Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра математики факультету інформатики

Використання квадратичних ядер у згорткових нейронних мережах

Текстова частина до курсової роботи

за спеціальністю „Комп'ютерні науки" 122

Керівник курсової роботи

к.ф.-м.н., ст.в. Швай Н.О.

_____

(підпис)

"____" _____ 2021 р.

Виконав студент КН-4

Мощицький М.Д.

"____" _____ 2021 р.

Київ 2021

Зміст

ВВЕДЕННЯ

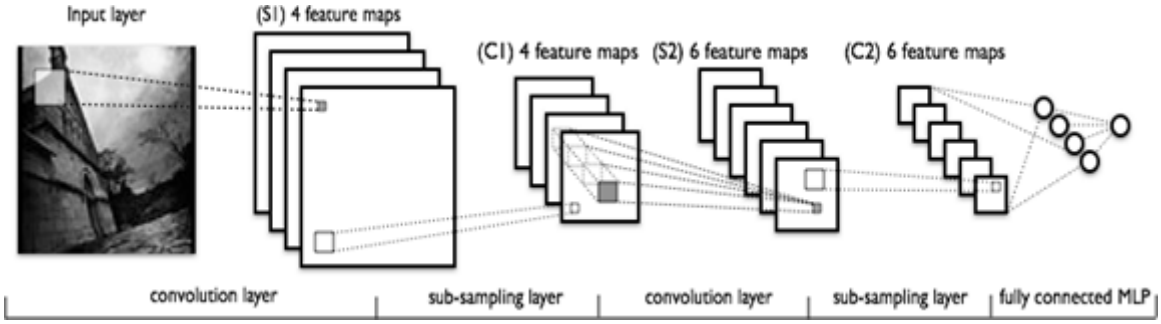
У сучасному світі, коли відбувається активний розвиток технологій, а автоматизація приходить в будь-який процес людського життя, не дивно, що комп'ютерний зір розвинувся і досяг часу, коли це вже не майбутнє, а дуже реальне сьогодення. Швидкий розвиток теорій комп'ютерного зору призвів до появи технологій машинного зору. Якщо коротко, то машинний зір - це технологія, яка допомагає обладнанню побачити процес виробництва чогось, проаналізувати дані і прийняти обгрунтоване рішення. Ще в 2012 році технології машинного зору були не настільки досконалими і могли розпізнати близько 65-70% об'єктів, які були в системі для аналізу. Але все змінилося з появою нейронних мереж. А саме, в 2012 році, коли Алекс Крєвскі виграв конкурс ImageNet, скоротивши рекорд похибки класифікації з 26%до 15%, що стало проривом.

Сьогодні глибоке навчання лягає в основі багатьох компаній: Facebook використовує нейронні мережі для автоматизованих алгоритмів тегування, Google для пошуку фотографій користувачів, Amazon для генерації рекомендацій щодо продуктів, Pinterest для персоналізації домашньої сторінки користувача та Instagram для пошуку користувача.

Але класичним, і, мабуть, найпопулярнішим, використанням мереж є обробка зображень.

1 ОПИС МЕТОДІВ ІАЛГОРИТМІВ, ПІДХОДИ ДО ВІЗУАЛІЗАЦІЇ

SNA складається з різних типів шарів: згорткових шарів, субдискретизації (субвибір), шарів і шарів "нормальної" нейронної мережі - персептикона (рисунок 1).



Рису. 1 Архітектура нейронної мережі Coertnaya

Перші два типи шарів (згортання, підвибірка), чергуються між собою, утворюють вхідний вектор знаків для багатошарового фартектора.

Архітектура нейронної мережі чітко передбачає отримання зображень на вході, що дозволяє враховувати певні властивості введення в саму мережеву архітектуру. Ці властивості дозволяють ефективніше реалізувати функцію прямого розподілу і значно зменшити загальну кількість параметрів в мережі.

1.1 Згортковий шар

Мережа названа на честь операції - комплект, вона часто використовується для обробки зображень і може бути описана за наступною формулою:

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l] * g[k,l]$$

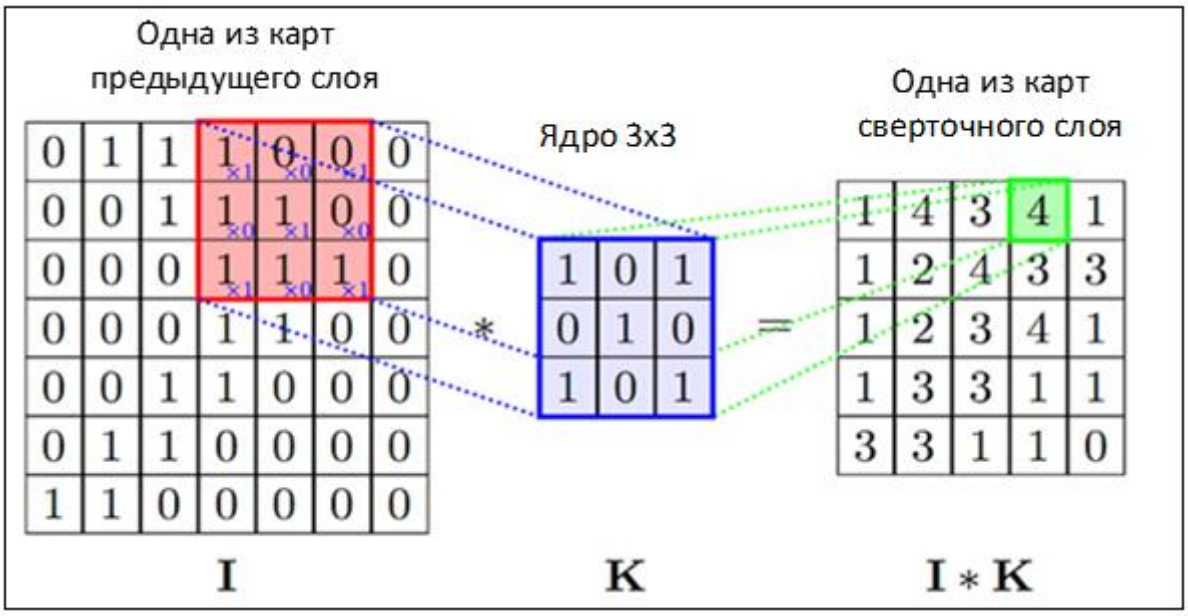
Для формування вихідної матриці необхідно нормалізувати піксельні значення зображення в діапазоні від 0 до 1, за формулою:

$$f(p, min, max) = \frac{p - min}{max - min}$$

Де f є функцією нормалізації, p — значення кольору пікселя від 0 до 255, min — мінімальне значення пікселя 0, максимальне — максимальне значення пікселя 255.

Після нормалізації отриманий масив подається на вхід свердлильним шаром. Цей шар, в свою чергу, має своє ядро (фільтр). Ядром єматриця, розмір якої зазвичай становить 3 x 3, 5 x 5, 7 x 7. Також розмір ядра вибирається таким чином, щоб розмір карти валків був рівномірним, це дозволяє не втрачати інформацію при зниженні розмірності в шарі підвибірки.

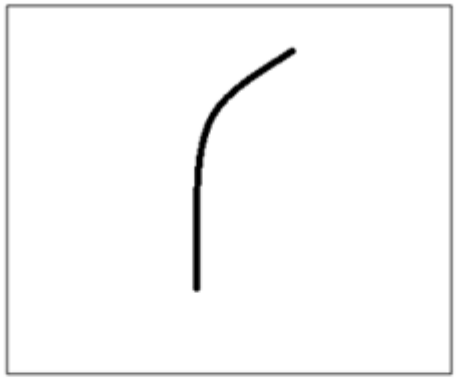
Це ядро ковзає по масиву вхідних значень і обчислює новий масив на основі цих даних, підсумовуючи елементне множення ядра ролу з поточною врахованою областю вводу (рисунок 2).



Рису. 2 Спільна робота

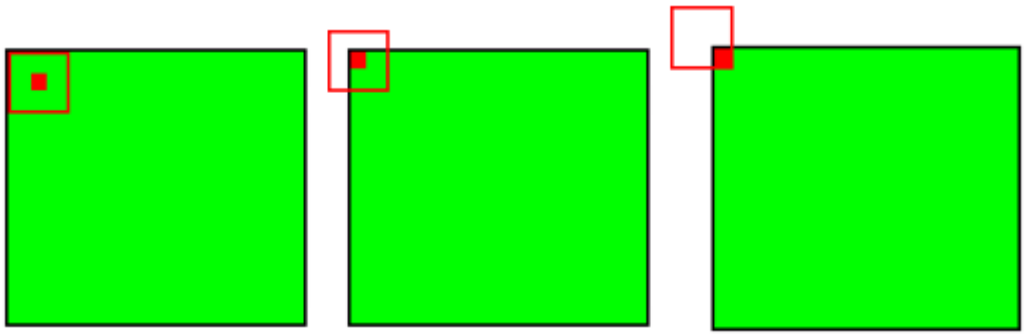
Наприклад, якщо мережа тренувалася на декількох обличчях, одне з ядер могло б виробляти найбільший сигнал воці, роті, бровах абоносі під час тренувань, інше ядровиявило б інші риси (рисунок 3).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0



Рису. 3 Зразок ядра з навченою ознакою

Також варто відзначити, що в залежності від способу обробки країв оригінальної матриці результат може бути менше вихідного зображення (дійсне), однакового розміру (того ж) або більшого (повного) (рисунок 4).



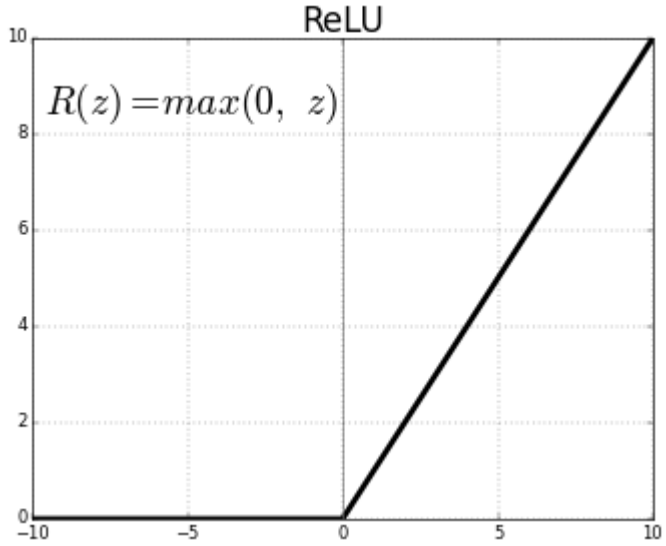
Рису. 4 Можливі методи обробки країв вхідного масиву

1.2 Рівень активації

Для того, щоб навчання масштабів ядер було ефективним, в результати пучками слід ввійти в нелінійність.

Зактивацією лона є певною функцією, яка застосовується до кожної кількості вхідних зображень.

Я вибрав функцію активації ReLU.



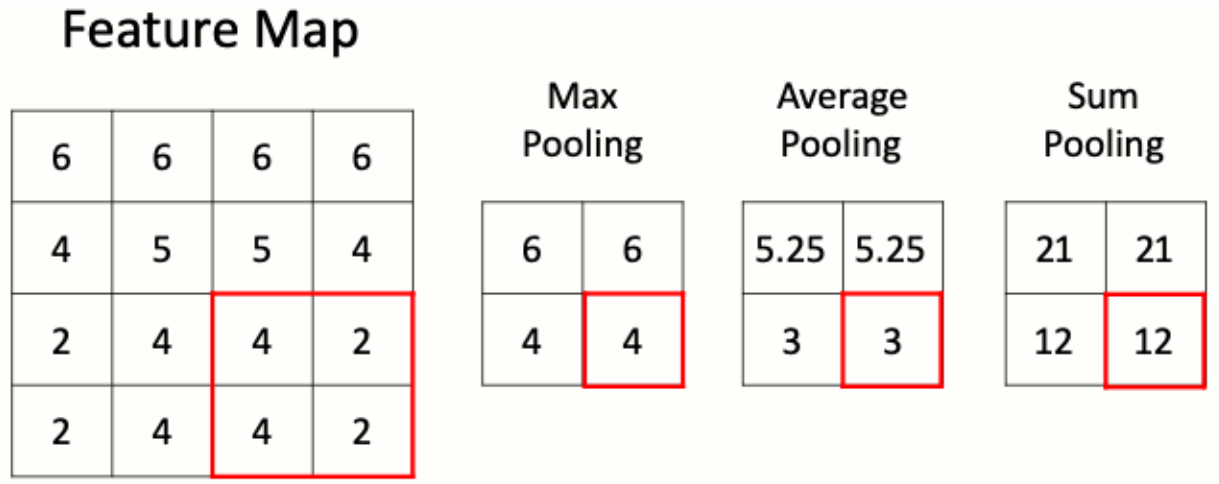
Рису. 5 Функція активації ReLU

Сутьданої операції полягає в операції відрізання від'ємної частини скалярного значення.

Станом на 2017 рік, ця функція та її модифікації (Noisy ReLU, Leaky ReLU та інші) є найбільш часто використовуваними функціями активації в глибоких нейронних мережах, особливо в згорнутих мережах.

1.3 Підвибірний шар(Sub - вибірка)

Цей шар зменшує простір функцій, зберігаючи найважливішу інформацію.



Рису. 6 Версії шару виділення

Метою шару є зменшення розмірності масиву попереднього шару. Якщо в попередній операції зведення вже були виявлені якісь ознаки, то для подальшої обробки такого детального зображення більше не потрібно, і воно ущільнюється до менш детального.

Під час процесу сканування ядро підвибірки не перетинаються на відміну від шару, що руйнується. Як правило, кожна карта має ядро 2x2, що дозволяє скоротити попередні рулонні карти в 2 рази. Вся карта функцій ділиться на 2 x2 клітини, з яких вибрано максимальне значення.

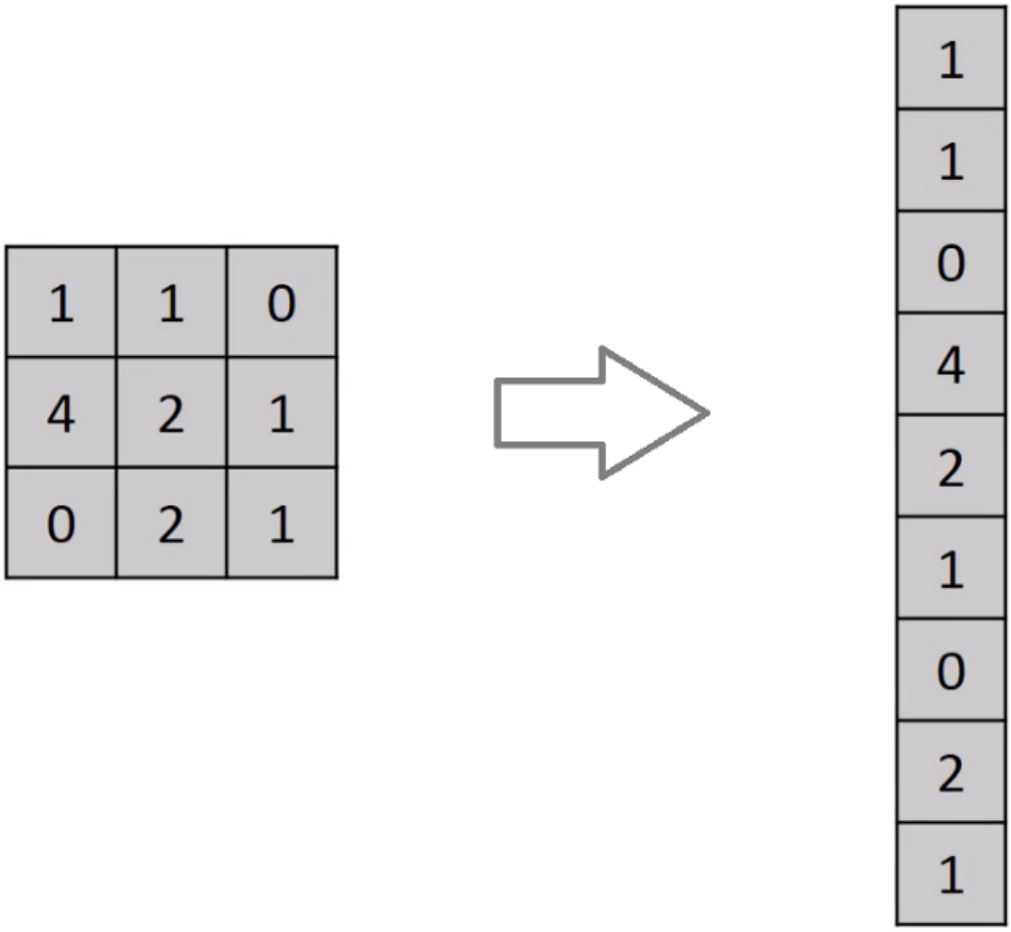
Формально шар можна описати за формулою:

$$x^l = a^l * subsample(x^{l-1}) + b^l$$

Де x^l - вихід l шару, l, b - коефіцієнтів зсуву шарів l, subsample- це операція відбору проб локальних максимумів.

1.4 Повністю зв'язаний шар

Після підвибірного шару нейрони подаються в плоский шар, суть якого полягає в перетворенні двовимірної матриці рис в вектор "8", якийможнаподається в повно зв'язуючий шар (рисунок 7).

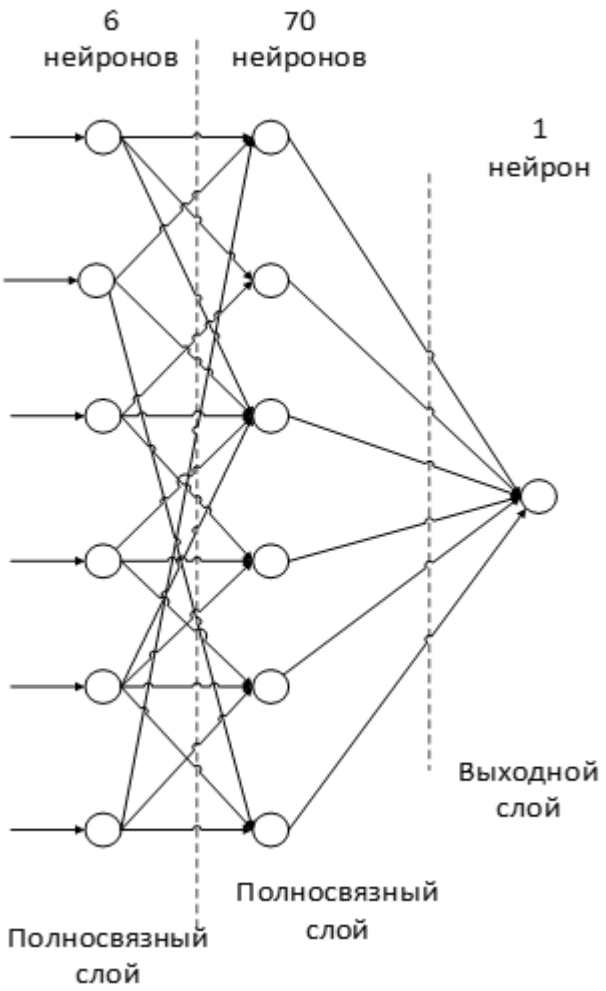


Рису. 7 Плоский шар

Різні нейронні мережі можуть виступати в якості повно з'єднаного шару, який буде класифікований за вектором вхідних параметрів. У моїй програмі цю роль виконує багатошаровий перцептрон.

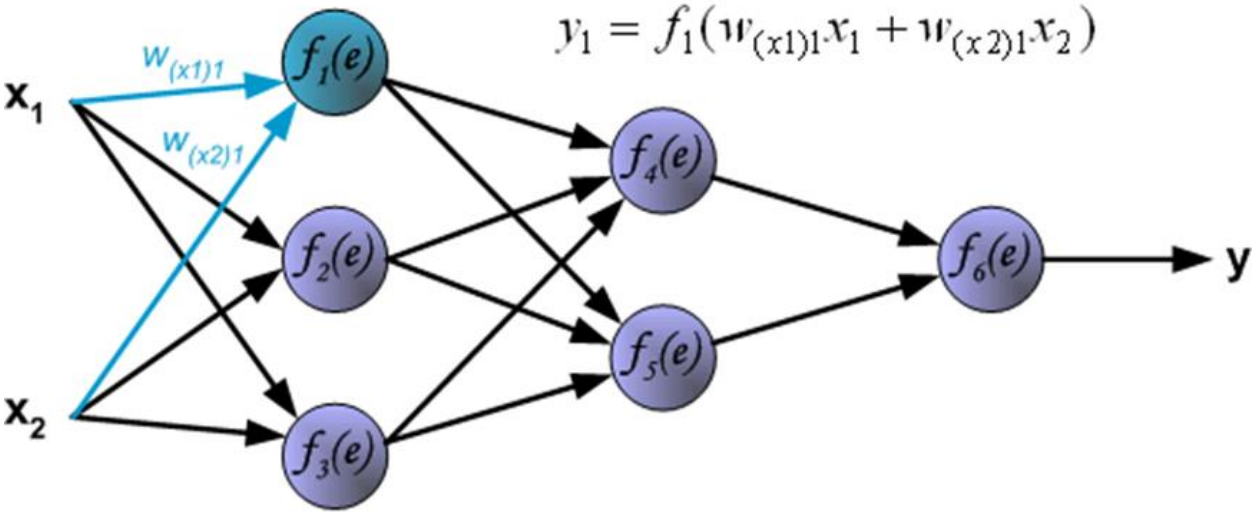
Цей шар м єскладною нелінійної функції, оптимізацією якої покращується якість розпізнавання.

На цьому шарі всі нейрони вхідного шару з'єднані синапсами з усіма нейронами першого прихованого шару, які в свою чергу з'єднані з усіма нейронами вихідного шару (рисунок 8).



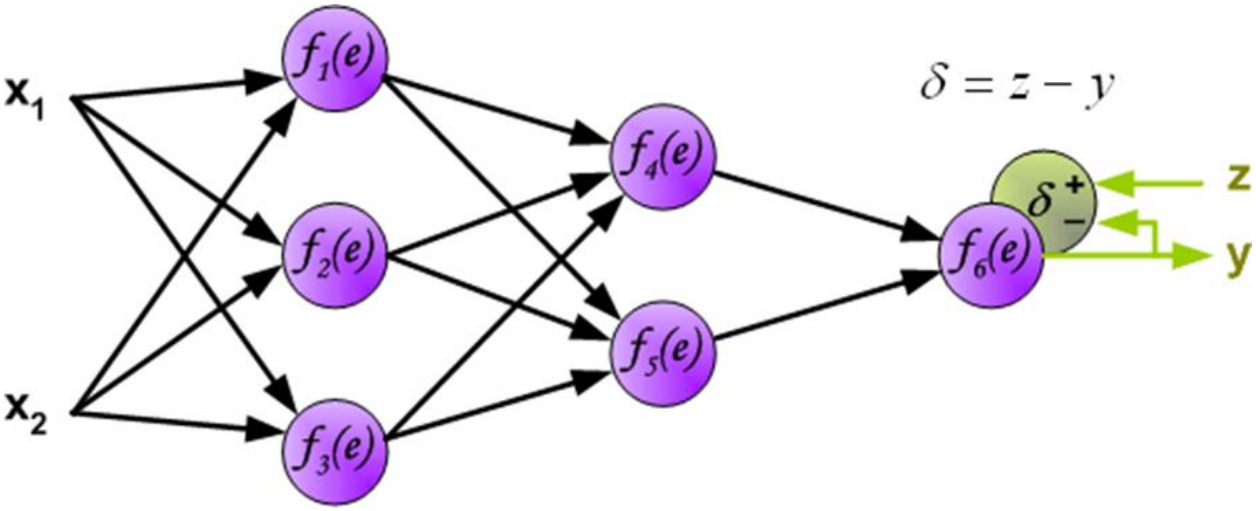
Рису. 8 Топологія MLP

Для всіх нейронів на прихованих шарах їх зважена кількість розраховується шляхом множення значення вхідного значення синапсу на нейрон попереднього шару, який пов'язаний з цим синапсом з нейроном прихованого шару, і замінюється у функції активації (рисунок 9).



Рису. 9 Розрахунок значення нейрона на прихованому шарі

Після розрахунку значення нейрона на вихідному шарі похибка розраховується шляхом віднімання від очікуваного результату (рисунок 10).

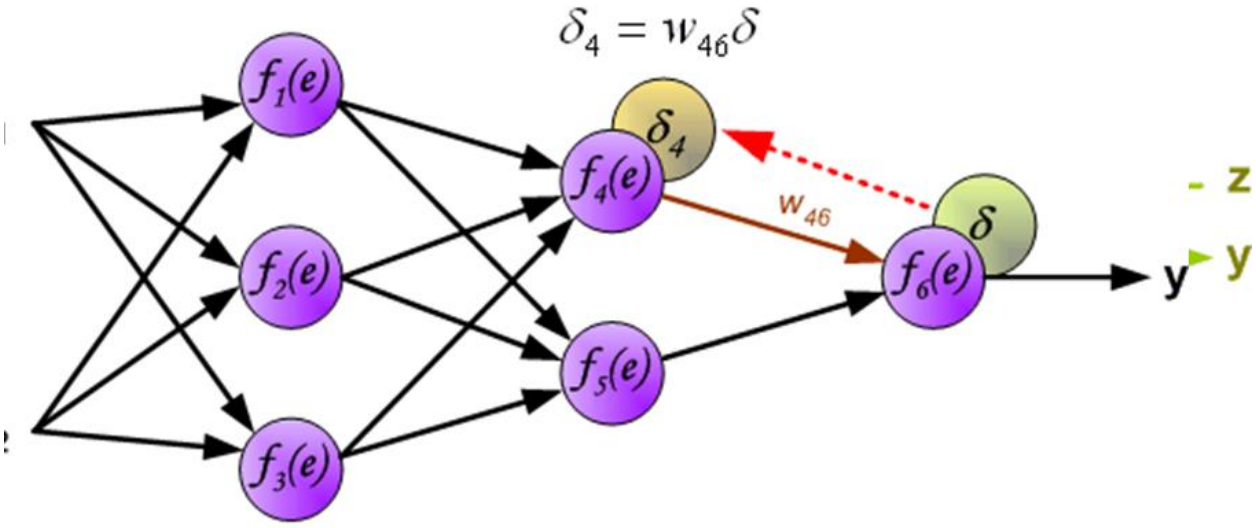


Рису. 10 Обчислення помилки

Наступне завдання - поширення похибки по всьому шару, а також по шарах підвибірки і рулону, при цьому виправляє синапси повно зв'язаного шару і ядра згорнутогошару.

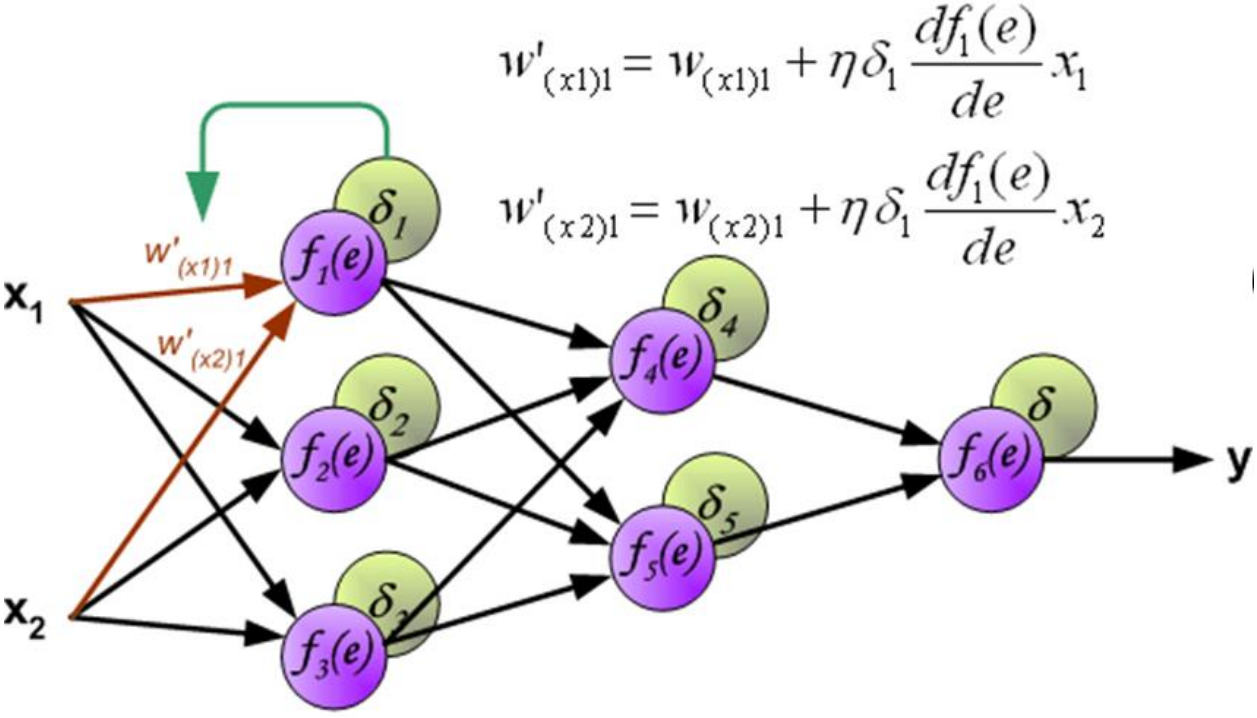
Поширення клопа для нейронів прихованого шару полягає в множенні похибки, отриманої на вихідному шарі, на вагу синапса, який зв'язує цей

нейрон з нейроном вихідного шару (рисунок 11).



Рису. 11 Розрахунок похибки на прихованому шарі

Після того, як похибка поширюється на всі нейрони в повному шарі, синапси коригуються (рисунок 12).

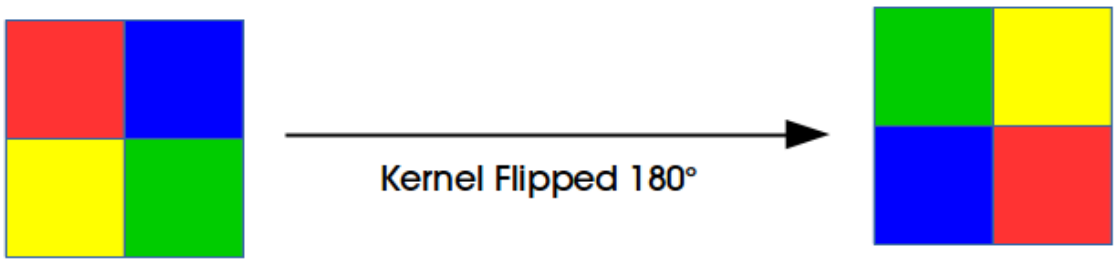


Рису. 12 Регулювання повношарових синапсів

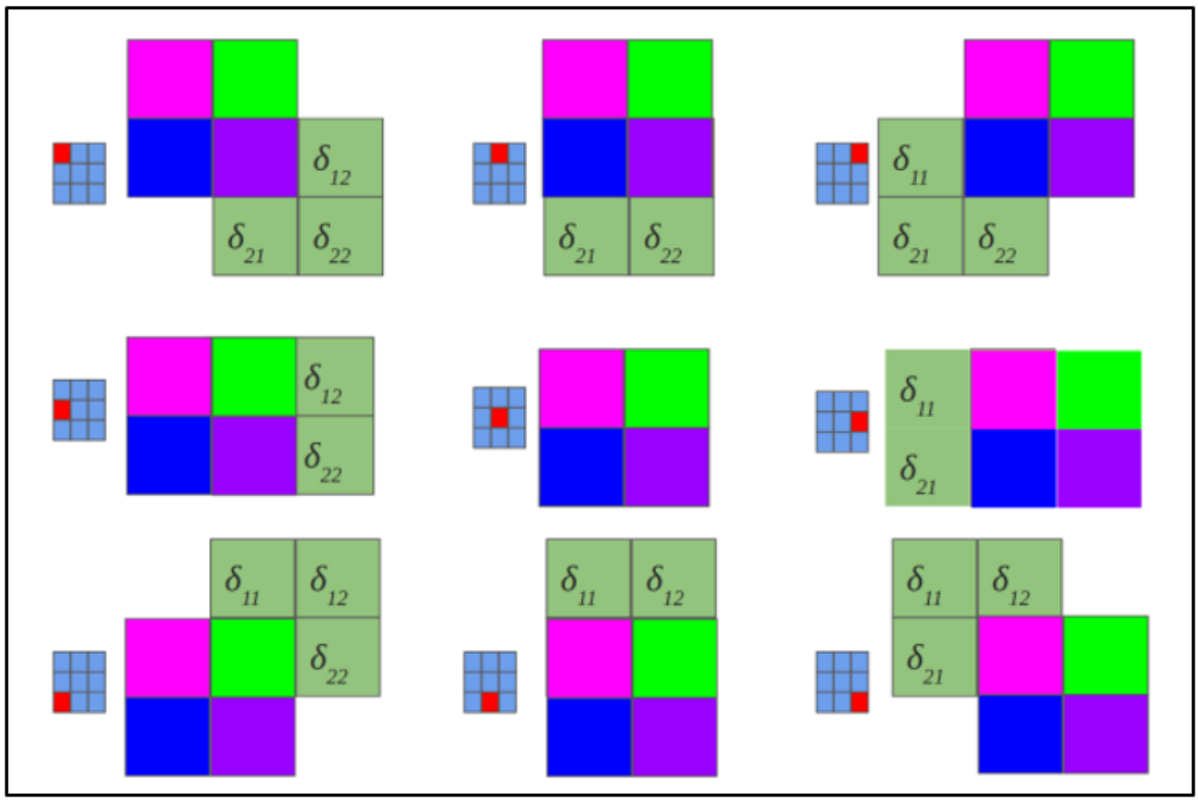
1.5 Обчислення похибки на підвиконно-виборчому шарі

Розрахунок похибки на шарі виділення з'являється в декількох варіантах. Перший випадок, коли шар підвибірки знаходиться перед повнокаранним, то він має нейрони і з'єднання того ж типу, що і в повно з'єднаному шарі, відповідно розрахунок δ похибки ніяк не відрізняється від розрахунку δ шару. Другий випадок, коли шар підвибірки знаходиться перед зведенням, розрахунок δ відбувається шляхом зворотного зведення.

Зворотний рулон - це той же спосіб розрахунку дельти, який полягає в повороті ядра на 180 градусів і ковзанні процесу сканування згорнутої карти зі зміненими краєтворчими ефектами. Тобто, необхідно взятисерцевину рулонної карти (слідом за шаром підвибірки), щоб повернути її на 180 градусів (рисунок 13) і зробити нормальний рулон на раніше розрахованих дельтах колідингної карти (рисунок 14),але щоб вікно сканування визирнула з карти.



Рису. 13 Поверніть ядро на 180 градусів



Рису. 14 Обчислення δ підвиду за допомогою зворотного рулону

1.6 Обчислення похибки на згортковому шарі

При прямому розподілі сигнальних нейронів підвибірного шару утворювалося невідкрите вікно сканування на буровому шарі, в процесі якого відбирали нейрони з максимальним значенням, при зворотному розподілі відбувається піднесення похибки до раніше обраного максимального нейрона, а решта отримують нульову дельту похибки.

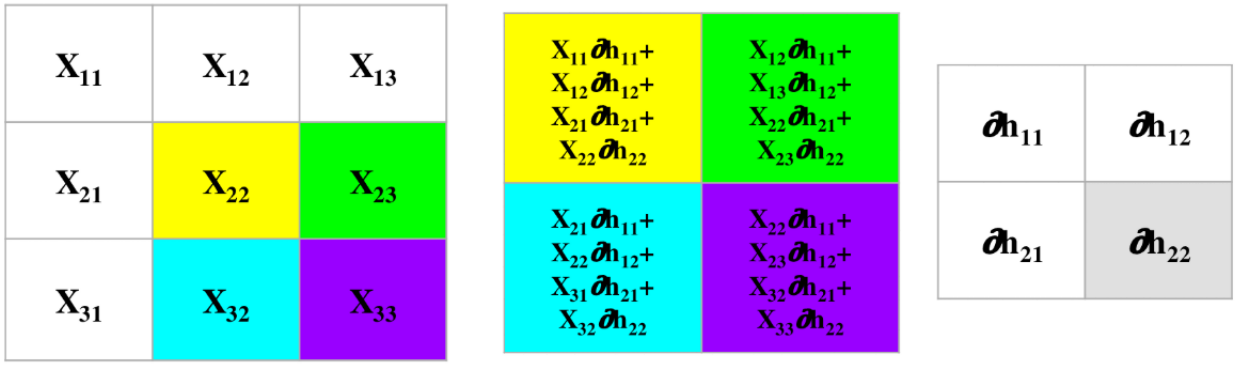
1.7 Налаштування фільтрів зведення

Щоб налаштувати значення фільтра, використовується формула:

$$W = W - \alpha \frac{\partial L}{\partial W}$$

Де W - значення елемента в ядрі - швидкість αнавчання, ∂L/∂W є похідною функції втрати W.

Розрахунок похідної функції W втрати показаний на рис. 15.



Рису. 15 Обчислення функції втрати W

З рису. 15 можна зрозуміти, що матриця X - це масив входів при прямому розподілі сигналу, а ∂h - масив помилок наступного шару.

1.8 Приклад навчання

Навчальний зразок складається з позитивних і негативних прикладів. При цьому осіб і "не осіб".

База даних LFW3D використовувалася як позитивний зразок навчання. Він містить кольорові зображення лобових граней (рисунок 16) типу JPEG, розміром 90x90 пікселів. База даних надається FTP, доступ до них здійснюється парою. Щоб отримати пароль, потрібно заповнити просту форму на головній сторінці сайту, де вказати своє ім'я та адресу електронної пошти.

Рисунок 16 Тренування обличчя

Оскільки негативні приклади навчання (рисунок 17) заснована на базі даних sun2012, вона містить величезну кількість різних сцен,які класифікуються.

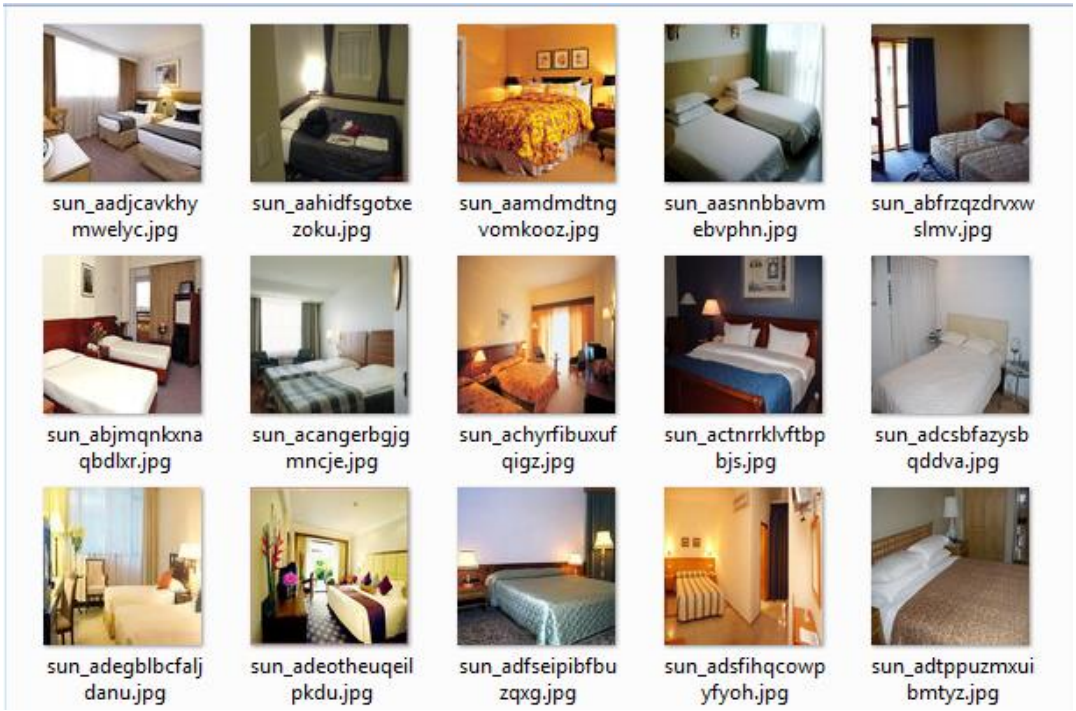
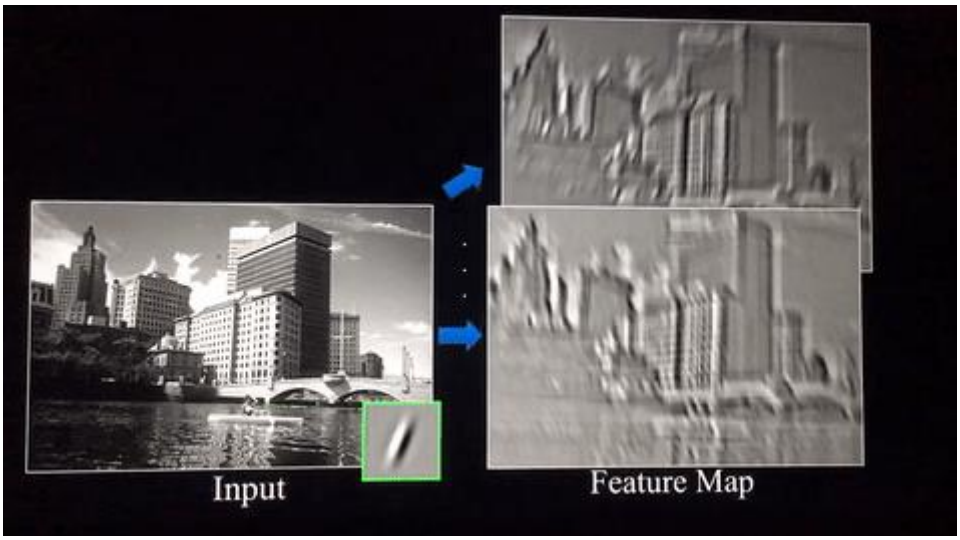


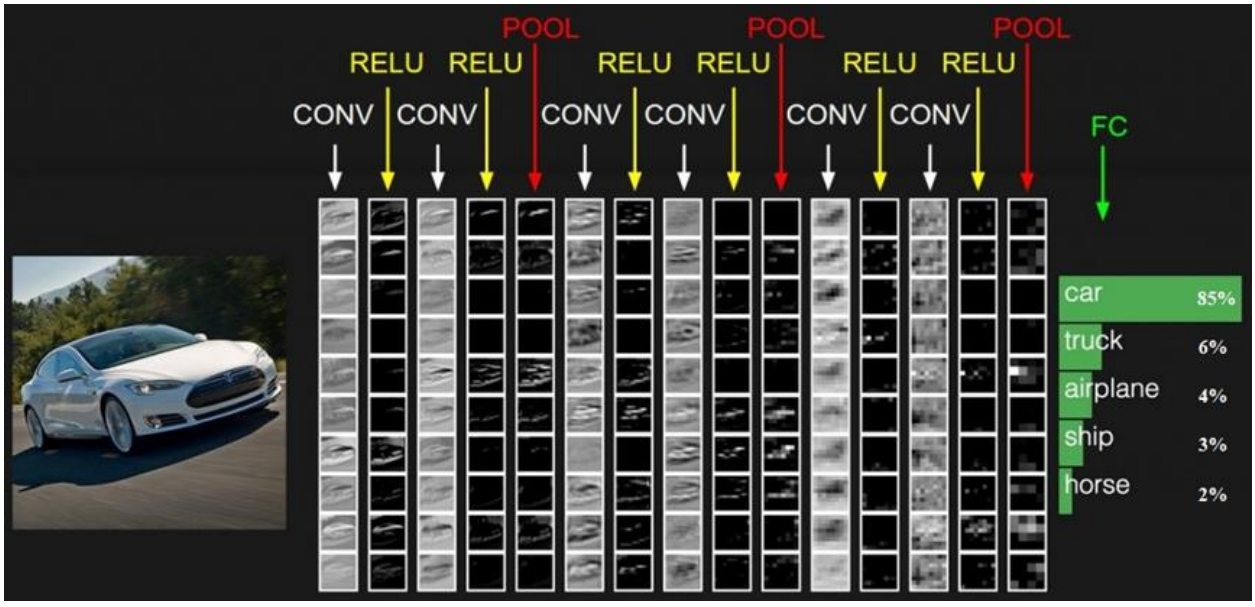
Рисунок 17 Зразок навчання без осіб

1.9 Підходи до візуалізації

Щоб візуалізувати роботу іскрометних нейронних мереж, було вирішено поступово продемонструвати користувачеві, як змінюються зображення при проходженні не тільки через шари комплекту (рисунок 18), але і шари підвибіру та активації (рисунок 19).



Рису. 18 Візуалізація процесу зведення



Рису. 19 Візуалізація всіх шарів в зв'язку нейронної мережі

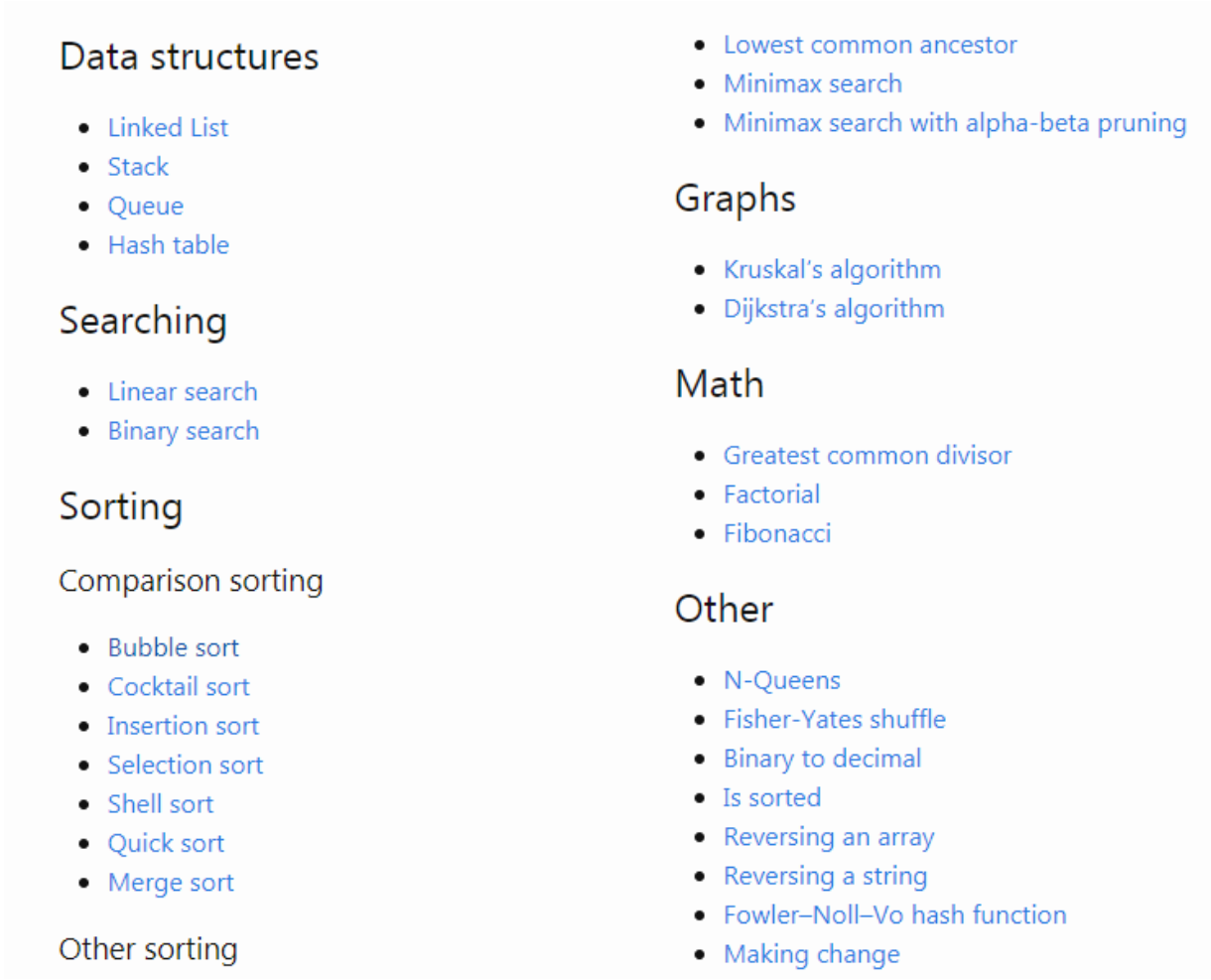
2 ОГЛЯД ТА АНАЛІЗ ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

У цьому розділі будуть розроблені існуючі програмні рішення для візуалізації алгоритмів, і на цій основі будуть створені вимоги до розробленого програмного забезпечення.

1. Альгорітм Вікі

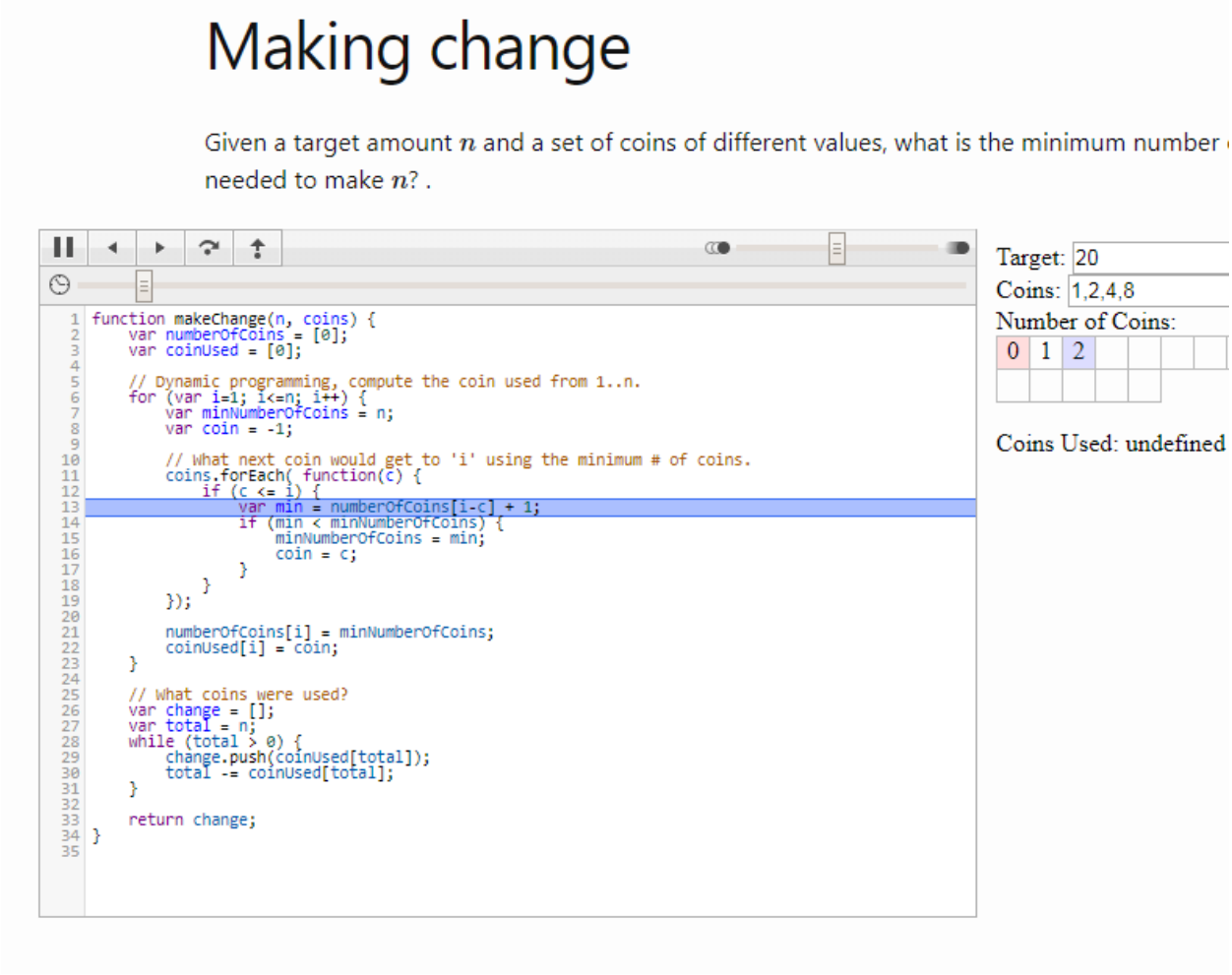
Ця платформа є експериментом зі створення інтерактивних алгоритмів в Інтернеті.

Ця платформа пропонує близько 40 алгоритмів,які можна візуалізувати(рисунок. 20).



Рису. 20 алгоритмів для вікі алгоритму візуалізації

До переваг цієї платформи можна віднести той факт, що при виконанні візуалізації є можливість відслідковувати дії алгоритму за допомогою коду, який розташований поруч з вікном візуалізації (рисунок 21). Ви можете зупинити програму, "перемотати" її або вперед. Крім того, деякі алгоритми надаються у відкритому на GitHub. Кожен алгоритм має короткий опис.



Рису. 21 Візуалізація одного з алгоритмів

Недоліки цієї платформи в тому, що деякі алгоритми не працюють, доступна мова тільки англійська , візуалізація деяких алгоритмів, передбачених в текстовій формі, без візуалізації як такої. Загаломз 40 візуалізованих алгоритмів користувач отримує можливість візуалізувати лише меншу їх частину.

2. Візуалізатор алгоритмів

Інтерактивна онлайн-платформа, яка візуалізує алгоритми за кодом.

Містить багато алгоритмів, які можна візуалізувати (рисунок 22).

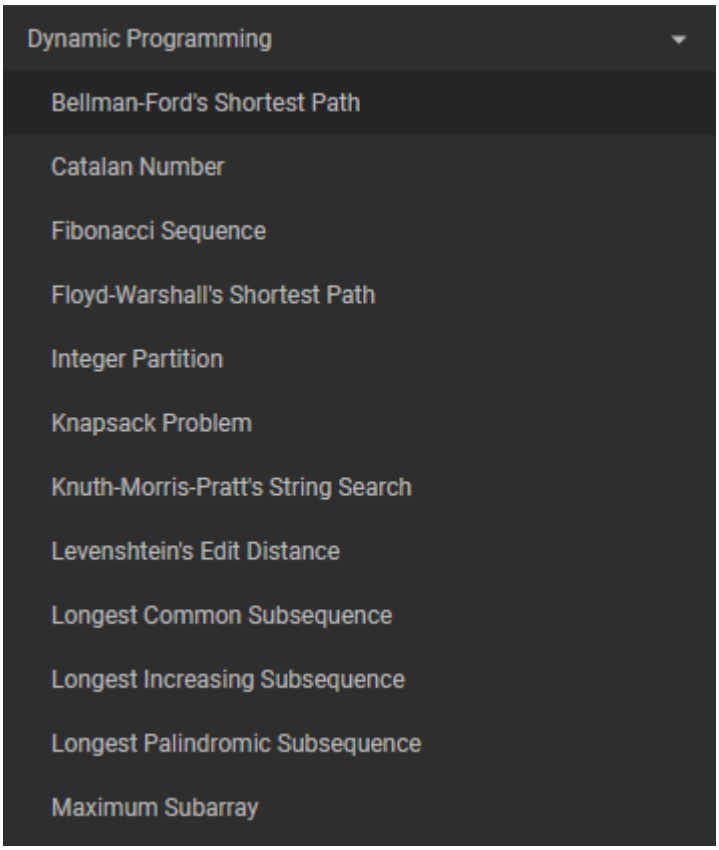


Рисунок 22 Частина реалізованих алгоритмів

Головною перевагою, крім забезпечення програмного забезпечення з відкритим вихідним кодом (рисунок 23), є можливістьзмінибудь-яких значень і ручокв кодовоїпрограмі, як перед процесом візуалізації, так і в процесі візуалізації, які відображаються в процесі програми.

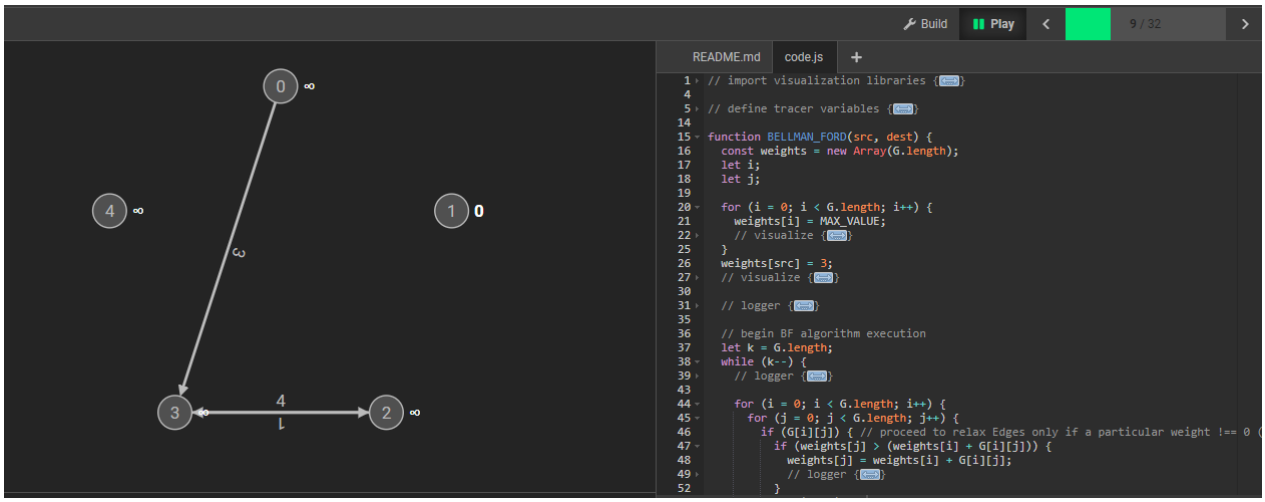


Рисунок 23 Графічнийtracer візуалізації

Як і в першому розглянутому програмному рішенні, є можливість збільшення швидкості процесу візуалізації, а також можливість зупинити виконання алгоритму. Також до кожного алгоритму додається анімована картина методу виконання (рисунок 24),крім надання теодоруетичної інформації про його роботу.

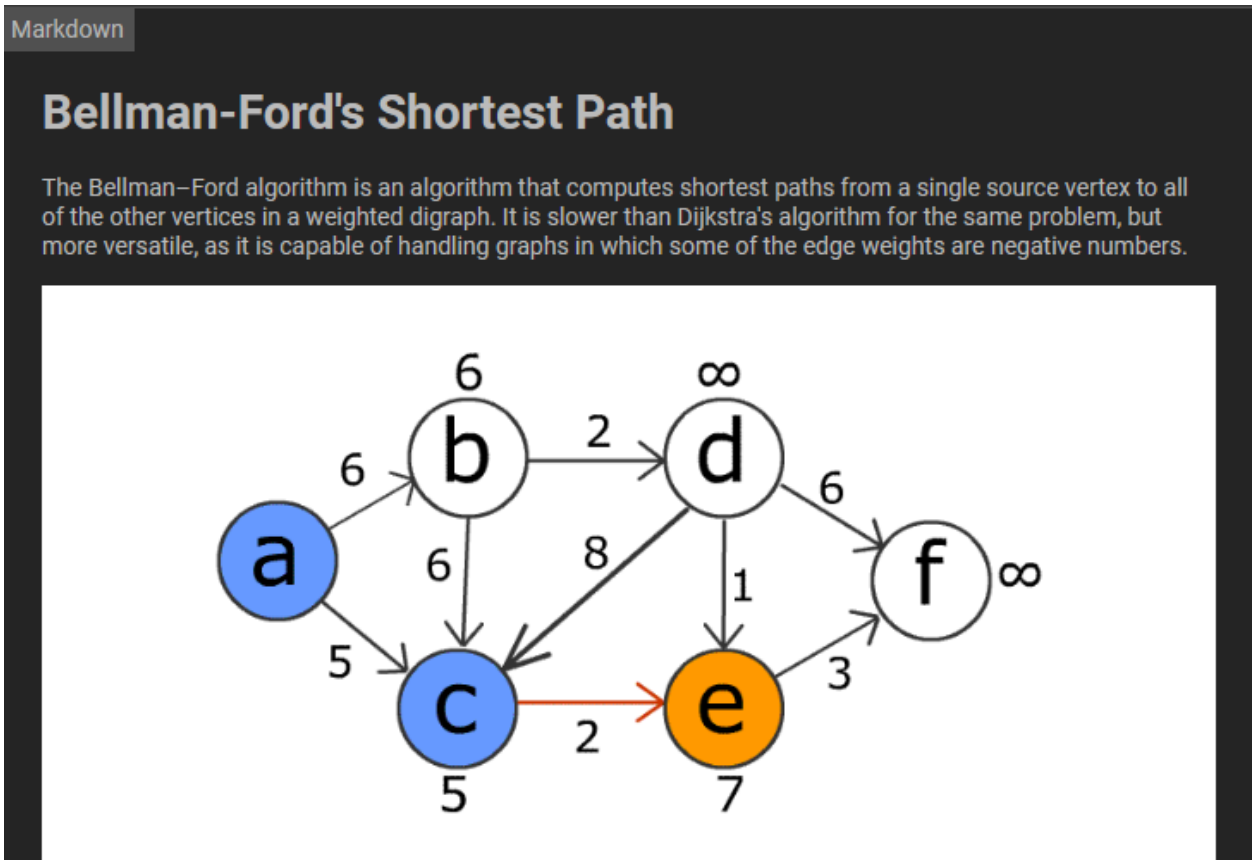


Рисунок 24 Теоретична інформація про роботу методу

Крім того, ця платформа надає API для розробників.

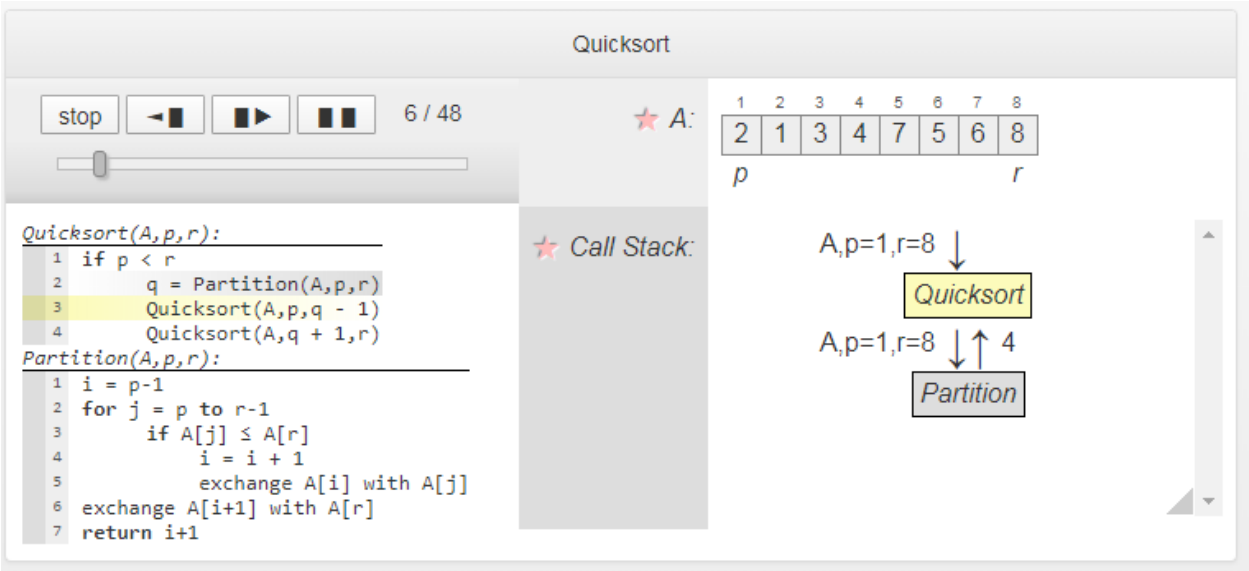
Одним з недоліків цієї платформи є те, що, маючи пристойну базу даних інформації, вона залишається орієнтованою на англомовних користувачів, оскільки немає можливості перекладу на будь-яку іншу мову.

3. Вамонос (ів)

Ця платформа призначена для динамічної візуалізації в браузері. Містить невелику кількість реалізованих алгоритмів (рисунок 25).

Рису. 25 Реалізовані алгоритми у Vamonos

Платформа не містить короткого опису методів, але має достатню функціональність для реалізації візуального представлення алгоритмів (рисунок 26).



Рису. 26 Алгоритм візуалізації у місті Вамонос

Користувачеві надається можливість редагувати вихіднідані, переглядати код виконання, припиняти запуск або прискорення.

3 ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, ЩО РОЗРОБЛЯЄТЬСЯ

Цей розділ, заснований на аналізі існуючих програмних рішень, представить вимоги до функціональності програми та інтерфейсу користувача.

Функціональні вимоги:

- Нейронна мережа повинна мати можливість вчитися на навчальному зразку зображень;
- Програма повинна класифікувати зображення на основі навченої мережі.
- Програма повинна мати покрокову візуалізацію результату кожного з шарів, що складають нейронну мережу;
- Користувач повинен мати можливість змінювати вагу ядер шару.

Вимоги до інтерфейсу користувача:

- В окремому вікні повинен бути реалізований алгоритм для візуалізації рулону, підвибірки та активації користувацької архітектури CNN.
- Навігаційні елементи інтерфейсу програми повинні забезпечити чітке розуміння користувачем їх значення і переконатися, що всі розділи програми доступні користувачеві і відображати відповідну інформацію;
- Інтерфейс програми повинен забезпечити візуальний, інтуїтивно зрозумілий погляд на структуру розміщеної інформації, швидкий і логічний перехід до відповідних розділів програми.
- Системний інтерфейс повинен бути розроблений для користувачів, які не мають технічного досвіду і навичок в комп'ютерних технологіях, і легко освоїти.

4 ОПИС КЛАСІВ ТА КОРИСТУВАЛЬНИЦЬКИХ МЕТОДІВ

Для реалізації всіх запланованих функцій і алгоритмів були необхідні наступні класи і методи:

Клас нейронів

Цей клас необхідний для формування вхідного вектора для перенесення його на повноприв'язний шар, а також для поширення помилки на всі існуючі шари зв'язаної нейронної мережі.

Цей клас має властивості: порядок - число нейрона в шарі, weightSum - зважена кількість нейрона, ActivationSum - активований стан нейрона, Помилка - помилка нейрона.

Навчальний image клас

Клас, призначений для об'єднання зображення та його мітки.

Властивості цього класу uri - розташування зображення та класу - очікуваний клас.

Ваговий клас

Клас необхідний для структурування луски (синапсів) повністю зв'язаного шару.

Він має властивості виходу - кількість нейронів, з яких бере початок синапс, вхідний - кількість нейронів, що приймає синапси, і значення - значення ваги синапса.

Клас ImageLoader

Цей клас необхідний для перетворення зображення в масив байтів *(метод LoadImage)* і побудови зображення на основі перенесеного масиву байтів *(BitmapFromByteArr).*

Клас шару

Цей клас необхідний для побудови структури складаної нейронної мережі при безпосереднього розподілу сигналу.

Містить властивості назви - ім'я шару, число - номер шару, startValues - колекція карт вводу, рушії - колекцію ядер поточного шару і кінцевих оцінок - вихідну карту.

Клас backLayers

Клас, призначений для зворотного виправлення помилки через шари комплекту, підвибіру та активації.

Має властивості назви - назва шару, номер - номер шару, startValues - колекція вхідних помилок, рушії - колекція ядер поточного шару і кінцевих оцінок - карти помилок вихідного шару.

Візуальний клас

Цей клас необхідний для організації візуалізації користувальницької нейронноїмережі.

Клас головного вікна

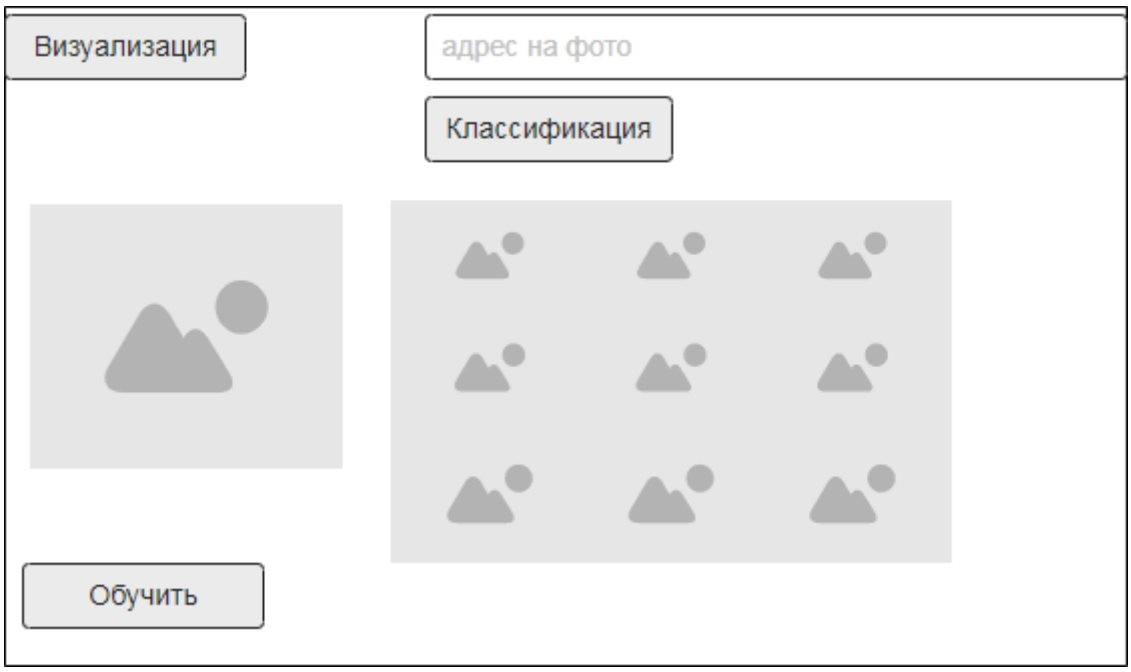
Базовий клас, в якому відбуваються всі розрахунки, має глобальне навчання Спеціалізовані параметри - швидкість навчання і countFilter - кількість фільтрів шару комплекту.

У даному класі описані основні методи повноцінної роботи складаної нейронної мережі: згортковий, метод, що реалізує вставку вхідного зображення з випадково призначеним ядром, згортковий WithoutRandom - реалізує рулон з попередньо встановленими ядрами, maxPooling - здійснює алгоритм підвибірного шару, в даному випадку Max Pooling, ReLu - шар активації нейронів після комплектації, нормалізаціяMLP - нормалізує вихідні значення шару підвибірки для подачі на повний зв'язуючий шар, backPropFlattedToCon backMaxPoolingToRelu - обчислює помилку рівня активації на основі помилки на підпунті -selection, findDerivativeEngine - обчислює градієнт функції втрати для ваг ядер рол-шару, getNewEngine обчислює нові значення ядер рулону, reverseConvolutional - виконує зворотний рулон.

5 РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

Для розробки користувальницького інтерфейсу потрібно враховувати всі плюси і мінуси програм, які були проаналізовані раніше. Крім того, ви повинні відповідати всім функціональним і вимогам інтерфейсу користувача, які також були виявлені раніше.

Відповідно до всіх раніше зазначених вимог, інтерфейс головного вікна(рис. 27).



Рису. 27 Інтерфейс головного вікна

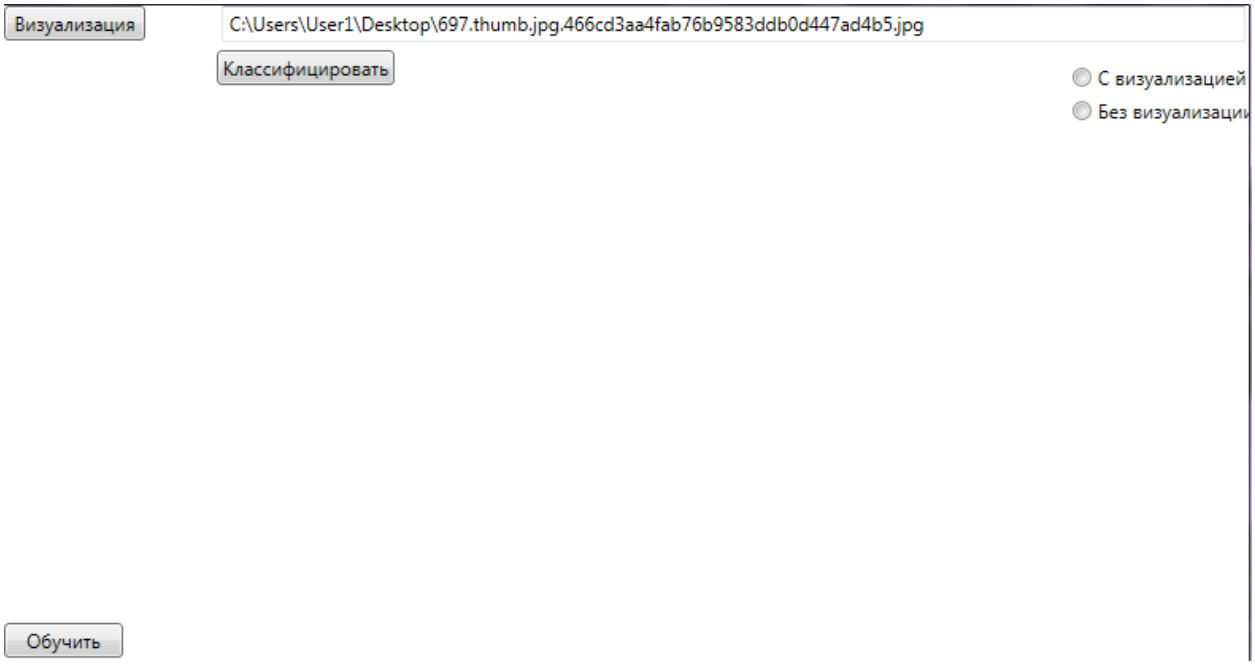
З цього вікна можна виконати всі основні функції цієї програми. Крім головного вікна для навчання і класифікації, необхідно створити інтерфейс для візуалізації складаної нейронної мережі (рисунок 28).

Рису. 29 Інтерфейс обробки зображень нейронної мережі

6 КЕРІВНИЦТВО КОРИСТУВАЧА

Це програмне забезпечення призначене для демонстрації того, як працює нейронна мережакомплекту.

При запуску програми відкривається головне вікно (рисунок. 30).

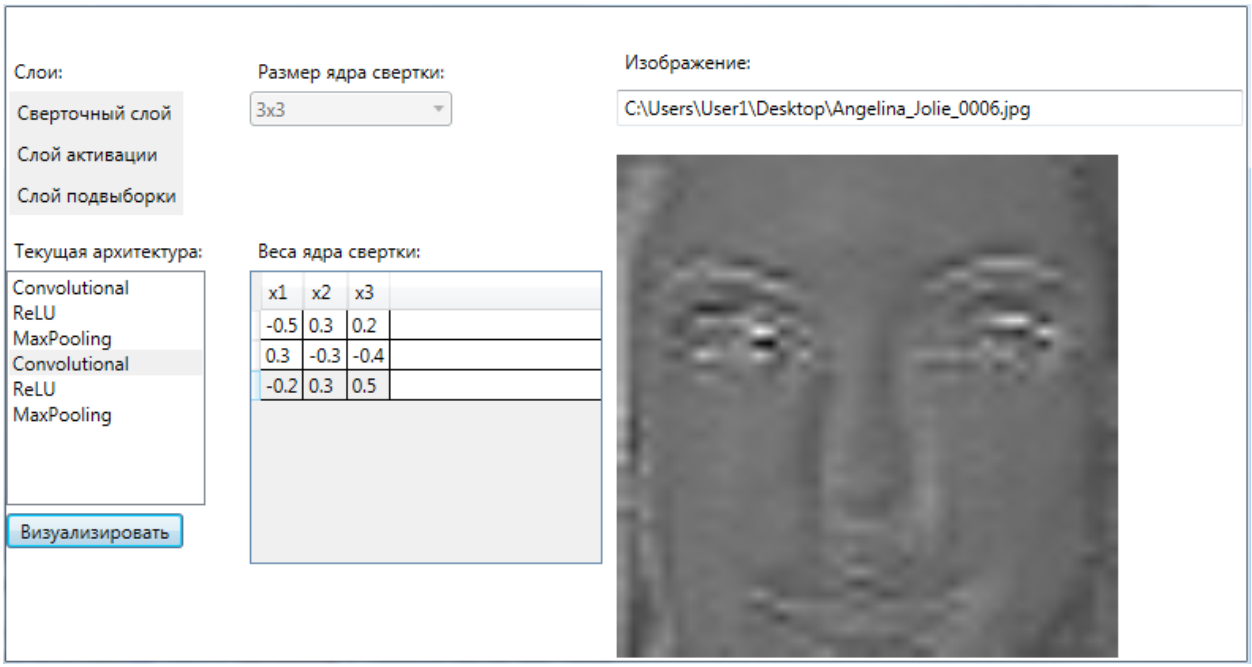


Рису. 30 Програма на головному екрані

Тренувати нейронну мережу потрібно, натиснувши кнопку "Навчити". Після успішного навчання нейронної мережі можна класифікувати різні зображення за допомогою покрокового перегляду обробки зображень кожним шаром (рисунок 31) або без (обраний один з параметрів "З візуалізацією" або "Без візуалізації")

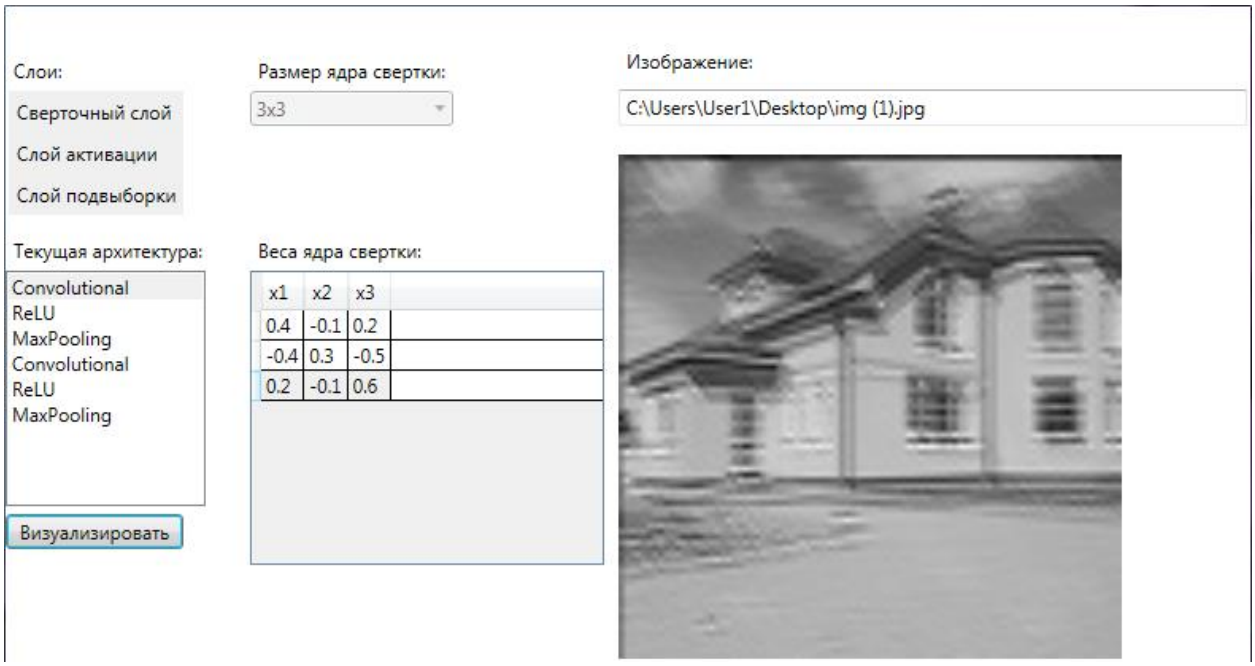
Рисунок.31 Обробка зображень рухомим шаром

Ви також можете створити власну архітектуру мережі та спостерігати, як зображення буде поетапно з кожним шаром (рисунок 32).



Рису. 32 CNN Користувальницькі роботи архітектури

У цьому вікні користувач може встановити свою архітектуру CNN, додавши три основних шари до потрібної послідовності: складіть, підвибір та активацію.



Рису. 33 CNN Користувальницькі роботи архітектури

ВИСНОВОК

Метою роботи було створення навчальної програми для демонстрації використання квадратичних ядер у згорткових нейронних мережах.

Ця програма має достатню функціональність, щоб продемонструвати взаємодію шарів нейронної мережі комплекту.

На додаток до використання цієї програми для демонстрації основних етапів в освітніх цілях, програма може бути використана для більш вузько орієнтованих завдань, таких як ідентифікація людей в потоковому відео, але це вимагає більшої кількості навчальних даних для підвищення обізнаності.

Таким чином, проект завершено, функціональність програми відповідає вимогам. Система займає мало місця, працює без затримок і готова до використання.

СПИСОК ВИКОРИСТОВУВАНИХ ЛІТЕРАТУР

1. Авраамian, А.В. Розробка інтерфейсу на основі системи Windows Presentation Foundation: підручник / А.В. Авраамian, М.Е. Авраамian. - Ростов-на-Дону, Таганрог : Видавництво Південного федерального університету, 2017. — 301 с. — ISBN 978-5-9275-2375-7. - Текст : Electronic / Електронна бібліотека системи IPR BOOKS : URL: http://www.iprbookshop.ru/87487.html (дата дзвінка:25.06.2020). - Режим доступу: для авторизації користувача.
2. CS231n: Нейронні мережі розпізнавання зображень :"Електронний ресурс" / Habr URL: https://habr.com/ru/post/456186/ (Дата перетворення: 25.06.2020)
3. Зворотне поширення в згорткових нейронних мережах - Інтуїція і код:qЕлектроннийресурс q / Середня URL-адреса: https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199 (датаадреси:25.06.2020)

4. Зворотнапагація згорткових нейронних мереж: від інтуїції до виведення: qЕлектроннийресурс // GRzegorz Gwardys URL: https://grzegorzgwardys.wordpress.com/2016/04/22/8/ (адреса дата:25.06.2020)
5. Горелов, С.В. Сучасна технологія програмування: розробка додатків Windows мовою S.E.C. У двох томах. Т.І.: підручник / С.В. Горелов; під редакцією П.В. Лук'янова. Москва – Прометей, 2019. — 362 с. — ISBN 978-5-907100-09-1. - Текст : Electronic / Електронна бібліотека системи IPR BOOKS : URL: http://www.iprbookshop.ru/94532.html (дата дзвінка: 29.06.2020).
6. Що таке рулонна нейронна мережа: "Електронний ресурс" / Habr URL: https://habr.com/ru/post/309508/ (Дата адреси: 25.06.2020)
7. Барський, АВ Логічні нейронні мережі / АВ Барських. Москва : Інтернет Університет інформаційних технологій (INTUIT), 2016. — 492 с. — ISBN 978-5-94774-646-4. - Текст : Electronic / Електронна бібліотека системи IPR BOOKS : URL: http://www.iprbookshop.ru/52220.html (дата дзвінка: 04.07.2020). Режим доступу: для автоістраторів. Користувачів
8. Сєдов, В.А. Введення в нейронні мережі : методичні інструкції до лабораторної роботи з дисципліни "Нейроінформатика" для студентів спеціальності 09.03.02 "Інформаційні системи та технології" / В.А. Сєдов, Н.А. Сєдова. - Саратов : Ai Pi Air Media, 2018. — 30 с. — ISBN 978-5-4486-0047-0. - Текст : Electronic / Електронна бібліотека системи IPR BOOKS : URL: http://www.iprbookshop.ru/69319.html (адреса: 04.07.2020). Режим доступу: для автоістраторів. Користувачів
9. Громадянин, Е.І. Нейронні мережі : навчальний посібник / Е.І. Горожаніна. Самара: Волзький державний університет телекомунікацій та інформатики, 2017. — 84 с. — ISBN 2227-8397. -

Текст : Electronic / Електронна бібліотека системи IPR BOOKS : URL: http://www.iprbookshop.ru/75391.html (дата дзвінка: 04.07.2020). Режим доступу: для автоістраторів. Користувачів

ДОДАТОК 1 ПРОГРАМА ЛІСТИНГУ

1 Шар *буріння*

```
загальнодоступний список<подвійний[,]> ConvolutionalWithoutRandom(Список<подвійний[,]>
матриця,Список<подвійний [,]> eng)//////////////++++++++++++
      {
Список<подвійний[,]> res = новий список<подвійний[,]>();
         if (count == 0)
         {
У 1980-х і 1990-х < 1990-х і 1990-х < 1990-х років. Кількість; f++)
Для (int p = 0; p < eng. Кількість; p++)
               {
double[,] newMatrix = новий подвійний[матриця[f]. GetUpperBound(0) + 3, матриця[f]. GetUpperBound(1) + 3];
для (int i = 0; i < newMatrix.GetUpperBound(0) + 1; i++)
для (int j = 0; j < newMatrix.GetUpperBound(1) + 1; j++)
                        newMatrix[i, j] = 0;
Для (int i = 0; i < матриця[f]. GetUpperBound(0) + 1; i++)
для (int j = 0; j < матриця[f]. GetUpperBound(1) + 1; j++)
                        newMatrix[i + 1, j + 1] = matrix[f][i, j];
                  У 1990-1990-1990-199
подвійний[,] двигун = новий двомісний[3, 3];
                  двигун = eng[p];
double[,] nextLayout = новий подвійний[(newMatrix.GetUpperBound(0) + 1) - (двигун. GetUpperBound(1) + 1) + 1,
(newMatrix.GetUpperBound(1) + 1) - (двигун. GetUpperBound(1) + 1) + 1];
                  Int curJ = 1;
                  int curI = 1;
for (int z = 0; z < nextLayout.GetUpperBound(0) + 1; z++)
для (int w = 0; w < nextLayout.GetUpperBound(1) + 1; w++)
for (int i = curI; i < newMatrix.GetUpperBound(0); i++)
для (int j = curJ; j < newMatrix.GetUpperBound(1); j++)
                           {
                              подвійна сума = 0;
Для (int k = 1; k < двигуна. GetUpperBound(0) + 1; k++)
для (int l = 1; l < двигуна. GetUpperBound(1) + 1; l++)
                                 {
                                    sum = newMatrix[i - 1, j - 1] * двигун[k - 1, l - 1] + newMatrix[i - 1, j] * двигун[k - 1, l] +
newMatrix[i - 1, j + 1] * двигун[k - 1, l + 1] + +
                                       newMatrix[i, j - 1] * двигун[k, l - 1] + newMatrix[i, j] * двигун[k, l] + newMatrix[i, j +
1] * двигун[k, l + 1] +
                                       newMatrix[i + 1, j - 1] * двигун[k + 1, l - 1] + newMatrix[i + 1, j] * двигун[k + 1, l] +
newMatrix[i + 1, j + 1] * двигун[k + 1, l + 1];
                                    k = рушій. GetUpperBound(0) + 1;
                                    l = двигун. GetUpperBound(1) + 1;
                                 }
                              if (подвійний. Чи єнегативний (сума) || Подвійний. IsPositiveInfinity(сума)|| Подвійний.
IsInfinity(сума))
                              { sum = 0; }
                              nextLayout[z, w] = sum;
                              Int curW = w;
                              curW++;
                              if (curW < nextLayout.GetUpperBound(1) + 1)
                              {
                                 w++; curJ++;
                              }
                              else { curJ = 1; curI++; w = 0; z++; }
                           }
                  Res. Додати(наступнийlayout);
               }
            повернення res;
```

```
        }
```

2 Зворотний рулон

```
загальнодоступний список<подвійний[,]> Зворотнийконволюціальний(Список<Неврон[,]> матриця,
Список<подвійний [,]> eng)
      {//double[,] op = новий двомісний[,] { 0,2, 0,4, 0,6, 0,9 }, { 0,5, 0,8, 0,7, 0,7 }, { 0,1, 0,2, -0,4, 0,6 }, { 0,5, 0,3, 0,7,
0,9 } };
        double[,] yadro = новий двомісний[,] { 0,1, 0,03, -0,2, 0,15 }, { 0,15, -0,05, 0,25, 0,001 }, { 0,005, 0,3, -0,2, 0,2
}, { -0,35, 0,4, 0,15, 0,3 } };
Список<подвійний[,]> res = новий список<подвійний[,]>();
        double[,] pi = ReLU(op);
        if (матриця. Кількість == eng. Кількість)
        {
Для (int k = 0; k < матриця. Кількість; k++)
          {
double[,] newMatrix = новий подвійний[матриця[k]. GetUpperBound(0) + 3, матриця[k]. GetUpperBound(1) + 3];
для (int i = 0; i < newMatrix.GetUpperBound(0) + 1; i++)
для (int j = 0; j < newMatrix.GetUpperBound(1) + 1; j++)
                newMatrix[i, j] = 0;
Для (int i = 0; i < матриця[k]. GetUpperBound(0) + 1; i++)
for (int j = 0; j < матриця[k]. GetUpperBound(1) + 1; j++)
                newMatrix[i + 1, j + 1] = matrix[k][i, j]. Помилка;
            У 1990-1990-1990-199
подвійний[,] двигун = новий двомісний[3, 3];
Для (int i = рушій. GetUpperBound(0); i >= 0; i--)
Для (int j = двигун. GetUpperBound(1); j >= 0; j--)
                двигун[двигун. GetUpperBound(0) - i, двигун. GetUpperBound(1) - j] = eng[k][i, j];
            eng = двигун;
double[,] nextLayout = новий подвійний[(newMatrix.GetUpperBound(0) + 1) - (двигун. GetUpperBound(1) + 1) + 1,
(newMatrix.GetUpperBound(1) + 1) - (двигун. GetUpperBound(1) + 1) + 1];
            Int curJ = 1;
            int curI = 1;
for (int z = 0; z < nextLayout.GetUpperBound(0) + 1; z++)
для (int w = 0; w < nextLayout.GetUpperBound(1) + 1; w++)
for (int i = curI; i < newMatrix.GetUpperBound(0); i++)
для (int j = curJ; j < newMatrix.GetUpperBound(1); j++)
                  {
                    подвійна сума = 0;
Для (int p = 1; p < двигуна. GetUpperBound(0) + 1; p++)
для (int l = 1; l < двигуна. GetUpperBound(1) + 1; l++)
                      {
                        sum = newMatrix[i - 1, j - 1] * двигун[p - 1, l - 1] + newMatrix[i - 1, j] * двигун[p - 1, l] +
newMatrix[i - 1, j + 1] * двигун[p - 1, l + 1] + +
                          newMatrix[i, j - 1] * двигун[p, l - 1] + newMatrix[i, j] * двигун[p, l] + newMatrix[i, j + 1]
* двигун[p, l + 1] +
                          newMatrix[i + 1, j - 1] * двигун[p + 1, l - 1] + newMatrix[i + 1, j] * двигун[p + 1, l] +
newMatrix[i + 1, j + 1] * двигун[p + 1, l + 1];
                        p = двигун. GetUpperBound(0) + 1;
                        l = двигун. GetUpperBound(1) + 1;
                      }
                    nextLayout[z, w] = sum;
                    Int curW = w;
                    curW++;
                    if (curW < nextLayout.GetUpperBound(1) + 1)
                    {
                      w++; curJ++;
                    }
                    else { curJ = 1; curI++; w = 0; z++; }
                  }
            Res. Додати(наступнийlayout);
          }
```

```
            повернення res;
        }
else повертає null;

    }
```

3 Шар виділення

```
загальнодоступний список<подвійний[,]>
MaxPooling(Список<подвійний[,]макет]>)//////////++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++
    {
Список<подвійний[,]> res = новий список<подвійний[,]>();
Для (int f = 0; f < компонування. Кількість; f++)
        {
double[,] результат = НОВИЙ подвійний[(макет[f]. GetUpperBound(0) + 1) / 2, (макет[f]. GetUpperBound(1) + 1) /
2];
Для (int z = 0; z < результат. GetUpperBound(0) + 1; z++)
У1980-х і 1990-х < 1990-х і 1990-х < 1990-х років. GetUpperBound(1) + 1; w++)
for (int i = 0; i < макет[f]. GetUpperBound(0) + 1; i += 2)
for (int j = 0; j < макет[f]. GetUpperBound(1) + 1; j += 2)
                    {
                      if ((i + 1) < макеті[f]. GetUpperBound(0) + 1 &amp;amp; (j + 1) < макеті[f]. GetUpperBound(1) + 1)
                      {
double[,] matrix = НОВИЙ подвійний[,] { макет[f][i, j], макет[f][i, j + 1] }, { layout[f][i + 1, j], layout[f][i + 1, j + 1] }
};
                        result[z, w] = Max(матриця);
                        якщо ((w + 1) < результат. GetUpperBound(1) + 1) w++;
                        else if ((z + 1) < результат. GetUpperBound(0) + 1) { z++; w = 0; }
                        else { z = результат. GetUpperBound(0) + 1; W = результат. GetUpperBound(1) + 1; i = макет[f].
GetUpperBound(0) + 1; j = макет[f]. GetUpperBound(1) + 1; }
                      }
                    }
          Res. Додати(результат);
        }
        повернення res;

    }
```

4 Пошук градієнта втрат для масштабів зведення

```
публічнийподвійний [,] FindDerivativeEngine(подвійна[,] матриця, подвійна[,] матрицяНаступнийдерів)
    {
      матриця = новий двомісний[4, 4] { {0.2,0.4,0.6,0.9 },{ 0.5,0.8,0.7,0.0.7},{0,1,0,2,-0,4,0,6 },{0,5,0,3,0,7,0,9 } };
      matrixNextDeriv = новий двомісний[4, 4] { { 0.2, 0.4, 0.8, 1.3 }, { 0.5, 1, 1.4, 1.7 }, { 0.1, 0.7, 0.2, 1.1 }, { 0.5,
0.4, 1.8, 1.4 } };
double[,] derivEngine1 = новий двомісний[3, 3];
подвійний[,] двигун = новий подвійний[matrixNextDeriv.GetUpperBound(0)+1, matrixNextDeriv.GetUpperBound(1)
+ 1];
Для (int і = рушій. GetUpperBound(0); i >= 0; i--)
Для (int j = двигун. GetUpperBound(1); j >= 0; j--)
          двигун[двигун. GetUpperBound(0) - i, двигун. GetUpperBound(1) - j] = matrixNextDeriv[i, j];
double[,] newMatrix = новий подвійний[матриця. GetUpperBound(0)+3, матриця. GetUpperBound(0) + 3];
У 1990 році він був < в 1990 році. GetUpperBound(0)+1; i++)
Для (int j = 0; j < матриця. GetUpperBound(1)+1; j++)
          newMatrix[i + 1, j + 1] = matrix[i, j];
      int strideY=-1, strideX = -1;
для (int i = 0; i < derivEngine1.GetUpperBound(0) + 1; i++)
      {
        крокY++;
```

```
для (int j = 0; j < derivEngine1.GetUpperBound(1) + 1; j++)
            {
               strideX++;
Для (int q = 0; q < рушія. GetUpperBound(0) + 1; q++)
Для (int w = 0; w < двигуна. GetUpperBound(1) + 1; w++)
                  {
                     derivEngine1[i, j] += newMatrix[q + strideY, w + strideX] * двигун[q, w];
                     if ((derivEngine1[i, j] > 3) || (derivEngine1[i, j] < -3))
                     { }
                  }
            }
            strideX = -1;
         }

         повернення derivEngine1;

      }
```

5 Зворотний розворот помилок

```
загальнодоступна порожнеча BackPropFlattedToConv(Список<Шар> шари, Список<Неврон> Помилка під'єднанняПідключення)
      {
Список<Накладання > = новий список<backLayers>();
List<Neuron[,]> cur = новий список<Неврон[,]>();
         кількість intCurLay = 19;
         int start = 0;
         int end = 0;
for (int i = 0; i < шари[шари. Кількість - 1]. Кінцеві значення.кількість;Кількість кінцевих оцінок;Кількість кінцевих оцінок;Кількість кінцев i++)
         {
            end = start + (шари[шари. Кількість - 1]. Кінцеві значення[i]. GetUpperBound(0) + 1) * (шари[шари. Кількість - 1]. Кінцеві значення[i]. GetUpperBound(1) + 1);
Нейрони = новий  Нейрон[шари[шари.] Кількість - 1]. Кінцеві значення[i]. GetUpperBound(0) + 1, шари[шари. Кількість - 1]. Кінцеві значення[i]. GetUpperBound(1) + 1];
Для (int z = start; z < кінець; z++)
for (int p = 0; p < шари[шари. Кількість - 1]. Кінцеві значення[i]. GetUpperBound(0) + 1; p++)
for (int f = 0; f < шари[шари). Кількість - 1]. Кінцеві значення[i]. GetUpperBound(1) + 1; f++)
                  {
                     нейрони[p, f] = помилкаПідключення домакс-пулу[z];
if ( подвійний. IsNaN(нейрони[p, f]. Помилка))
                     { }
                     z++;
                  }
            початок = кінець;
            Cur. Додати(нейрони);
         }


Для (int i = 0; i < 4; i++)
         {


            Список<Neuron[,]> erMp19ToRel18 = BackMaxPoolingToRelu(шари[countCurLay-1], cur);
Лежав. Додати(нові backLayers;"MaxPooling",countCurLay, cur, erMp19ToRel18));

            countCurLay--;

List<Neuron[,]> neur1 = новий список<Неврон[,]>();
для (int z = 0; z < erMp19ToRel18.Count; z++)
            {
Нейрони = новий  Нейрон[erMp19ToRel18[z]. GetUpperBound(0) + 1, erMp19ToRel18[z]. GetUpperBound(1) + 1];
for (int k = 0; k < erMp19ToRel18[z]. GetUpperBound(0) + 1; k++)
for (int j = 0; j < erMp19ToRel18[z]. GetUpperBound(1) + 1; j++)
```

```
                {
нейрони[k, j] = новий нейрон(шари[countCurLay]. StartValues[z][k, j], erMp19ToRel18[z][k, j]. Помилка); 2.
                }
            neur1. Додати(нейрони);
        }
Лежав. Додамо, що нові backLayers("ReLU", countCurLay, erMp19ToRel18, neur1));

        countCurLay--;


        Список<подвійний[,]>'ємReluToConv = Зворотнийконволюція(neur1, шари[countCurLay]. Двигуни); 2.
List<Neuron[,]> neur2 = новий список<Неврон[,]>();
для (int z = 0; z < помилкиReluToConv.Count; z++)
        {
Нейрони = новий нейрон[errorReluToConv[z].   GetUpperBound(0) + 1, помилкаReluToConv[z]. GetUpperBound(1) +
1];
for (int k = 0; k < помилкиReluToConv[z]. GetUpperBound(0) + 1; k++)
for (int j = 0; j <reluToConv[z]. GetUpperBound(1) + 1; j++)
                {
нейрони[k, j] = новий нейрон(шари[countCurLay]. StartValues[z][k, j], помилкаReluToConv[z][k, j]);
                  if ((нейрони[k, j]. ВагаСумна > 5) || (нейрони[к, дж]. ВагаСумна < -5))
                  { }
                }
            neur2. Додати(нейрони);
        }

Лежав. Додати(нові backLayers("Згортки",countCurLay, neur1, neur2, layers[countCurLay]. Двигуни));

        countCurLay--;

Список<Неврон[,]> neur3 = новий  список<Неврон[,]>();
Для (int z = 0; z < neur2. Кількість; z++)
        {
Нейрони = новий  нейрон[neur2[z]. GetUpperBound(0) + 1, neur2[z]. GetUpperBound(1) + 1];
for (int k = 0; k < neur2[z]. GetUpperBound(0) + 1; k++)
для (int j = 0; j < neur2[z]. GetUpperBound(1) + 1; j++)
                {
нейрони[k, j] = новий нейрон(шари[countCurLay]. StartValues[z][k, j], neur2[z][k, j]. Помилка); 2.
                }
            neur3. Додати(нейрони);
        }
Лежав. Додамо, що нові backLayers("ReLU",countCurLay, neur2, neur3));

        countCurLay--;
        Список<подвійний[,]> помилкиReluToConv1 = Зворотнийконволюція(neur3, шари[countCurLay].
Двигуни); 2.
List<Neuron[,]> neur4 = новий  список<Неврон[,]>();
        if (countCurLay != 0)
        {

для (int z = 0; z < помилкиReluToConv1.Count; z++)
            {
Нейрони = новий нейрон[errorReluToConv1[z].   GetUpperBound(0) + 1, помилкаReluToConv1[z].
GetUpperBound(1) + 1];
for (int k = 0; k < помилкиReluToConv1[z]. GetUpperBound(0) + 1; k++)
for (int j = 0; j <reluToConv1[z]. GetUpperBound(1) + 1; j++)
                {
нейрони[k, j] = новий нейрон(шари[countCurLay]. StartValues[z][k, j], помилкаReluToConv1[z][k, j]);
                }
            neur4. Додати(нейрони);
        }

Лежав. Додамо, що нові backLayers,countCurLay,neur3, neur4, layers[countCurLay]. Двигуни));
        }
```

```
            Ще
            {
Нейрони = новий нейрон[errorReluToConv1[0].    GetUpperBound(0) + 1, помилкаReluToConv1[0].
GetUpperBound(1) + 1];
for (int  k = 0; k < помилкуReluToConv1[0]. GetUpperBound(0) + 1; k++)
for (int  j = 0; j <reluToConv1[0]. GetUpperBound(1) + 1; j++)
                {
нейрони[k, j] = новий  нейрон(шари[countCurLay]. StartValues[0][k, j], помилкаReluToConv1[0][k, j]);
                  if ((нейрони[k, j]. ВагаСумна > 3) || (нейрони[к, дж]. ВагаСумна < -3))
                  { }
                }
              neur4. Додати(нейрони);
Лежав. Додамо, що  нові backLayers,countCurLay,neur3, neur4, layers[countCurLay]. Двигуни));
            }

            countCurLay--;

            cur = neur4;

      }
```