МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ ТА МОДЕЛЮВАННЯ СКЛАДНИХ СИСТЕМ

КВАЛИФИКАЦІЙНА РОБОТА

Тема роботи:

«Моделювання систем комп'ютерного відтворення та розпізнавання цифрових зображень»

Завідувач кафедри — д.ф.-м.н., проф. Лисенко О. В.

Керівник роботи — к. ф.-м. н. Сушко Т. С.

Виконавець — студент факультета електроніки та інформаційних технологій
гр. ПМ-41
Черногор Максим Станіславович

Затверджено на засіданні кафедри прикладної математики та моделювання складних систем від "25" жовтня 2017 року., протокол №3

Суми, 2018

Реферат

Кваліфікаційна робота присвячена розробці та аналізу моделей з використанням штучних нейронних мережі для розв'язку задачі класифікації рукописних цифр. У роботі проведено огляд наявних методів та алгоритмів у тематиці комп'ютерного зору, розпізнавання та класифікації.

Мета - розробка та аналіз моделей з використанням штучних нейронних мережі для розв'язання задачі класифікації рукописних цифр.

Побудована та оптимізована власна модель, в основі якої стоїть композиція загорткової нейронної мережі та повнозв'язного класифікатора. Також побудовані інші моделі, такі як перцептрони, з якими були проведені експерименти, та досліджено вплив початкових значень гіперпараметрів на результат класифікації, попередньої обробки даних та застосовані різні підходи до зміни елементів архітектури.

Автоматичне розпізнавання і класифікація рукописних цифр має прикладне значення при автоматизованому сортуванні пошти за поштовим кодом, автоматизоване читання чеків, податкових декларацій, а також введення інформації даних у бази даних комп'ютера. Результати даної роботи можуть бути застосовані для вирішення більш глобальної задачі детектування і розпізнавання рукописного тексту на фото та відеопотоці.

Загальний обсяг роботи 42 сторінки, 15 малюнків, 1 таблиця, 2 додатки, 13 бібліографічних найменувань.

КЛЮЧОВІ СЛОВА: РОЗПІЗНАВАННЯ, ЗГОРТКОВА НЕЙРОННА МЕРЕЖА, ПЕРЦЕПТРОН, ПОВНОЗВ'ЯЗНА НЕЙРОННА МЕРЕЖА, РУКОПИСНІ ЦИФРИ, КЛАСИФІКАЦІЯ.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

Датасет – набір даних, вибірка

Tensorflow – назва бібліотеки машинного навчання

ВСТУП

Програми розпізнавання тексту наразі користуються великою популярністю. Кожного дня на пошту поступає дуже велика кількість конвертів з рукописними індексами та адресами, до банків листи з номерами рахунків та депозитів, написаних від руки. Також досі існує велика кількість стародавніх рукописів, які ще не встигли оцифрувати. І саме тому системи автоматичного розпізнавання тексту є дуже актуальними на цей час.

Дана робота присвячена побудові оптимальної системи для розв’язку задачі розпізнавання та класифікації інформації, записаної від руки. А конкретно, вирішується задача класифікації рукописних цифр.

Прийнятним результатом буде досягнення точності 98%, що можна порівняти з людською здатністю розпізнавання. Але зусилля при вирішенні даної задачі спрямовані на те, щоб досягти точності, близької до 100%. Це важливо, адже при некоректній класифікації, наприклад, номеру рахунку в банку, чи адреси на поштовому листі, гроші чи лист відправляться не за адресою, що є проблемою і для бізнесу, і для клієнтів.

Розв’язати з високою точністю дану задачу дозволяють штучні глибокі нейронні мережі. Але під кожну конкретну задачу потрібно підібрати оптимальний тип архітектури нейромережі, дослідити та виконати передобробку датасета, дослідити вплив гіперпараметрів на точність класифікації. Що і було здійснено під час виконання кваліфікаційної роботи.

1 ОГЛЯД КЛАСИЧНИХ МЕТОДІВ В ГАЛУЗІ КОМП'ЮТЕРНОГО ЗОРУ, МАШИННОГО НАВЧАННЯ ТА РОЗПІЗНАВАННЯ ОБРАЗІВ

1.1 Машинне навчання

Машинне навчання - це великий підрозділ штучного інтелекту, який вивчає методи побудови алгоритмів, здатних навчатися, та знаходиться на перетині таких класичних математичних дисциплін як дискретний аналіз, математична статистика, теорія ймовірностей, чисельні методи оптимізації. Також ця галузь має власну специфіку, що тісно пов'язана з проблемами перенавчання та обчислювальної ефективності.

Розрізняють два типи навчання. Індуктивне навчання (Навчання по прецедентах), яке засновано на виявленні закономірностей в емпіричних даних. Та дедуктивне навчання, яке передбачає формалізацію знань експертів і їх перенесення в комп'ютер у вигляді бази даних знань. Дедуктивне навчання прийнято відносити до області експертних систем і саме тому терміни машинне навчання і навчання по прецедентах можна вважати синонімами.

Розглянемо основні засоби навчання:

- Навчання з учителем - для кожного прецеденту задається пара «ситуація та необхідний розв'язок»
- Навчання без вчителя - для кожного випадку задається тільки «ситуація», а система повинна згрупувати об'єкти в кластери, використовуючи дані попарної схожості об'єктів, і / або знизити розмірність даних
- Навчання з підкріпленням - для кожного прецеденту є пара «ситуація, прийняте рішення»

Також існують такі типи навчання, як активне навчання, навчання з частковим залученням вчителя, трансдуктивне навчання та інші. Усі перелічені типи актуальні на даний час та як кожен тип, так і їх комбінація можуть бути застосовані для конкретних вузькоспеціалізованих задач.

Прикладом можуть служити задача класифікації, яка, зазвичай, виконується за допомогою навчання з учителем, задача кластеризації, яка виконується за допомогою навчання без учителя та задача регресії, яка виконується за допомогою навчання з учителем на етапі тестування.

Основними сферами застосування є розпізнавання рукописного тексту, розпізнавання мови, розпізнавання жестів, розпізнавання образів, технічна та медична діагностика, прогнозування часових рядів та біоінформатика. [1]

1.2 Комп'ютерний зір та розпізнавання образів на зображенні

Комп'ютерний зір - це науковий напрямок в області штучного інтелекту і пов'язаних з ним технологій отримання зображень об'єктів реального світу, їх обробки і використання отриманих даних для вирішення різного роду прикладних задач без участі (повної або часткової) людини.

Однією з найважливіших частин цього напрямку є задачі саме розпізнавання та класифікація образів. [2]

Класичні методи розпізнавання образів можна умовно розділити на такі групи:

- Методи фільтрації
- Методи обробки та аналізу
- Методи навчання

1.3 Фільтрація зображення

З широким впровадженням цифрових систем зв'язку, збільшується актуальність вирішення завдань відновлення зображень, отриманих за допомогою фото та відеокамер, з метою фільтрації зображень. На практиці часто зустрічаються зображення, пошкоджені шумом, який з'являється на етапах формування та передачі його по каналам зв'язку.

Задачею обробки зображення може бути поліпшення (відновлення та реставрація) зображення по якомусь певному критерію, так і спеціальне перетворення, яке кардинально змінює зображення. В останньому випадку обробка зображень може бути проміжним етапом для подальшого розпізнавання зображення (наприклад, для виділення контуру об'єкта).
Розглянемо найпоширеніші методи фільтрації.

1.3.1 Бінаризація по порогу, вибір області гістограми

Для RGB зображення і зображення в градаціях сірого порогом є значення кольору. Вибір порога, по якому відбувається бінаризація, багато в чому визначає процес бінаризації. Зазвичай бінаризація здійснюється за допомогою адаптивного алгоритму, вибираючого поріг. Таким алгоритмом можуть бути:

- Метод Янні

Метод полягає в знаходження максимального значення амплітуди сірого g_{max} і мінімального значення амплітуди сірого g_{min}. Потім вираховується середня кількість пікселів, що потрапили в діапазон від мінімального до середини. Так вираховується оптимальний поріг Янні: $T_{opt} = (g_{max} - g_{\min} \sum_{g=g_{mid}}^{g*mid} p(g))$

- Метод Середнього

Метод полягає в знаходженні мінімального g_{min} і максимального значення g_{max} амплітуди сірого і знаходження середнього значення між ними: $T_{opt} = (g_{max} - g_{\min})/2$

1.3.2 Класична фільтрація: Фур'є

Перетворення Фур'є майже не використовується при обробці зображень в чистому вигляді, оскільки для аналізу зображень одновимірного перетворення зазвичай не вистачає і виникає необхідність використання куди більш ресурсоємного двовимірного перетворення. Цей метод застосовується тільки в разі якщо необхідний аналіз спектра, оскільки використання згортки цікавить області з уже готовим фільтром виявляється швидше і простіше в реалізації. Один з небагатьох вийнятків, коли використовується одновимірне перетворення Фур'є, - компресія зображень.

1.3.3 Вейвлет-перетворення

Для згортки з сигналом (областю зображення) використовується характеристична функція, вейвлет, що визначається як вейвлет-перетворювання. Вейвлети - це сімейства функцій, локальних за часом і по частоті, в яких всі функції виходять з однієї з її допомогою зрушень і розтягувань по осі часу. Існує набір класичних функцій, використовуваних в вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морлі, вейвлет MHat та інші.

На практиці вейвлет-аналізом називається пошук довільного патерну на зображенні за допомогою згортки зображення з моделлю цього патерну. Класичні вейвлети використовуються для стиснення або класифікації зображень.

Загалом, вейвлет-перетворення може бути виражено наступним рівнянням: $F(a,b) = \int_{-\infty}^{\infty} f(x)\psi^*_{(a,b)}(x)dx$, де * - символ комплексної спряженості і функція ψ - деяка функція.

1.3.4 Обчислення показників кореляції

Обчислення кореляції, що лежить в основі вейвлет-перетворення, є незамінним інструментом в системах комп'ютерного зору.

Обчислення кореляції використовується в своєму природному вигляді, наприклад, для знаходження зрушень або оптичних потоків (Кореляція відеопотоку). На основі різницевого корелятора реалізується найпростіший детектор зсуву.

1.3.5 Фільтрації контурів

Окремий клас фільтрів - фільтрація границь і контурів. Контури дуже корисні, якщо ми хочемо перейти від роботи з зображенням до роботи з об'єктами на цьому зображенні. Коли об'єкт досить складний, але добре виділяється, то часто єдиним способом роботи з ним є виділення його контурів. Існує цілий ряд алгоритмів, які вирішують задачу фільтрації контурів:

- алгоритм Кенні
- алгоритм Собеля
- алгоритм Лапласа
- алгоритм Прюітт
- алгоритм Робертса

1.4 Обробка та аналіз

Фільтрація дає набір придатних для обробки даних. Але дуже часто можна просто взяти і використовувати ці дані без їх передобробки. У цьому розділі ми розглянемо класичні методи, які дозволяють перейти від зображення до властивостей об'єктів, або до самих об'єктів.

1.4.1 Морфологія

Математична морфологія — наука, яка вивчає методи та алгоритми аналізу і обробки геометричних структур, заснована на теорії множин,

топології і випадкових функцій. Застосовується при обробці цифрових зображень, але також може бути застосована до графів, полігональної сітки, стереометрії і багатьох інших просторових структур. [3]

Основними операціями математичної морфології є нарощування, ерозія, замикання і розмикання. У цих назвах відображена суть операцій: нарощування збільшує область зображення, а ерозія робить її менше, операція замикання дозволяє замкнути внутрішні отвори області та усунути затоки вздовж кордону області, операція розмикання допомагає позбутися від маленьких фрагментів, які виступають назовні області поблизу її кордону. [4]

1.4.2 Контурний аналіз

Контурний аналіз дозволяє описувати, зберігати порівнювати і знаходити об'єкти, що знаходяться в формі контурів.

При застосуванні цього методу передбачається, що контур містить необхідну інформацію про форму об'єкта. Внутрішні точки об'єкта при цьому не беруться до уваги. Це обмежує область застосування алгоритмів контурного аналізу, але в задачах, коли для розв'язку цього достатньо, розгляд лише контурів дозволяє перейти від двомірного простору образу до простору контурів і, отже, зменшується обчислювальна і алгоритмічна складність. Контурний аналіз дає можливість ефективно вирішувати головні завдання розпізнавання шаблонів - перенесення, поворот і масштабування зображення об'єкта. Методи контурного аналізу інваріантні щодо таких перетворень.

1.4.3 Особливі точки

Особливі точки є унікальними характеристиками об'єкта, які дозволяють порівнювати об'єкт сам з собою або зі схожими класами об'єктів. Існує кілька десятків способів дозволяють виділити такі точки. Деякі способи виділяють особливі точки в сусідніх кадрах, деякі через великий проміжок

часу і при зміні освітлення, деякі дозволяють знайти особливі точки, які залишаються такими навіть при поворотах об'єкта. Розглянемо три класи особливих точок:

Перший клас. Особливі точки, які є стабільними протягом секунд. Такі точки служать для того, щоб вести об'єкт між сусідніми кадрами відео, або для відомості зображення з сусідніх камер. До таких точок можна віднести локальні максимуми зображення, кути на зображенні, точки в яких досягається максимуми дисперсії, певні градієнти та інше.

Другий клас. Особливі точки, які є стабільними при зміні освітлення і невеликих рухах об'єкта. Такі точки служать в першу чергу для навчання і подальшої класифікації типів об'єктів. Деякі з раніше згаданих вейвлетів можуть є базою для таких точок. Наприклад, примітиви Хаара, пошук відблисків, пошук інших специфічних функцій. До таких точок відносяться точки, знайдені методом гістограм спрямованих градієнтів (HOG).

Третій клас. SURF і SIFT. Ці методи дозволяють знаходити особливі точки навіть при повороті зображення. Розрахунок таких точок здійснюється значно довше в порівнянні з іншими методами. [5]

1.5 Навчання

В задачах розпізнавання сенс навчання є наступним. Ми маємо тестову вибірку, на якій позначені декілька класів об'єктів. Для кожного зображення є набір ознак. І в цьому випадку алгоритм навчання повинен побудувати таку модель, за якою він зможе проаналізувати нове зображення та прийняти рішення, який з об'єктів чи класів є на зображенні. Кожне з тестових зображень - це точка в просторі ознак. Її координатами є вага кожного з ознак на зображенні. Всі ознаки ми виділяємо вже існуючими детекторами. Метою класифікатора є виділення в просторі ознак областей, що характеристичні для об'єктів класифікації.

Алгоритмів існує дуже велика кількість. Розглянемо деякі з них.

K-means - один з найпростіших алгоритмів навчання. Працює в ситуації, коли групи об'єктів мають добре рознесені центр мас і не мають великого перетину.

AdaBoost - один з найпоширеніших класифікаторів. Зазвичай використовують коли потрібна бінарна класифікація, але в деяких випадках використовують і для більшої кількості класів.

SVM - Один з найпотужніших класифікаторів, що має безліч реалізацій. Вважається досить швидким, але його навчання складніше, ніж у AdaBoost і потрібно вибір правильного ядра. [6]

Але, при всіх плюсах перелічених методів, вони мають також ряд недоліків, які в деяких випадках роблять задачу практично нерозв'язаною за допомогою класичних методів комп'ютерного зору. Наприклад, потрібно знайти розв'язок задач, в яких невідомі закономірності розвитку ситуації і залежності між вхідними та вихідними даними, або треба розв'язати задачу за наявності великого числа неінформативних, шумових вхідних сигналів, або треба класифікувати зображення у новому середовищі та яке не зустрічалось раніше при навчанні.

Всі ці задачі з легкістю можна розв'язати за допомогою штучних нейронних мереж.

2 ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

Прототипом для створення штучних нейронних мереж, як це не дивно, послужили біологічні нейронні мережі. Дві третини всієї сенсорної інформації, яка до нас потрапляє, приходить з зорових органів сприйняття. Більше однієї третини поверхні нашого мозку зайняті двома найголовнішими зоровими зонами - дорсальний зоровий шлях і вентральний зоровий шлях.

Дорсальний зоровий шлях починається в первинної зорової зоні, в нашому тім'ячку і продовжується угору, в той час як вентральний шлях

починається на нашій потилиці і закінчується приблизно за вухами. Все важливе розпізнавання образів, яке у нас відбувається, все, що несе зміст, що ми усвідомлюємо, проходить саме там же, за вухами.

Ця інформація потрібна для розуміння нейронних мереж. Справа в тому, що всі області, які використовуються в нейронних мережах для розпізнавання образів, прийшли до нас саме з вентрального зорового шляху, де кожна маленька зона відповідає за свою строго певну функцію.

Зображення потрапляє до нас з сітківки ока, проходить низку зорових зон і закінчується в скроневій зоні.

У 60-ті роки XX сторіччя, коли тільки починалося вивчення зорових зон мозку, перші експерименти проводилися на тваринах, тому що не було fMRI. Досліджували мозок за допомогою електродів, вживлених в різні зорові зони.

Перша зорова зона була досліджена Девідом Хьюбел і Торстеном Візелем в 1962 році. Вони проводили експерименти на кішках. Кішкам показувалися різні рухомі об'єкти. На що реагували клітини мозку, те і було тим стимулом, який розпізнавала тварина. Навіть зараз багато експерименти проводяться цими драконівськими способами. Але тим не менше це найефективніший спосіб дізнатися, що робить кожна найдрібніша клітинка в нашому мозку.

Таким самим методом було відкрито ще багато важливих властивості зорових зон, які люди використовують в deep learning зараз. Одне з найважливіших властивостей - це збільшення рецептивних полів наших клітин у міру просування від первинних зорових зон до скроневих часток, тобто більш пізнім зоровим зонам. Рецептивне поле - це та частина зображення, яку обробляє кожна клітинка нашого мозку. У кожної клітини своє рецептивне поле. Ця ж властивість зберігається і в нейронних мережах.

Також зі зростанням рецептивних полів збільшуються складні стимули, які зазвичай розпізнають нейронні мережі.

2.1 Огляд основних типів архітектур нейронних мереж

2.1.1 Перцептрон

В основі перцептрона лежить математична модель сприйняття інформації мозком. У найзагальнішому своєму вигляді (як його описував Розенблатт) він представляє систему з елементів трьох різних типів: сенсорів (S-елементи), асоціативних елементів (A-елементи) і реагуючих елементів (R-елементи). Типова архітектура перцептрона представлена на рисунку 2.1.

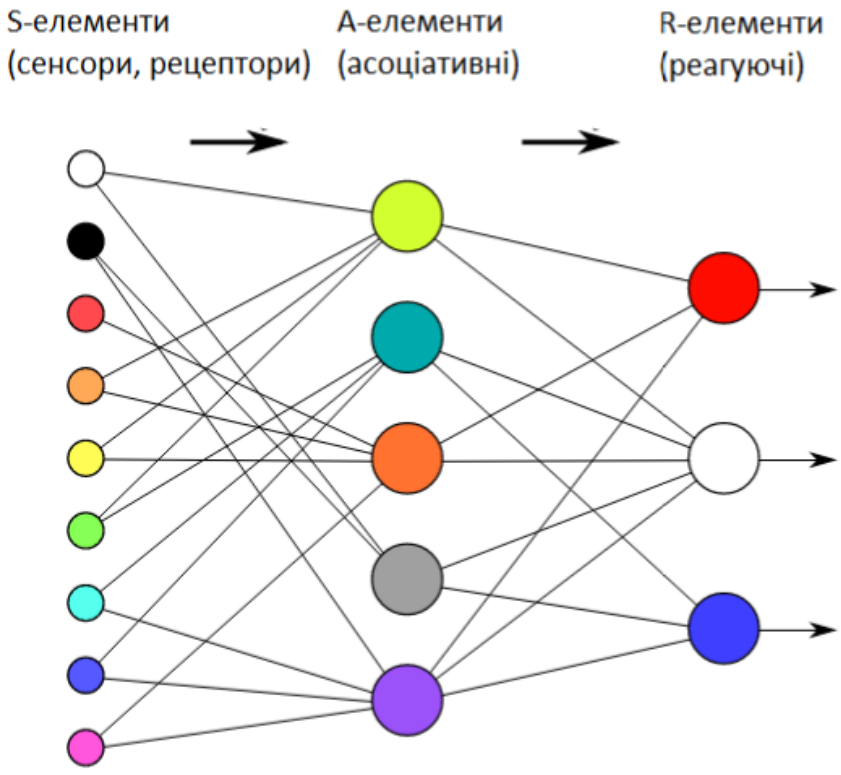


Рисунок 2.1 – Типова архітектура перцептрона

Розглянемо принцип роботи перцептрону:

1. В роботу першими вступають S-елементи. Вони можуть перебувати або в стані спокою, або в стані збудження.

2. S-елементів передають сигнали A-елементам по так званим S-A зв'язкам.

3. Сигнали потрапляють у A-елементи, які ще називають асоціативними елементами. Якщо сигнали, що надійшли на A-елемент, в сукупності перевищують деякий його поріг, то цей A-елемент збуджується і видає сигнал, рівний 1. В іншому випадку генерується нульовий сигнал .

4., Далі сигнали направляються до суматору (R-елемент). R-елемент складає один з одним зважені сигнали від A-елементів та, якщо перевищено певний поріг, генерує вихідний сигнал, рівний 1.

Це означає, що в загальному потоці інформації від очей ми розпізнали обличчя людини. Якщо поріг не перевищено, то вихід перцептрона дорівнює -1. Тобто ми не виділили образ із загального потоку інформації.

Перцептрон застосовується для вирішення класичних задач машинного (класифікація, регресія) навчання як окрема модель, так і в складі більш складних моделей. [7]

2.1.2 Згорткові нейронні мережі

Згорткова нейронна мережа - спеціальна архітектура штучних нейронних мереж, односпрямована (без зворотних зв'язків), принципово багатошарова.

Головна відмінність згорткової архітектури від звичайного перцептрону полягає в тому, що перцептрон представляє собою повнозв'язну нейронну мережу, кожен нейрон пов'язаний з усіма нейронами попереднього шару, причому кожен зв'язок має свій персональний ваговий коефіцієнт. У згорткової нейронної мережі в операції згортки використовується лише обмежена матриця ваг невеликого розміру, яку «рухають» по всьому оброблюваного шару (на самому початку - безпосередньо по вхідному зображенню), формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією. Тобто для різних нейронів

вихідного шару використовуються одна і та ж матриця ваг, яку також називають ядром згортки. Її інтерпретують як графічне кодування якої-небудь ознаки, наприклад, наявність похилої лінії під певним кутом. Тоді наступний шар, що вийшов в результаті операції згортки такою матрицею ваг, показує наявність даної ознаки в оброблюваному шарі і її координати, формуючи так звану карту ознак.

Ідея згорткових нейронних мереж полягає в чергуванні зготкових шарів (C-layers), субдіскретізірующіх шарів (S-layers) і наявності повнозв'язних (F-layers) шарів на виході. Типова структура згорткової нейронної мережі представлена на рисунку 2.2 [10].

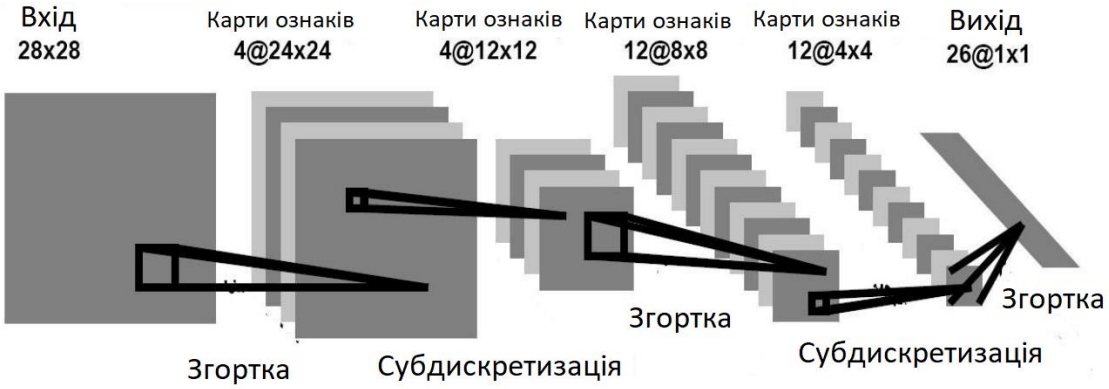


Рисунок 2.2 – Типова структура згорткової нейронної мережі

Шари мережі об'єднують в собі три архітектурні ідеї для досягнення інваріантності до зсуву і спотворення вихідного зображення: локальні поля сприйняття, ваги, що розділяються і просторова субдискретизація.

Локальне сприйняття має на увазі, що на вхід одного нейрона подається не все зображення (або виходи попереднього шару), а лише деяка його область. Такий підхід дозволив зберігати топологію зображення від шару до шару.

Концепція поділюваних ваг припускає, що для великої кількості зв'язків використовується дуже невеликий набір ваг. Суть субдискретизації і

S-шарів в згортальних нейронних мережах полягає в зменшенні просторової розмірності зображення.

Формула, яка відображає рух ядра згортки по вхідному зображенню або карті ознаків:

$$x_{i,y}^{l} = \sum_{a=-\infty}^{+\infty} \sum_{b=-\infty}^{+\infty} w_{a,b}^{l} * y_{(i*s-a)(j*s-b)}^{l-1} + b^{l} \quad \forall i \in (0, \dots, N)\ \forall j \in (0, \dots, M),$$

Де $i, j\ , a, b$ – індекси елементів в матрицях, s – величина шагу згортки.

l та $l-1$ це індекси шарів. x^{l-1} - вхідне зобраежння, або вихід попередньої функції, y^{l-1} – це x^{l-1} після проходження функції активації, w^{l} – ядро згортки, b^{l} – нейрон зміщення, x^{l} – результат операції згортки.

Функція втрат зазвичай має вид: $E = \sum_{i=0}^{n} \frac{1}{2}(y_i^{truth} - y_i^{l})^2$, де n - кількість класів, y^{l} – вихід моделі, y^{truth} -правильні відповіді. Або в якості функції втрат для перевірки дії часто використовують перехресну ентропію, яку ми докладніше розглянемо у наступних пунктах.

До класичних задач згорткових нейронних мереж можна віднести такі завдання, як ідентифікація об'єкта, семантична сегментація, розпізнавання осіб, розпізнавання частин тіла людини, семантичне визначення меж, виділення об'єктів уваги на зображенні і виділення нормалей до поверхні (Рисунок 2.3).

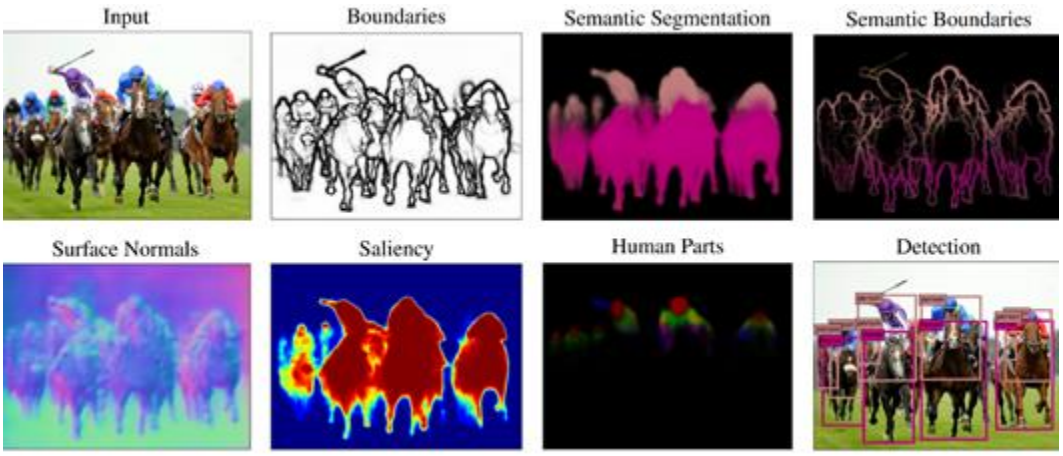


Рисунок 2.3 – Класичні задачі згорткових нейронних мереж

2.2 Вибір функцій активації

Одним з найважливіших аспектів побудови нейронних мереж є функція активації, яка привносить в мережу нелінійність. Вибір тієї чи іншої функції активації часто залежить від умов завдання та структури мережі. Деякі з розглянутих функцій застосовуються лише у застарілих системах або в цілях навчання, але вважаються класичними і тому важливо їх згадати у рамках цієї роботи

2.2.1 Сигмоїд

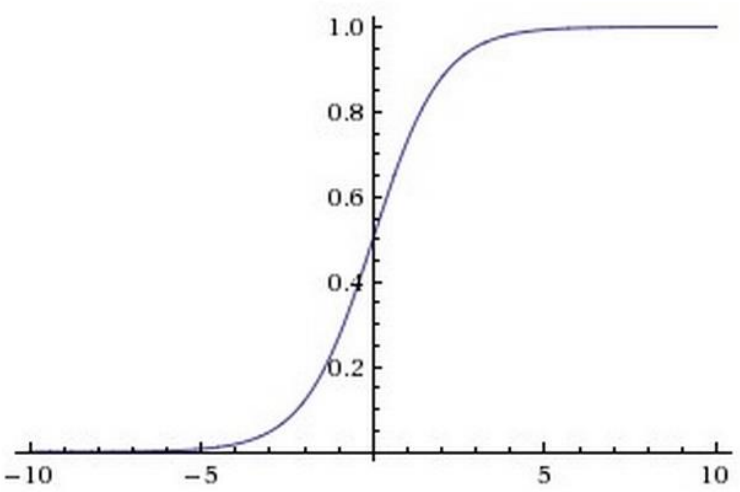


Рисунок 2.4 – Сигмоїд

Сигмоїд є монотонно зростаючою всюди диференційованою S-образною нелінійною функцією з насиченням (Рисунок 2.4). Виражається наступною формулою:

$\sigma(x) = \frac{1}{1+e^{-x}}$, де x - вхід нейрона.

Ця функція приймає на вході довільне дійсне число, а на виході дає дійсне число в інтервалі від 0 до 1. Зокрема, великі (за модулем) негативні

числа перетворюються в нуль, а великі позитивні - в одиницю. Наразі сигмоїд втратив свою колишню популярність і використовується дуже рідко. Ця функція має два серйозні недоліки:

- Насичення сигмоїда призводить до загасання градієнтів.

Тобто, при насиченні функції з тієї чи іншої сторони (0 або 1), градієнт на цих ділянках стає близький до нуля. Також, слід бути дуже обережними при ініціалізації ваг сігмоїдних нейронів, щоб запобігти насичення. Наприклад, якщо вихідні ваги мають занадто великі значення, більшість нейронів перейде в стан насичення, в результаті чого мережа буде погано навчатись.

- Вихід сигмоїда не відцентрований відносно нуля.

Ця властивість є небажаною, оскільки нейрони в наступних шарах отримуватимуть значення, що не відцентровані відносно нуля, що впливає на динаміку градієнтного спуску.

2.2.2 Гіперболічний тангенс

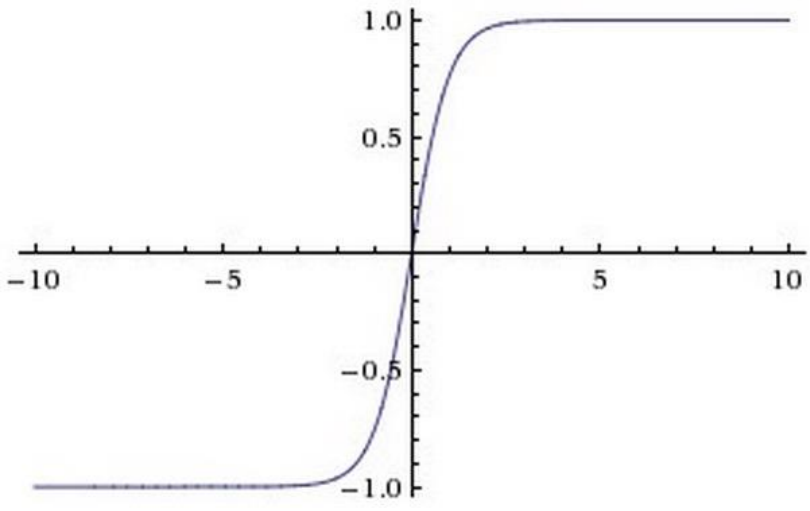


Рисунок 2.5 – Гіперболічний тангенс

Гіперболічний тангенс приймає на вході довільне дійсне число, а на виході дає дійсне число в інтервалі від -1 до 1 (Рисунок 2.5). Як і сигмоїд, гіперболічний тангенс може насичуватися. Однак, на відміну від сигмоїда,

вихід даної функції відцентрований відносно нуля. Отже, на практиці завжди краще використовувати гіперболічний тангенс, а не сигмоїд. Виражається наступною формулою:

$x = \frac{e^{2x}-1}{e^{2x}+1}$, де x - вхід нейрона

2.2.3 ReLU (Rectified Linear Unit)

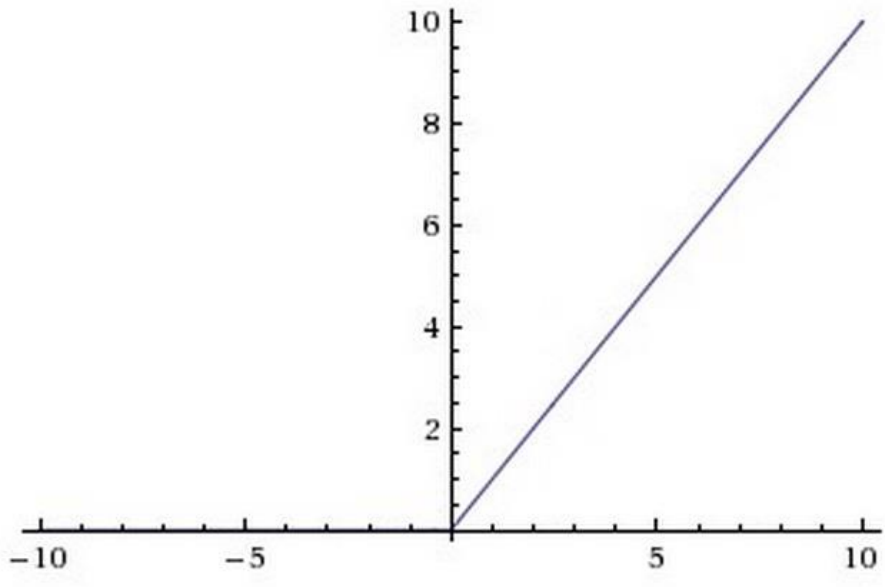


Рисунок 2.6 – ReLU

Нейрони з цією функцією активації називаються ReLU (rectified linear unit). ReLU має наступну формулу $f(x) = \max(0, x)$ і реалізує простий пороговий перехід в нулі (Рисунок 2.6).

Розглянемо позитивні і негативні сторони ReLU.

Позитивні сторони:

- Обчислення сигмоїд і гіперболічного тангенса вимагає виконання ресурсномістких операцій, таких як зведення в ступінь, в той час як ReLU може бути реалізований за допомогою простого порогового перетворення матриці активацій в нулі. Крім того, ReLU не схильний до насичення.

- Застосування ReLU істотно підвищує швидкість збіжності стохастичного градієнтного спуску в порівнянні з сигмоїдом та гіперболічним тангенсом. Вважається, що це обумовлено лінійним характером і відсутністю насичення даної функції.

Негативні сторони:

- На жаль, ReLU не завжди достатньо надійні в процесі навчання і можуть виходити з ладу. Наприклад, великий градієнт, що проходить через ReLU, може привести до такого оновлення ваг, що даний нейрон ніколи більше не активується. Якщо це станеться, то, починаючи з даного моменту, градієнт, що проходить через цей нейрон, завжди буде дорівнює нулю. Відповідно, даний нейрон буде необоротно виведений з ладу. Наприклад, при дуже великій швидкості навчання (learning rate), може виявитися, що до 40% ReLU «мертві» (тобто, ніколи не активуються). Ця проблема вирішується за допомогою вибору належної швидкості навчання.

Варто зауважити, що існує ціле сімейство різних модифікації ReLU, які вирішують проблеми надійності цієї передавальної функції при проходженні через нейрон великих градієнтів: Leaky ReLU, Parametric ReLU, Ran-domized ReLU. [8]

2.2.4 Softmax

Softmax - це узагальнення логістичної функції для багатовимірного випадку. Функція перетворює вектор z розмірності K в вектор σ тієї ж розмірності, де кожна координата σ_i отриманого вектора представлена дійсним числом в інтервалі [0,1] і сума координат дорівнює 1. [9]

Координати σ_i обчислюються наступним чином:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$$

Функція Softmax часто застосовується для задач класифікації, коли кількість можливих класів більше ніж два. Координати σ_i отриманого вектора при цьому трактуються як ймовірності того, що об'єкт належить до класу i. Вектор-стовпець z при цьому розраховується наступним чином:

$$z = w^{\mathrm{T}} x - \theta$$

де x - вектор-стовпець ознак об'єкта розмірності $\mathrm{M} \times 1$; w^{T} - транспонована матриця вагових коефіцієнтів ознак, що має розмірність $\mathrm{K} \times \mathrm{M}$; θ - вектор-стовпець з граничними значеннями розмірності $\mathrm{K} \times 1$, де K - кількість класів об'єктів, а M - кількість ознак об'єктів.

2.3 Алгоритми навчання нейронних мереж

Кожна створена нейронна мережа вимагає навчання, в іншому випадку шуканий результат буде отримати неможливо. Методів навчання існує велика кількість. Розглянемо деякі з них.

2.3.1 Пакетний градієнтний спуск

Цей метод є одним з найпоширеніших. Свою назву метод заслужив тому, що в процесі його функціонування вихідна помилка мережі, яка обчислюється на кожній ітерації, поширюючись по нейронної мережі від виходу до входу (тобто в напрямку, протилежному поширенню сигналу), використовується для розрахунку коригування ваг нейронів кожного прихованого шару. Розраховується за формулою:

$\theta = \theta - \eta \nabla_\theta \mathrm{C}(\theta)$, де θ – параметри моделі; η – це крок методу (швидкість навчання).

У завданнях машинного навчання його називають швидкістю навчання, який виконується в напрямі локального мінімуму.

2.3.2 Стохастичний градієнтний спуск

Метод стохастичного градієнтного спуску сходиться швидше, ніж попередній, який розраховує градієнт для подібних екземплярів датасету, а потім лише раз оновлює параметри.

Стохастичний градієнтний спуск оновлює параметри для кожного екземпляра з навчальної вибірки: $(x_{(i)},y_{(i)})$: $\theta = \theta - \eta\nabla_\theta \mathrm{C}(\theta; x(i); y(i))$, де θ – параметри моделі; η – це крок методу (швидкість навчання).

2.3.3 Міні-пакетний градієнтний спуск

Міні-пакетний градієнтний спуск є «золотою серединою» між пакетним ГС і стохастичним ГС: $\theta = \theta - \eta\nabla_\theta \mathrm{C}(\theta; x(i{:}\, i+n); y(i{:}\, i+n))$, де θ – параметри моделі; η – це крок методу (швидкість навчання).

Цей метод зменшує флуктуації при оновленні параметрів, що часто приводить до покращення збіжності. Розмір міні-пакету зазвичай обирається між 50 і 256 екземплярами. При навчанні нейронних дуже часто обирають саме міні-пакетний ГС.

2.3.4 Метод моменту

Метод моменту – це метод, який допомагає прискорити Стохастичний градієнтний спуск у відповідному напрямку і згладжує коливання.

При використанні метода моментів буде Момент буде збільшувати величину оновлення параметрів для вимірів, градієнти яких вказують в тому ж напрямі, куди відбувається рух, і зменшувати величину оновлення для вимірів, градієнти яких змінюють напрям.

$$v_t = yv_{t-1} + \eta\nabla_\theta \mathrm{C}(\theta), \theta = \theta - v_t$$

Величина моменту γ встановлюється зазвичай близько до 0.9.

2.3.5 Адам

Адам – це метод стохастичної оптимізації**.** Розраховується за формулою:

$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t,\ v_t = \beta_2 v_{t-1} + (1-\beta_2) g_{t2},$

Де m_t – оцінка першого моменту (середнє);

v_t– оцінка другого моменту (дисперсія) градієнту.

На значення моментів в цьому методі накладається штраф:

$m_{\hat{t}}=m-\beta_{1t}\ ,\ v_{\hat{t}}=v_{t1}-\beta_{2t}$

Далі отримані значення використовують для оновлення параметрів за формулою: $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}}+\epsilon}\widehat{m_t}$

Автори методу пропонують наступні значення для параметрів:

$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

2.4 Регуляризація

Перенавчання (overfitting) - одна з проблем глибоких нейронних мереж, що складається в наступному: модель добре пояснює тільки приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того щоб вчитися класифікувати приклади, які не брали участі в навчанні (втрачаючи здатність до узагальнення).

Регуляризація - спосіб запобігти перенавчання моделі. Нижче розглянемо деякі методи регуляризації.

2.4.1 L2-регуляризація

Даний метод штрафує модель за допомогою квадратів ваг. Тобто, для кожної ваги ω ми додаємо до цільової функції доданок $\frac{1}{2}\lambda\omega 2$, де λ - коефіцієнт регуляризації. Множник $\frac{1}{2}$ використовується для того, щоб градієнт цього доданка по параметру ω дорівнював $\lambda\omega$, а не $\lambda\omega 2$. Інтуїтивна інтерпретація L2-регуляризації полягає в тому, що вона штрафує вектори ваг з великими значеннями, і лише трохи зачіпає вектори з середніми значеннями.

2.4.2 L1-регуляризація

В рамках цього методу для кожного ваги ω ми додаємо до цільової функції доданок $\lambda|\omega|$. Застосовується також комбінація L1- і L2-регуляризації: $\lambda 1|\omega| + \Lambda 2\omega 2$

L1-регуляризація має цікаву властивість, що полягає в тому, що в її результаті вектори ваг стають розрідженими (тобто дуже близькими до нуля). Іншими словами, нейрони з L1-Регуляризацією в підсумку використовують тільки невелику підмножину найбільш важливих входів і, відповідно, майже не схильні до впливу зашумлених входів.

2.4.3 Обмеження норми вектора ваг

В рамках даного методу ми задаємо абсолютну верхню межу для норми вектора ваг кожного нейрона. Дотримання обмеження забезпечується за допомогою проектованого градієнтного спуску. На практиці це реалізується в такий спосіб: оновлення ваг виконується як зазвичай, а потім вектор ваг ω кожного нейрона обмежується так, щоб виконувалася умова$\|\omega\|2 < c$. Зазвичай значення c складає приблизно 3 або 4. Деякі дослідники повідомляють про позитивний ефект при використанні даного методу регуляризації. Одне з корисних властивостей цього методу полягає в тому, що він дозволяє запобігти різкого зростання ваг навіть при дуже великій швидкості навчання, тому що оновлення ваг завжди обмежене.

2.4.4 Dropout

Дропаут (dropout) - простий і дуже ефективний метод регуляризації, що доповнює вищезгадані методи. Він був недавно запропонований в роботі [11]. Суть методу полягає в тому, що в процесі навчання із загальної мережі випадковим чином багаторазово виділяється підмережа, і оновлення ваг виконується тільки в рамках цієї підмережі. Нейрони потрапляють в підмережа з ймовірністю p, яка називається коефіцієнтом дропаута. Під час

тестування дропаут не застосовується, замість цього ваги множаться на коефіцієнт дропаута, в результаті чого можна отримати усереднену оцінку для ансамблю всіх підмереж. На практиці коефіцієнт дропаута p зазвичай вибирають рівним 0,5, але його можна підібрати за допомогою валідаційного набору даних. На рисунку 2.7 зображений принцип роботи Dropout. [13]

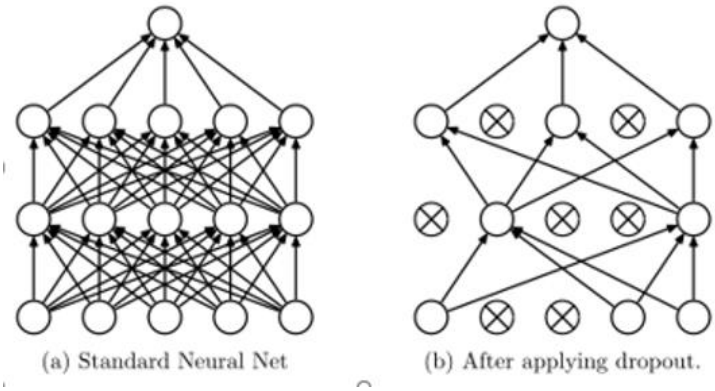


Рисунок 2.7 – принцип роботи Dropout

2.4.5 Перехресна ентропія і функція втрат

Для того, щоб нейронна мережа працювала правильно, її необхідно навчити. Для цього, в свою чергу, треба визначити «оцінку якості» роботи нейромережі. Одним з найбільш дієвих у цьому плані є перехресна ентропія.

При застосуванні цього метода для певного вихідного імовірнісного вектора $\bar{y}$, що порівнюється з фактичним вектором $\bar{\hat{y}}$, втрата (для k-го класу) буде визначатися як $\mathcal{L}(\bar{y}, \bar{\hat{y}}) = -\sum_{i=1}^{k} \bar{\hat{y}} \log y_i$

3 РОЗРОБКА СИСТЕМИ КЛАСИФІКАЦІЇ

3.1 Опис вхідних даних

Вхідними даними до задачі класифікації рукописних цифр є база даних з рукописних цифр MNIST. Набір даних складається всього з 70 000 зображень: 60 000 навчальних (використовуваних для створення моделі) і 10

000 тестових (застосовуваних для оцінки точності моделі). Кожне зображення MNIST - це оцифрована картинка однієї цифри, написаної від руки. Кожне зображення має розмір 28 × 28 пікселів. Кожне значення пікселя лежить в діапазоні від 0 (представляє білий колір) до 255 (представляє чорний колір). Проміжні значення відображають відтінки сірого. На рисунку 3.1 показані перші вісім зображень в навчальному наборі. Сама цифра, яка відповідає кожному зображенню, очевидна людині, але для комп'ютерів ідентифікація цифр - дуже складне завдання. [12]

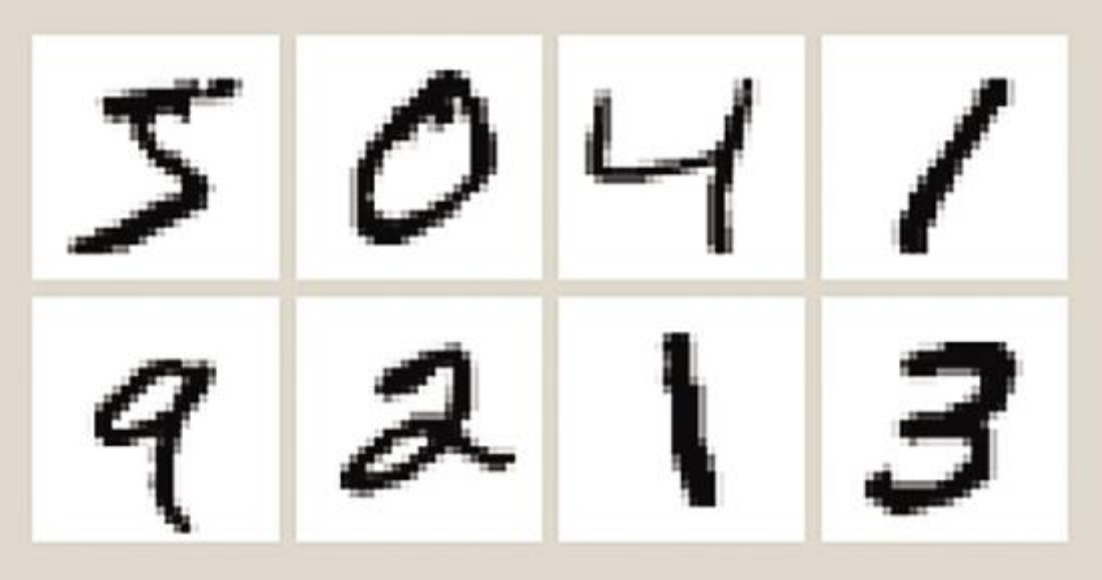


Рисунок 3.1 — Перші вісім навчальних зображень MNIST

Вихідними даними є класи цифр 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, на зображеннях яких навчалась нейронна мережа, і яке вона до того не бачила. Одже, буде побудована модель, що може класифікувати цифри на 10 класів, які представлені в навчальній вибірці.

3.2 Розмірність задачі

Ми маємо базу даних рукописних цифр, розмір яких формує розмірність задачі. Основні статистики вхідних даних представлена в таблиці 3.1.

Таблиця 3.1

Розмірність задачі

Кількість тренувальних прикладів	60000
Кількість тестових прикладів	10000
Кількість класів	10
Строк пікселів на зображенні	28
Стовбців пікселів на зображенні	28

3.3 Дослідження впливу архітектури та гіперпараметрів на результати класифікації

Однією з найважливіших речей, які впливають на точність класифікації та якість моделі нейронної мережі є її архітектура. Наразі не існує універсального методу проектування архітектури нейронних мереж. Кожна задача вимагає особливого підходу до проектування власної архітектури для оптимального розв’язку.

Тож, у даному розділі будуть описані спроби відшукати оптимальну архітектуру нейронної мережі та підібрати оптимальні гіпермараметри моделей, зважаючи на наявні обчислювальні ресурси і особливості задачі, що вирішується.

У даній задачі в якості бекенду для розрахунків використовується Keras, бібліотека для машинного навчання Tensorflow та мова програмування Python.

3.3.1 Одношаровий перцептрон

Побудуємо один з найпростіших типів архітектур нейронних мереж – одношаровий перцептрон.

На вхід подаються зображення з бази рукописних цифр MNIST, яка складає в собі зображення 28x28 пікселів. Створюємо шар, що складається з 10 нейронів, в якому кожен нейрон пов'язаний з усіма входами та задаємо

функцію активації вихідного шару – ‘Softmax’, яка застосовується для задач класифікації. На виході маємо 10 класів.

Роздрукуємо короткі характеристики даної моделі (рисунок 3.2).

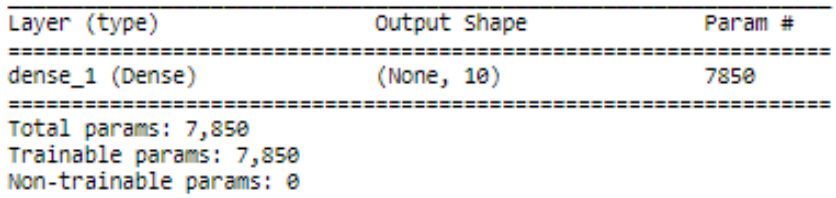


Рисунок 3.2 – Характеристики одношарового перцептрону

Навчання виконуємо "пачками" (mini-batch) по 128 прикладів. При цьому, здійснюється 20 проходів за всіма вхідними прикладами.

Результатом роботи такої моделі є точність на тестових даних: 90.59%.

3.3.2 Двошаровий перцептрон

Ускладнимо модель, додавши до неї другий шар.

Перший шар буде складатися з 200 нейронів, з'єднаних з усіма входами. Функція активації 'Relu'. Другий шар – з 10 нейронів, та функції активації 'Softmax'.

Роздрукуємо короткі характеристики даної моделі (рисунок 3.3).

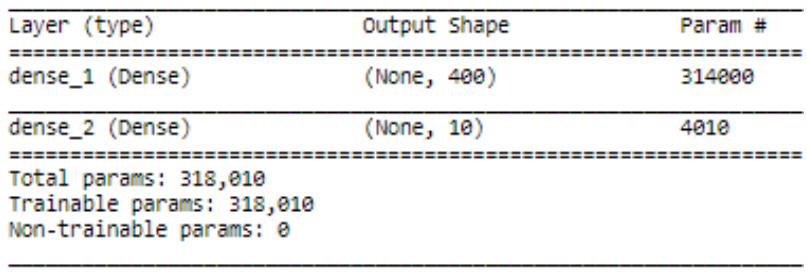


Рисунок 3.3 – Характеристики двошарового перцептрону

Навчання виконується тим самим методом, як і для одношарового перцептрону.

Результатом роботи такої моделі є точність на тестових даних: 93.26%

В ході роботи також виконувалась нормалізація даних. (Рисунок 3.4)

```
# Нормалізація даних (інтенсивність кожного пиксела від 0 до 255)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

Рисунок 3.4 – нормалізація даних

Заради експерименту, була виконати ті самі дії, навчити ту саму модель, але без передобробки. Результатом була точність роботи на тестових даних 9.58%, що є катастрофічно неприйнятним результатом.

3.3.3 Тришаровий перцептрон

Спробуємо досягти більшої точності, додавши третій (прихований) шар, який буде складатись з 400 нейронів.

Тоді наша модель приймає наступний вид. Перший шар складається з 200 нейронів та функції активації 'Relu', другий прихований – 400 нейронів та функції активації 'Relu' та третій й шар – з 10 нейронів, та функції активації 'Softmax'.

Роздрукуємо короткі характеристики даної моделі (рисунок 3.5).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 200)               157000
_________________________________________________________________
dense_2 (Dense)              (None, 400)               80400
_________________________________________________________________
dense_3 (Dense)              (None, 10)                4010
=================================================================
Total params: 241,410
Trainable params: 241,410
Non-trainable params: 0
```

Рисунок 3.5 – Характеристики тришарового перцептрону

Результатом роботи такої моделі є точність на тестових даних: 94.12%

Тож, ми бачимо, що двошарова мережа працює істотно краще одношарової, та тришарова мережа показала лише трохи кращі результати, ніж двошарова. Також відомо, при збільшенні числа шарів у послідовній повнозв'язній мережі може виникнути ефект перенавчання, тому, подальшим кроком є використання згорткових шарів.

3.3.4 Згорткова нейронна мережа №1

Побудуємо найпростіший тип нейронної мережі, де перший шар – згортка з функцією активації «ReLU», другий шар – шар субдискретизації (MaxPooling). Потім за допомогою функції «flatten» перетворюємо 2-мірне зображення в одномірне та додаємо повнозв'язний шар для класифікації зображення з функцією активації «Softmax»

Роздрукуємо короткі характеристики даної моделі (рисунок 3.6).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 24, 24)        832
_________________________________________________________________
activation_1 (Activation)    (None, 32, 24, 24)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 12, 12)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                46090
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
=================================================================
Total params: 46,922
Trainable params: 46,922
Non-trainable params: 0
_________________________________________________________________
```

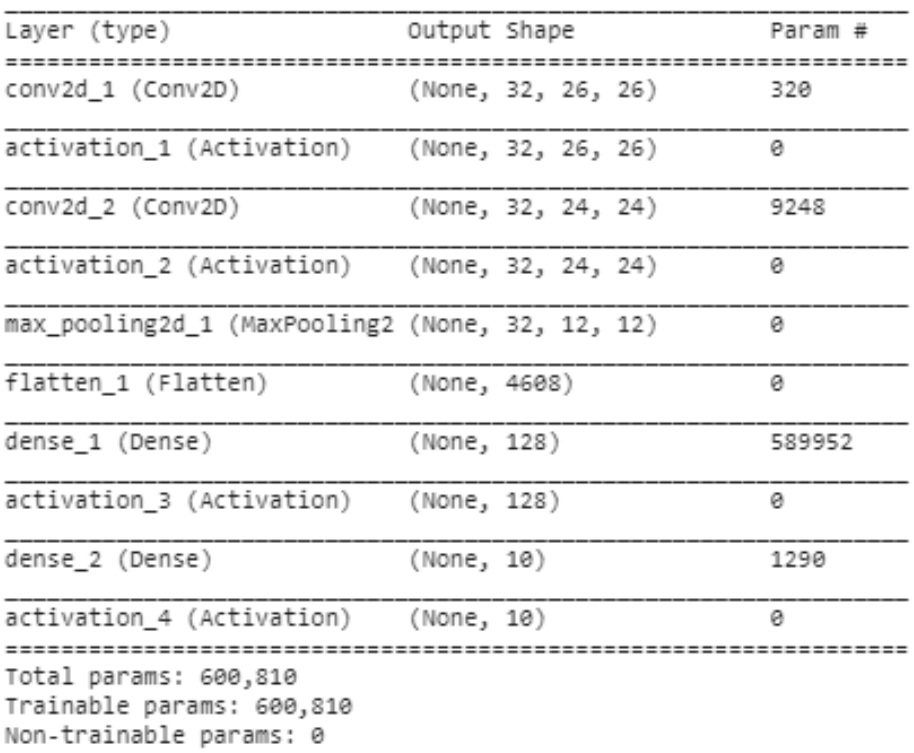
Рисунок 3.6 – Характеристики згорткової мережі №1

Результатом роботи такої моделі є точність на тестових даних: 98.69%

3.3.5 Згорткова нейронна мережа №2

Додамо кількість шарів, тим самим підвищимо глибину мережі.

Роздрукуємо короткі характеристики даної моделі (рисунок 3.7).



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 26, 26)	320
activation_1 (Activation)	(None, 32, 26, 26)	0
conv2d_2 (Conv2D)	(None, 32, 24, 24)	9248
activation_2 (Activation)	(None, 32, 24, 24)	0
max_pooling2d_1 (MaxPooling2	(None, 32, 12, 12)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 128)	589952
activation_3 (Activation)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290
activation_4 (Activation)	(None, 10)	0

Total params: 600,810
Trainable params: 600,810
Non-trainable params: 0

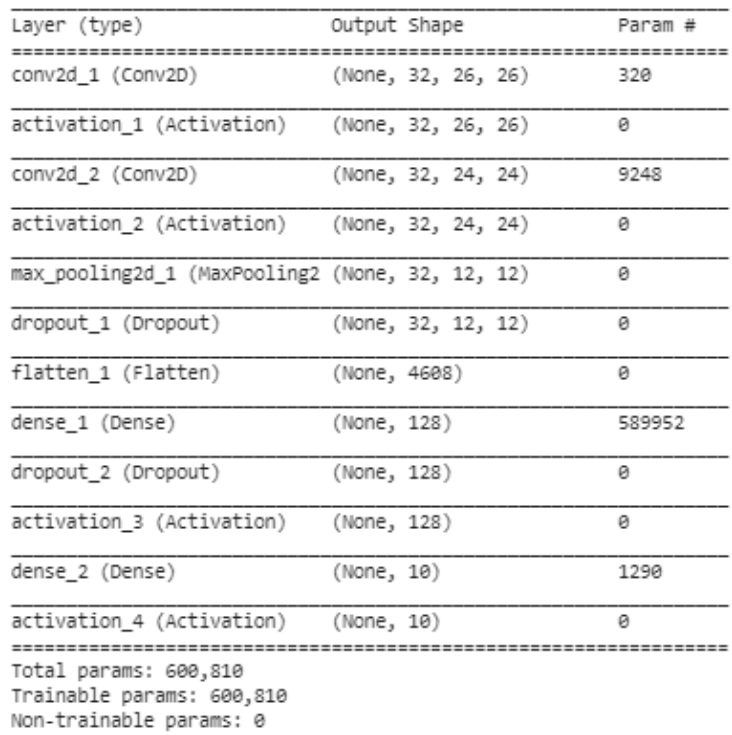
Рисунок 3.7 – Характеристики згорткової мережі №2

Результатом роботи такої моделі є точність на тестових даних: 99.21%

3.3.6 Згорткова нейронна мережа №3

Додамо шар «Dropout», оскільки він перешкоджає появі залежних зв'язків між вузлами, що дозволяє мережі ефективніше пізнавати більш надійні відносини.

Роздрукуємо короткі характеристики даної моделі (рисунок 3.8).



Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 26, 26)	320
activation_1 (Activation)	(None, 32, 26, 26)	0
conv2d_2 (Conv2D)	(None, 32, 24, 24)	9248
activation_2 (Activation)	(None, 32, 24, 24)	0
max_pooling2d_1 (MaxPooling2	(None, 32, 12, 12)	0
dropout_1 (Dropout)	(None, 32, 12, 12)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 128)	589952
dropout_2 (Dropout)	(None, 128)	0
activation_3 (Activation)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290
activation_4 (Activation)	(None, 10)	0

Total params: 600,810
Trainable params: 600,810
Non-trainable params: 0

Рисунок 3.8 – Характеристики згорткової мережі №1

Результатом роботи такої моделі є точність на тестових даних: 99.22

Отже, згорткова нейронна мережа, як і очікувалось, показала кращій результат ніж послідовна. Після збільшення глибини мережі шляхом додавання кількості шарів точність тесту підвищилась. Додавання Dropout шару веде за собою додаткове збільшення точності.

Отримана точність роботи 99.22 є прийнятним результатом і означає завершення навчання.

В додатках А та Б приведений код двох найефективніших варіантів з побудованих архітектур – тришарового перцептрону та згорткові архітектури №3 з додаванням шара Dropout.

4 ВИСНОВКИ

В ході даної роботи була спроектовані архітектура штучної нейронної мережі на базі згорткової нейронної мережі і повнозв'язного перцептрона – класифікатора.

Отримана нейронна мережа була навчена на наборі рукописних цифр та дала точність класифікації на валідаційному наборі даних 99,22%. З точки зору передобробки даних була проведена нормалізація, що сприяло покращенню результатів.

Було проведено ряд експериментів над архітектурою моделі, що емпіричним шляхом дозволило досягти високих результатів та дійти висновку, що поєднання згорткових та послідовних шарів нейроннох мережі веде до покращення результатів, так само, як і збільшення кількості згорткових та субдискретизуючих шарів.

Розроблена модель може бути використана для вирішення більш глобальної задачі детектування і розпізнавання рукописного тексту на фото та відеопотоці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. http://znaimo.com.ua/Машинне_навчання
2. https://oxozle.com/2015/03/29/metody-raspoznavaniya-obrazov-chast-1
3. https://ru.wikipedia.org/wiki/%CC%E0%F2%E5%EC%E0%F2%E8%F7%E5%F1%EA%E0%FF_%EC%EE%F0%F4%EE%EB%EE%E3%E8%FF
4. https://habr.com/post/113626/
5. http://um.co.ua/8/8-2/8-201382.html
6. https://habr.com/post/208090
7. https://neuralnet.info/chapter/персептроны/
8. http://datareview.info/article/eto-nuzhno-znat-klyuchevyie-rekomendatsii-po-glubokomu-obucheniyu-chast-2
9. https://ru.wikipedia.org/wiki/Softmax
10. Y. Le Cun and Yoshua Bengio. Convolutional networks for images, speech, and time series. In Michael A.Arbib, editor, The Handbook of Brain Theory and Neural Networks, pages 255–258. MITPress, Cambridge, Massachusetts, 1995.
11. A Simple Way to Prevent Neural Networks from Overfitting Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov; 15(Jun):1929−1958, 2014.
12. Матеріали официального сайту THE MNIST DATABASE of handwritten digits http://yann.lecun.com/exdb/mnist
13. http://datareview.info/article/eto-nuzhno-znat-klyuchevyie-rekomendatsii-po-glubokomu-obucheniyu-chast-2/

Додаток А

Програмна реалізація розпізнавання рукописних цифр з набора даних MNIST за допомогою повнозв'язного перцептрону.

```
import numpy #модуль для роботи з масивами
from keras.datasets import mnist
from keras.models import Sequential #модель нейронної марежі, вузли якої з'єднані один з одним
from keras.layers import Dense #тип з'єднання нейронів, в которой все нейроны поперед. рівня поєд. зі всіма нейр. наст. рівня
from keras.utils import np_utils #утилити для роботи з масивами
from ._conv import register_converters as _register_converters
Using TensorFlow backend.
# Встановлюємо seed для повторюємості результатів
numpy.random.seed(41)
# Завантажуємо дані
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# Перетворювання размірности зображень
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
# Нормалізація даних (інтенсивність кожного пиксела від 0 до 255)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
# Перетворювання міток в категорії
# 0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
# 3 = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
Y_train = np_utils.to_categorical(y_train, 10)
```

```
Y_test = np_utils.to_categorical(y_test, 10)
# Створюємо послідовну модель
model = Sequential()
# Добаємо рівні мережі
# Обидва шара мережі мають тип Dense
# В першому шарі 800 нейронів, у кожного 784 входа,
# функція активації ReLU, ваги ініціюються випадковими значеннями за допо могою нормального розподілу
# У другому шарі 10 нейронів, функція активации softmax (нормализована екс поненціальна функція),
# ваги ініціюються випадковими значеннями за допомогою нормального розп оділу
model_3l.add(Dense(200, input_dim=784, activation="relu", kernel_initializer="n ormal"))
model_3l.add(Dense(400, input_dim=784, activation="relu", kernel_initializer="n ormal"))
model_3l.add(Dense(10, activation="softmax", kernel_initializer="normal"))# Ко мпілюємо модель
# Метод навчання SGD (стохастичного градієнтного спуску), міра похибки c ategorical_crossentropy, метрика - точність
model.compile(loss="categorical_crossentropy", optimizer="SGD", metrics=["acc uracy"])
# Навчаємо мережу
# X_train - зображення, Y_train - правильні відповіді
# Для оптимизації вик. стохастичний метод градієнтного спуску, розмір мін і виборки 200 элементів, кількість эпох 25
# verbose дозволяє виводити статистичну інформацію
# validation_split=0.2 означає чщо 20 відсотків з X_train буде використов. для перевірочної вибірки
```

```
# Оцінюємо якість навчання на тест. вибірці
scores = model.evaluate(X_test, Y_test, verbose=0)
print("Точність роботи на тестових даних: %.2f%%" % (scores[1]*100))
```

Точність роботи на тестовых даних: 94.12%

Додаток Б

1. Програмна реалізація розпізнавання рукописних цифр з набора даних MNIST за допомогою глибокої згорткової нейронної мережі.

```
import time # функія для часу

import matplotlib.pyplot as plt # графіки

import numpy as np # модуль для роботи з масивами

from keras.utils import np_utils #утилити для роботи з масивами

from keras.models import Sequential #послідовна модель нейронної марежі, вузли якої з'єднані один з одним

from keras.layers.convolutional import Convolution2D, MaxPooling2D #згортк шар, що прац з двовимірними даними, зменшення розмірності

from keras.layers import Activation, Flatten, Dense, Dropout #активація, перетворення з двовимірного простору у одновимірний, повнозв'язна НМ, регуляризація (техніка проти перенавчання)

from keras.optimizers import SGD # метод стохастичного градієнтного спуска

from keras.layers.normalization import BatchNormalization # Batch normalization - це техніка регуляризації мережі.

% matplotlib inline

np.random.seed(2018)

# Завантажуємо дані

from keras.datasets import mnist

(train_features, train_labels), (test_features, test_labels) = mnist.load_data()
```

```
_, img_rows, img_cols =  train_features.shape

num_classes = len(np.unique(train_labels))

num_input_nodes = img_rows*img_cols

print ("Кількість тренувальних прикладів: %d"%train_features.shape[0])

print ("Кількість тестових прикладів: %d"%test_features.shape[0])

print ("Кількість класів: %d"% num_classes)

print ("Строк на зображенні: %d"%train_features.shape[1])

print ("Стовбців на зображенні: %d"% train_features.shape[2])

Кількість тренувальних прикладів: 60000

Кількість тестових прикладів: 10000

Кількість класів: 10

Строк на зображенні: 28

Стовбців на зображенні: 28

# Нормалізація даних (інтенсивність кожного пиксела від 0 до 255)

train_features = train_features.reshape(train_features.shape[0], 1, img_rows, img_cols).astype('float32')

test_features = test_features.reshape(test_features.shape[0], 1, img_rows, img_cols).astype('float32')

train_features /= 255

test_features /= 255

# перетворимо мітки класів на мітки бінарного класу

train_labels = np_utils.to_categorical(train_labels, num_classes)

test_labels = np_utils.to_categorical(test_labels, num_classes)
```

```
# функція точності
def accuracy(test_x, test_y, model):
    result = model.predict(test_x)
    predicted_class = np.argmax(result, axis=1)
    true_class = np.argmax(test_y, axis=1)
    num_correct = np.sum(predicted_class == true_class)
    accuracy = float(num_correct)/result.shape[0]
    return (accuracy * 100)
# визначемо архітектуру мережі
model06 = Sequential()
model06.add(Convolution2D(32, 3, 3, border_mode='valid', input_shape=(1, 28, 28)))
model06.add(Activation("relu"))
model06.add(Convolution2D(32, 3, 3, border_mode='valid'))
model06.add(Activation("relu"))
model06.add(MaxPooling2D(pool_size=(2, 2)))
model06.add(Dropout(0.25))
model06.add(Flatten())
model06.add(Dense(128))
model06.add(Dropout(0.5))
model06.add(Activation("relu"))
model06.add(Dense(num_classes))
```

```
model06.add(Activation("softmax"))

# компілюємо

model06.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# тренування моделі

start = time.time()

model06_info = model06.fit(train_features, train_labels, batch_size=128, \

                nb_epoch=20, verbose=1, validation_split=0.2)

end = time.time()

# точність

print ("Точність на тестових даних: %0.2f"%accuracy(test_features, test_labels, model06))
```

Точність роботи на тестовых даних: 99.22%