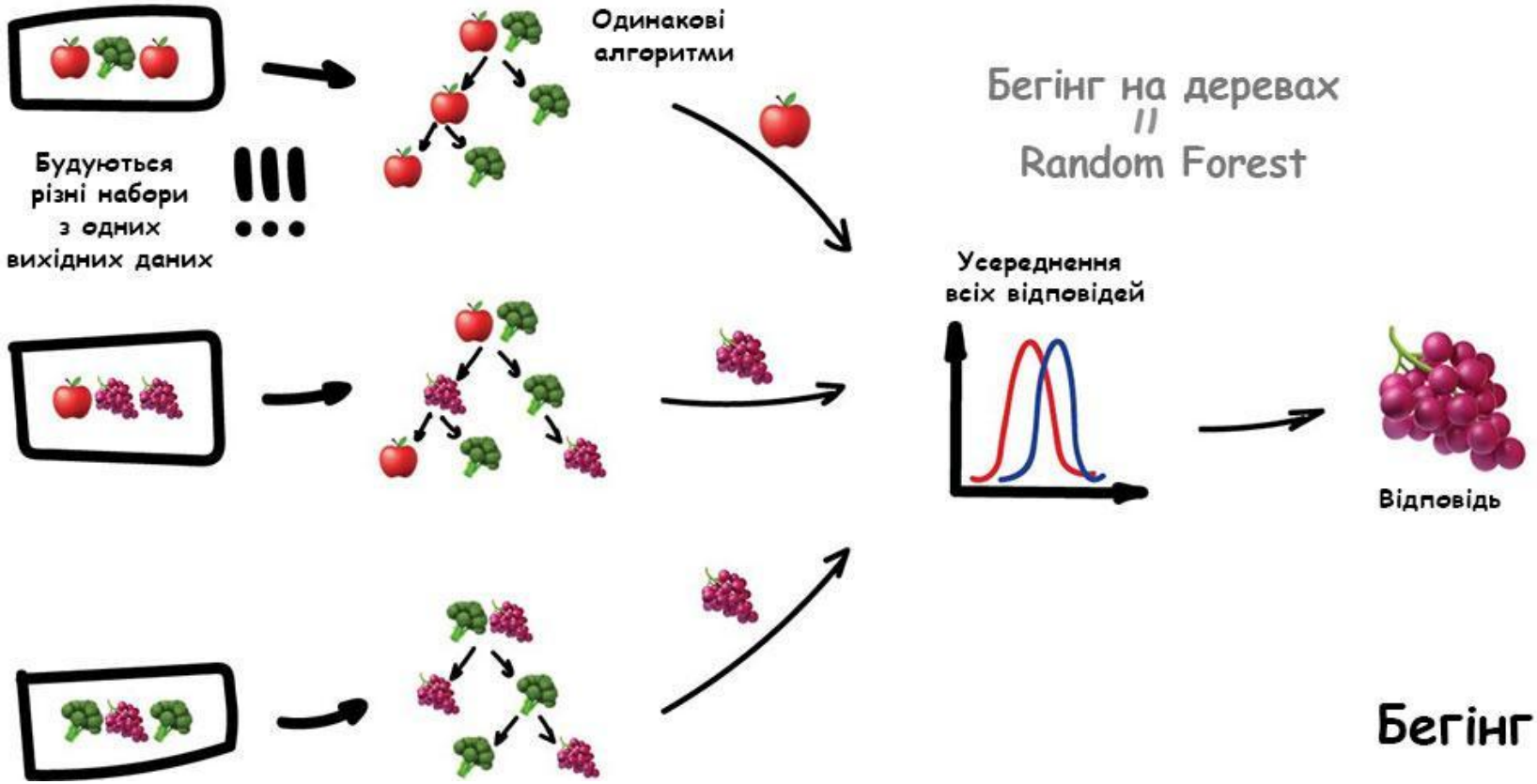


Ключове слово - різних алгоритмів, адже один і той же алгоритм, навчений на одних і тих же даних не має сенсу. Стекінг на практиці застосовується рідко, оскільки наступні два інших методу є точнішими.

**Беггінг (Bootstrap AGGregatING)**

Беггінг - технологія класифікації, де елементарні класифікатори навчаються і працюють паралельно (незалежно один від одного). Ідея полягає в тому, що класифікатори не виправляють помилки один одного, а компенсують їх при голосуванні. Базові класифікатори повинні бути незалежними, це можуть бути класифікатори, що засновані на різних групах методів або ж навчені на незалежних наборах даних.

Один алгоритм навчається багато разів на випадкових вибірках з вихідних даних. Наприкінці відповіді усереднюються. Дані в випадкових вибірках можуть повторюватися. Тобто, з набору 1-2-3 можна робити вибірки 2-2-3, 1-2-2, 3-1-2 і так далі. На них навчається один і той же алгоритм кілька разів, а в кінці відповідь обчислюється простим голосуванням.



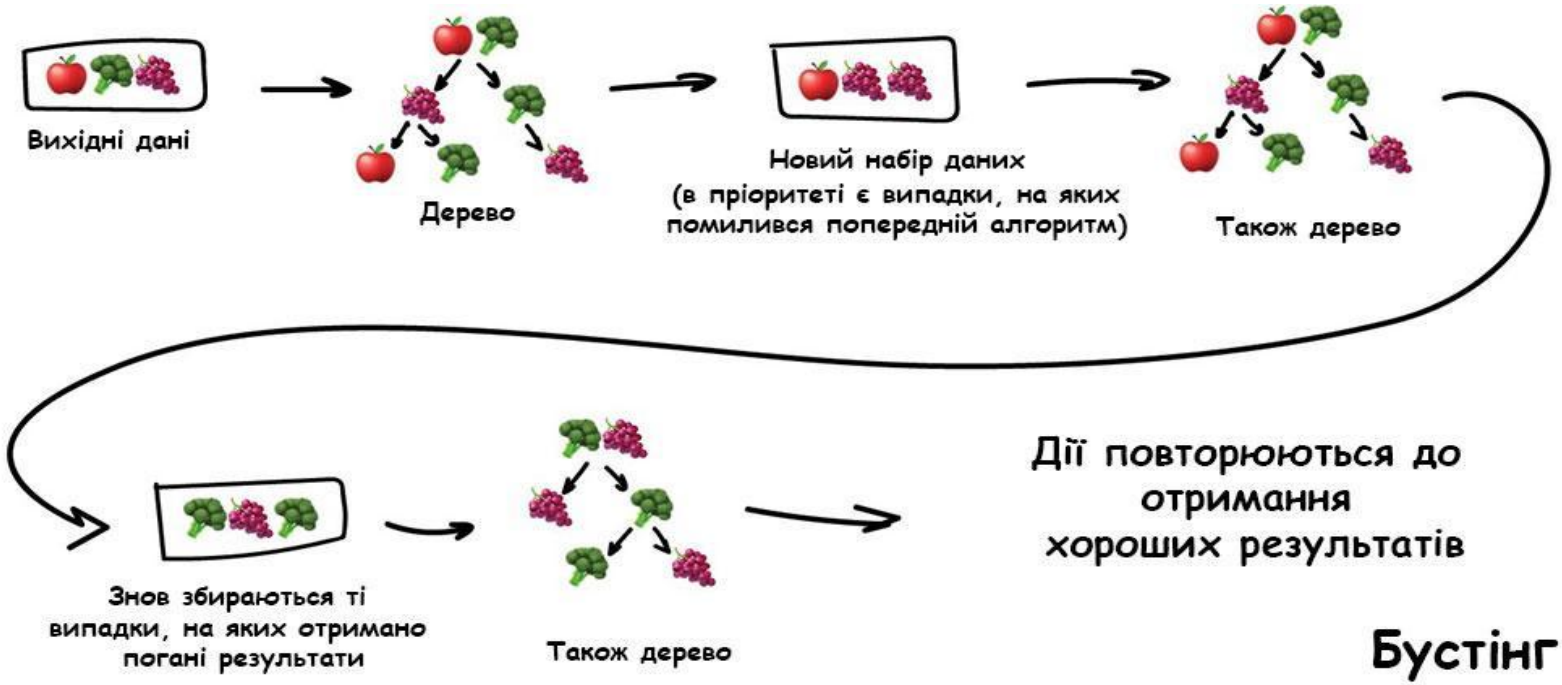

Найпопулярніший приклад Бегінгу - алгоритм Random Forest, бегінг на деревах, який і зображено на рисунку. В камері на телефоні жовтими прямокутниками окреслюються обличчя людей в кадрі - швидше за все це бегінг. Нейромережа буде занадто повільною в реальному часі, а бегінг є ідеальним, адже він може обчислювати свої дерева паралельно на всіх процесорах відеокарти.


Здатність паралелелізму надає бегінгу перевагу над наступним методом, який працює точніше, але тільки в один потік.

**Бустінг (Boosting)**

Бустінг - це процедура послідовної побудови композиції алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів. Алгоритми виконуються послідовно, кожен наступний приділяє особливу увагу тим випадкам, на яких помилився попередній.

Як в бегінгу, здійснюються вибірки з вихідних даних, але не зовсім випадково. В кожну нову вибірку береться частина тих даних, на яких попередній алгоритм відпрацював неправильно. Тобто як би новий алгоритм довчається на помилках попереднього.




Значним плюсом є надвисока точність класифікації, але мінусом є не паралельний процес, хоча цей алгоритм працює швидше нейромереж.

MatrixNet (Матрикснет) - спеціальний алгоритм машинного навчання, що заснований на бустінгу. Завдяки використанню Матрикснет пошукові машини Яндексу стали враховувати набагато більшу кількість критеріїв ранжирування, ніж до введення цього алгоритму.

**Частина 4. Нейромережі і глибоке навчання**

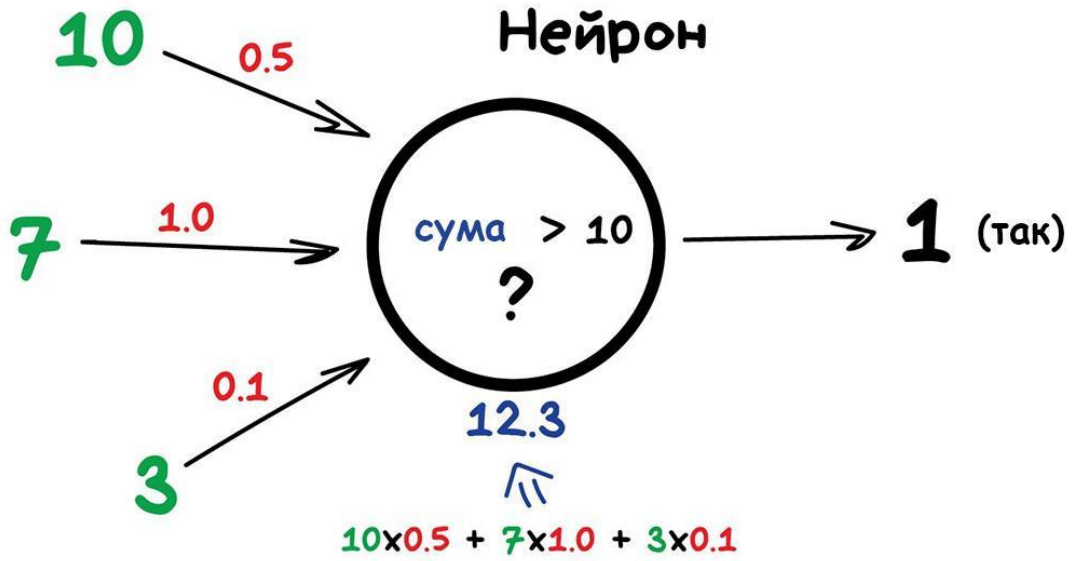
Сьогодні використовують для:

Замість всіх перелічених вище алгоритмів.
Визначення об'єктів на фото і відео.
Розпізнавання і синтез мови.
Обробка зображень, перенесення стилю.
Машинний переклад.

Популярні архітектури: Перцептрон, Згорткові Мережі (CNN), Рекурентні Мережі (RNN), Автокодувальники.

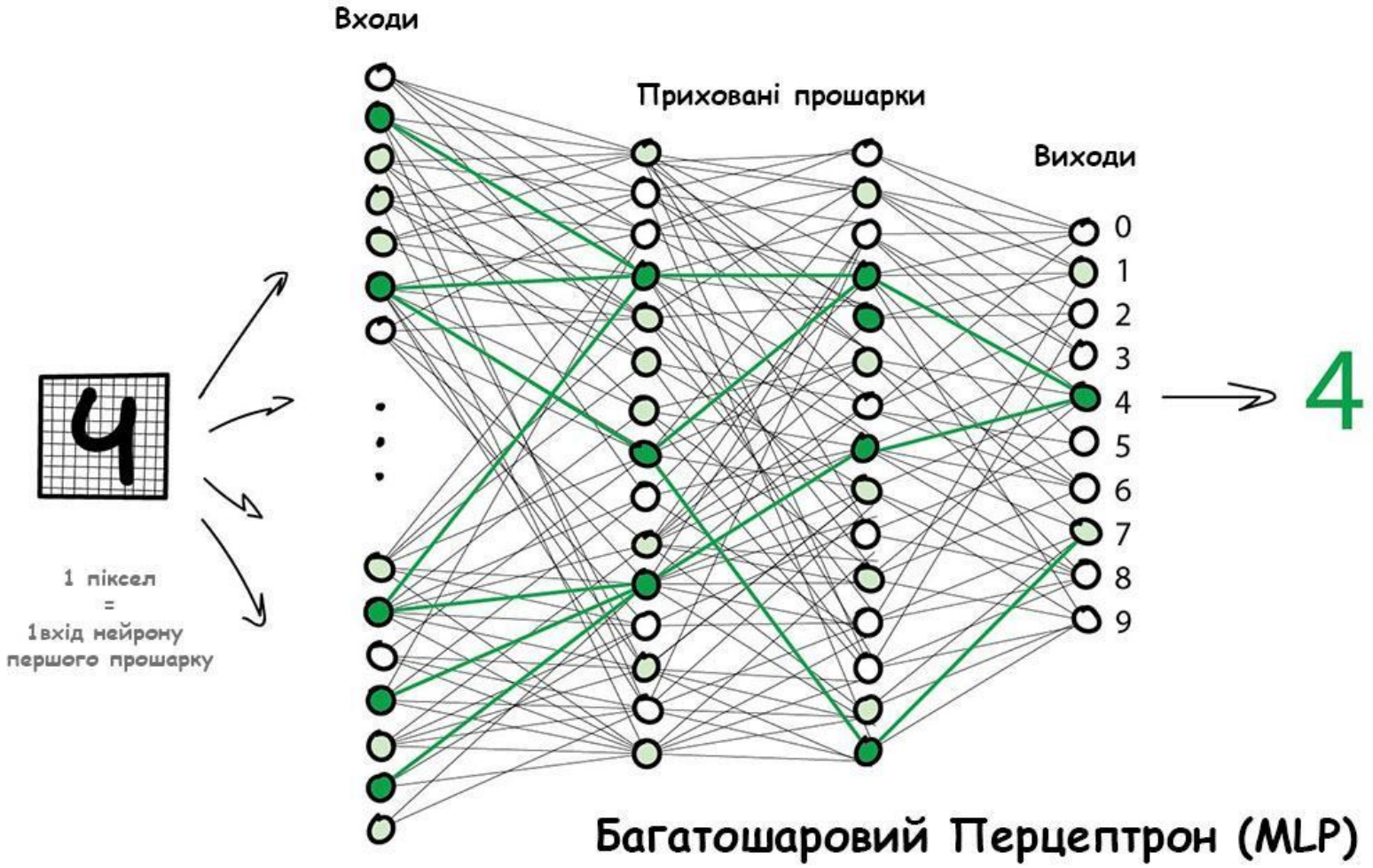
Будь-яка нейромережа - це набір нейронів і зв'язків між ними. Нейрон найкраще представляти просто як функцію з багатьма входами і одним виходом. Завданням нейрону є взяти числа зі своїх входів, виконати над ними функцію і віддати результат на вихід. Простий приклад корисного нейрона: підсумувати всі цифри зі входів, і якщо їх сума більше за N - видати на вихід одиницю, інакше - нуль.

Зв'язки - це канали, через які нейрони надсилають один одному сигнали. У кожного зв'язка є своя вага - її єдиний параметр, який можна умовно представити як вагу зв'язку. Якщо через зв'язок з вагою 0.5 проходить число 10, воно перетворюється в 5. Сам нейрон не розбирається, що до нього прийшло і підсумовує все підряд - ось ваги і потрібні, щоб керувати на які входи нейрон повинен реагувати, а на які ні.




Нейрони розподілені по прошарках. Всередині одного прошарку нейрони не пов'язані, але з'єднані з нейронами наступного і попереднього прошарків. Дані в такій мережі йдуть строго в одному напрямку - від входів першого шару до виходів останнього. Мережа, де кілька шарів і нейрони між ними всі пов'язані, називається перцептроном (MLP) і вважається найпростішою архітектурою для початківців.

Якщо створити достатню кількість прошарків і правильно розставити ваги в такій мережі, виходить наступне - подавши на вхід, наприклад, зображення написаної від руки цифри 4, чорні піксели активують пов'язані з ними нейрони, ті активують наступні прошарки, і так далі і далі, поки в результаті не активується вихід, який відповідає за четвірку. Результат досягнутий.




Після побудови мережі, слід правильно розставити ваги, щоб нейрони реагували на потрібні сигнали. Даними для мережі є приклади «входів» і правильних «виходів». Для нейромережі показують зображення цифри 4 і змушують її підлаштовувати ваги так, щоб на виході при такому вході завжди спалахувала четвірка.

Спочатку всі ваги розставляються випадково, при пред'явленні цифри 4, вона видає випадкову відповідь (не налаштовані ваги), далі відбувається порівняння, наскільки результат відрізняється від потрібного. Потім обчислення по мережі відбуваються в зворотному напрямку, від виходів до входів, і корегується вага зв'язків кожного нейрона.

Через значну кількість циклів «прогнали-перевірили-скорегували», ваги в мережі налаштовуються так, як потрібно. Науково цей підхід називається BackPropagation або «Метод зворотного поширення похибки».

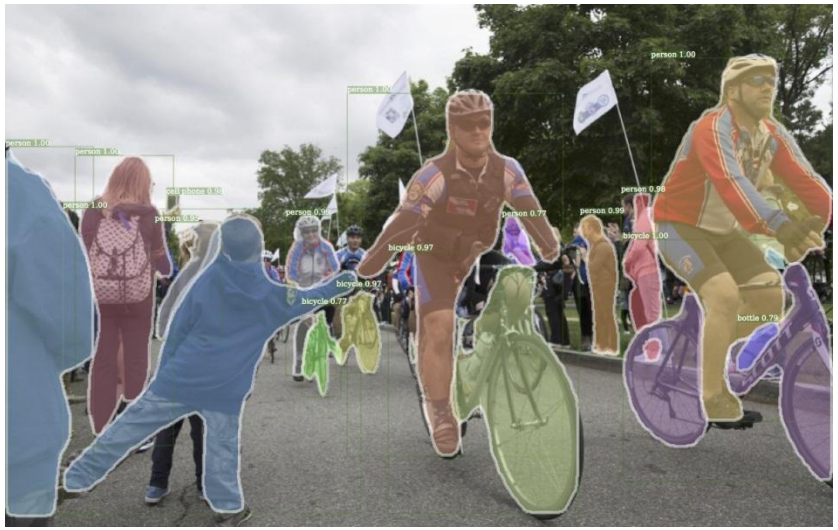
Така універсальність нейромереж зробила їх сильно популярними. Дослідники доводили, що відтворили архітектуру людського мозку, потрібно просто зібрати багато прошарків і навчити їх на будь-яких даних. Але історично за ці роки відбувалося кілька етапів збільшення та зменшення інтересу до нейронних мереж.

У 2012 році в конкурсі ImageNet представлено згорткову нейромережу, через що в світі раптово згадали про методи глибокого навчання, які описані ще в 90-х роках.

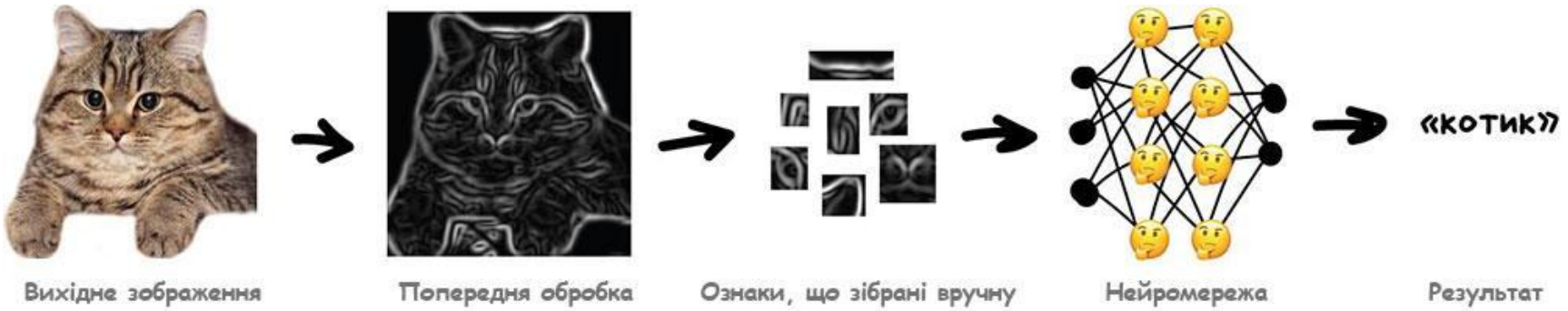
Відмінність глибокого навчання від класичних нейромереж полягала в нових методах навчання, які справлялися з великими розмірами мереж. Однак сьогодні лише теоретики розділяють, яке навчання можна вважати глибоким, а яке не надто. Практики, використовують популярні «глибокі» бібліотеки типу Keras, TensorFlow і PyTorch навіть коли потрібно зібрати міні-мережу на п'ять прошарків, оскільки вони є зручнішими від того, що було

**Згорткові Нейромережі (CNN)**

Згорткові мережі зараз на піку популярності. Вони використовуються для пошуку об'єктів на фото і відео, розпізнавання осіб, перенесення стилю, генерації і домальовування зображень, створення ефектів уповільненого часу і покращення якості фотографій. Сьогодні CNN застосовують всюди, де є картинки або відео. Навіть в айфоні є кілька таких мереж, що розпізнають об'єкти на них.




Проблема з зображеннями завжди була в тому, що незрозуміло, як виділяти на них ознаки. Текст можна розділити за реченнями, взяти властивості слів зі словників. Картинки же доводилося розмічати руками, пояснюючи машині, де у котика на фотографії вуха, а де хвіст. Такий підхід навіть назвали «handcrafting ознак» і раніше все так і робили.




Проблем у ручного крафтінга багато.

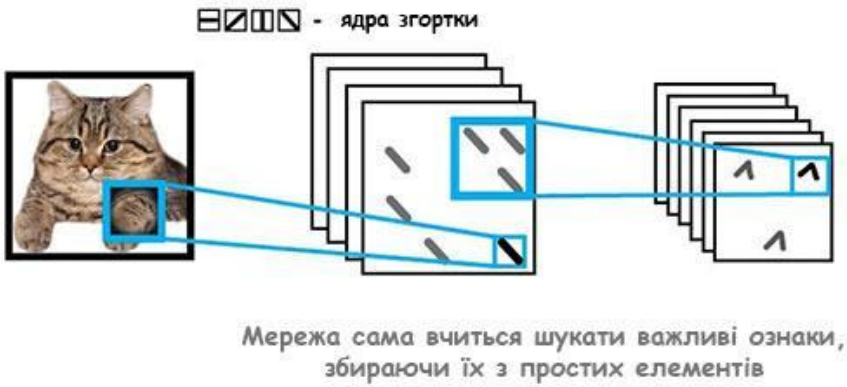
По-перше, якщо котик на фотографії притиснув вушка або відвернувся - нейромережа нічого не побачить.

По-друге, важко сформувати хоча б десять характерних ознак, що відрізняють котиків від інших тварин. Однак людина, навіть краєм ока може розрізнити хто є котиком, а хто собакою. Тому що людина не дивиться тільки на форму вух і кількість лап - вона оцінює об'єкт за множиною різних ознак, про які сама навіть не замислюється. А значить, не розуміє і не може пояснити машині.

Виходить, машині потрібно самій вчитися шукати ці ознаки, складаючи з якихось базових ліній. Підхід буде таким: для початку зображення розділяється на блоки 8x8 пікселів і вибирається, яка лінія домінує в кожному - горизонтальна [-], вертикальна [|] або одна з діагональних [\] [/].

На виході отримується кілька масивів ліній, які є простими ознаками наявного образу об'єкта на зображенні. Це теж є картинки, просто з ліній. Значить можна знов вибрати блок 8x8 і подивитися, як ці лінії поєднуються одна з одною. І так далі.

Така операція називається згорткою, звідки і пішла назва методу. Згортку можна уявити як прошарок нейромережі, адже нейрон – це абсолютно будь-яка функція.





Коли через нейромережу проходить багато фотографій котів, вона автоматично розставляє великі ваги тим сполученням з ліній, які побачила найчастіше. Причому неважливо, це пряма лінія спини або складний геометричний об'єкт типу мордочки - щось обов'язково буде яскраво активуватися.

На виході ставиться простий перцептрон, який буде дивитися, які поєднання активувалися і визначати, для кого вони є більш характерними – для котів чи собак.

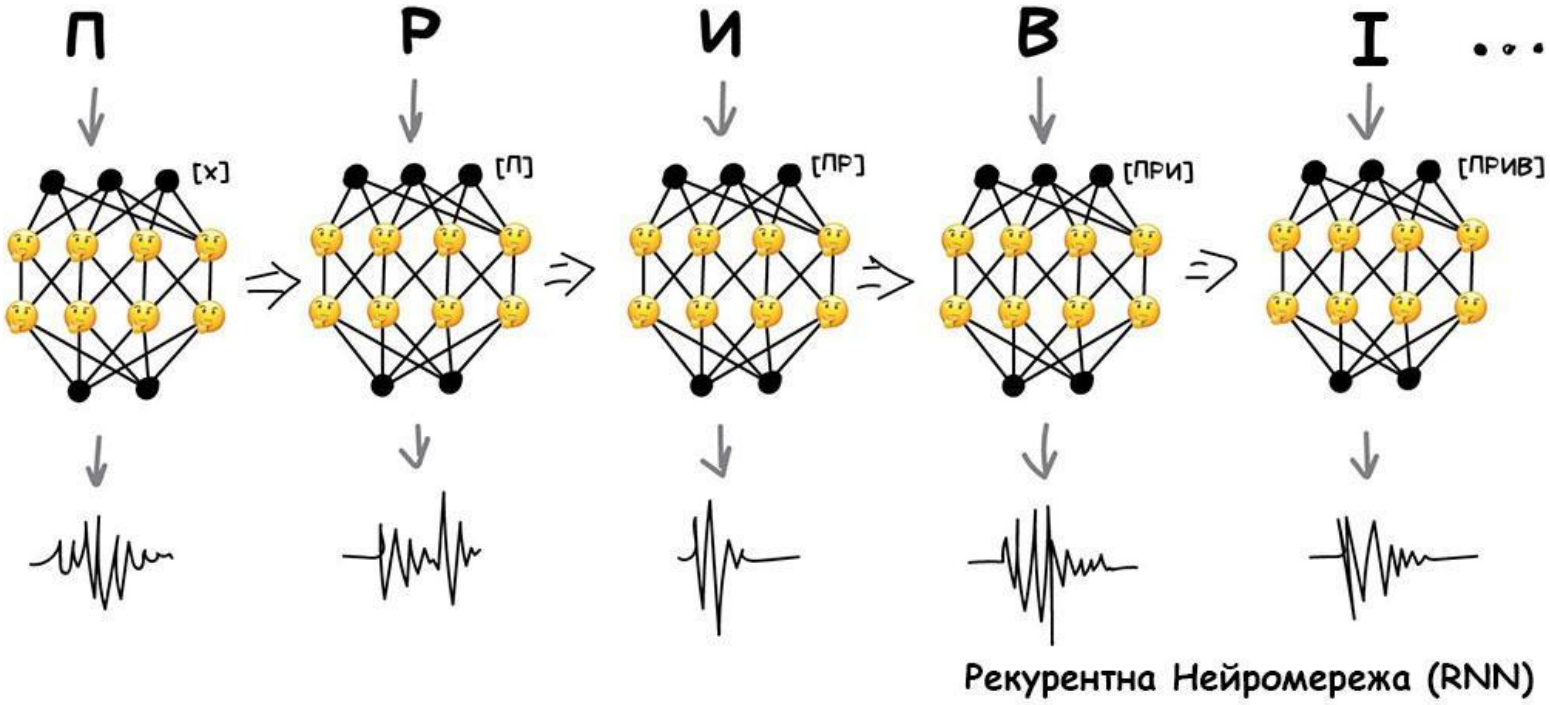
Згорткова нейромережа сама знаходить характерні ознаки об'єктів, складає карти ознак і навчається визначати що завгодно.

**Рекурентні Нейромережі**
**(RNN)**

Наступна за популярністю архітектура на сьогоднішній день. Завдяки рекурентним мереж є такі корисні речі, як машинний переклад текстів і комп'ютерний синтез мови. На них вирішують всі завдання, пов'язані з послідовностями - голосові, текстові або музичні.

Досить легко навчити мережу вимовляти окремі слова або букви. Береться множина аудіо файлів, розмічених на слова і навчається за вхідним словом видавати послідовність сигналів, схожих на його вимову. Здійснюється порівняння з оригіналом від диктора і намагання максимально наблизитися до ідеалу. Для таких дій підійде навіть перцептрон.

Втім, перцептрон не запам'ятовує, що він генерував раніше, для нього кожен запуск як в перший раз. З'явилася ідея додати до кожного нейрона пам'ять. Так були винайдені рекурентні мережі, в яких кожен нейрон запам'ятовував всі свої попередні відповіді і при наступному запуску використовував їх як додатковий вхід. Тобто, нейрон міг сказати самому собі в майбутньому - наступний звук повинен звучати вище, голосна була довшою (дуже спрощений приклад).




Рекурентна Нейромережа (RNN)

Була лише одна проблема - коли кожен нейрон запам'ятовував всі минулі результати, в мережі утворювалася надвелика кількість входів, що навчити таку кількість зв'язків стало нереально. Коли нейромережа не вміє забувати - її не можна навчити.

Спочатку проблему вирішили просто – відімкнули для кожного нейрона пам'ять. Але потім придумали як для цієї «пам'яті» використовувати спеціальні комірки, що схожі на пам'ять комп'ютера або регістри процесора. Кожна комірка дозволяла записати в себе числа, прочитати або скинути - їх назвали комірками довгої і короткотермінової пам'яті (LSTM).

Коли нейрону потрібно було поставити собі нагадування на майбутнє - він заносив це в комірку, коли навпаки вся історія ставала непотрібною (наприклад, речення закінчилося) - комірки скидалися, залишаючи тільки «довготермінові» зв'язки, як в класичному перцептроні. Іншими словами, мережа навчалася не тільки встановлювати поточні зв'язки, а й ставити нагадування.