МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра математичних методів системного аналізу

Звіт з передипломної практики

на тему «Моделі нейронних мереж для класифікації пошкоджень на склі автомобіля»

Виконав: Перевірив:

студент 4 курсу керівник практики

групи КА – 76 доцент Тимощук О.Л.

Славінський Всеволод

Оцінка \_\_\_\_\_\_\_\_\_\_\_\_

"\_\_\_" \_\_\_\_\_\_\_\_ 2020 р.

**Науковий керівник дипломної роботи: проф. Недашківська Н.І.**

**рекомендовано оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Підпис

Київ 2021

ЗМІСТ

[0 ВСТУП 3](#_Toc72201823)

[1 РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 5](#_Toc72201824)

[1.1 Актуальність роботи 5](#_Toc72201825)

[1.2 Опис даних 5](#_Toc72201826)

[1.3 Аналіз даних 7](#_Toc72201827)

[1.4 Постановка задачі 10](#_Toc72201828)

[1.5 Формальна постановка задачі 11](#_Toc72201829)

[1.6 Висновки до розділу 1 11](#_Toc72201830)

[2 РОЗДІЛ 2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ 13](#_Toc72201831)

[2.1 Огляд згорткових нейронних мереж 13](#_Toc72201832)

[2.1.1 Згорткові нейронні мережі 14](#_Toc72201833)

[2.2 Архітектура нейронної мережі 17](#_Toc72201834)

[2.2.1 Input layer: 17](#_Toc72201835)

[2.2.2 Convolution layer: 17](#_Toc72201836)

[2.2.3 ReLu layer (активація): 18](#_Toc72201837)

[2.2.4 Pooling layer (об'єднання): 18](#_Toc72201838)

[2.2.5 Fully connected layer 18](#_Toc72201839)

[2.3 Види шарів 19](#_Toc72201840)

[2.3.1 Convolution 19](#_Toc72201841)

[2.3.2 Шари активації 20](#_Toc72201842)

[2.3.3 Pooling 22](#_Toc72201843)

[2.3.4 Fully connected layer 23](#_Toc72201844)

[2.3.5 Dropout 23](#_Toc72201845)

[2.4 Оптимізатори 24](#_Toc72201846)

[2.4.1 Градієнтний спуск 24](#_Toc72201847)

[2.4.2 Стохастичний градієнтний спуск 25](#_Toc72201848)

[2.4.3 Градієнтний спуск з моментом 26](#_Toc72201849)

[2.4.4 Adagrad 27](#_Toc72201850)

[2.4.5 AdaDelta 28](#_Toc72201851)

[2.4.6 Adam 28](#_Toc72201852)

[2.5 Preprocessing Image Data 29](#_Toc72201853)

[2.5.1 Нормалізація 30](#_Toc72201854)

[2.5.2 Аугментація 30](#_Toc72201855)

[2.6 Відомі архітектури 31](#_Toc72201856)

[2.6.1 MobileNet 31](#_Toc72201857)

[2.6.2 EfficientNet 32](#_Toc72201858)

[2.6.3 VGG 33](#_Toc72201859)

[2.6.4 Inception 35](#_Toc72201860)

[2.6.5 ResNet 37](#_Toc72201861)

[2.7 Висновки до розділу 2 39](#_Toc72201862)

[3 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 40](#_Toc72201863)

1. ВСТУП

У США один з пріоритетних напрямків страхування займає автострахування. Більшість випадків які припадають на автострахування потрапляють під категорії визначення пошкоджень які присутні на лобовому склі.

Страхові компанії дуже ретельно підходять до визначення категорії цього пошкодження і необхідного дії яку потрібно прийняти. Існує досить великий список категорій, який відповідає зразковому візуальному відображенню на склі а також іншим характеристикам.

За правилами цих пошкоджень існує 2 основних способи впливу при прийнятті рішень, такі як заміна лобового скла повністю або зробити ремонт за допомогою страхових партнерів.

Завдання полягає в тому щоб створити просте і гнучке рішення на основі веб і мобільного додатків, яке б дозволяло дуже просто створювати підтвердження для страхових компаній

Важливість роботи полягає в створенні моделі нейронної мережі, яка допоможе з класифікацією пошкоджень на склі автомобіля. Модель, отримана в результаті, значно полегшить роботу страховим компаніям у визначенні пошкоджень, а також зробить систему, яка не буде залежати від людського фактору, що позитивно вплине на страхових компаніях, а також на посередників

Об’єктом дослідження є аналіз зображень з пошкодженнями на склі автомобіля, які були отримані за допомогою реальних користувачів вже працюючого додатка, який було написано для однієї з компаній-клієнтів, а також зображення отриманні в результаті алгоритмів аргументації.

Предметом дослідження визначено згорткові нейронні мережі для класифікації зображень .

Пояснювальна записка складається з чотирьох розділів. У першому розділі проводиться постановка задачі, уточнення її актуальності, а також проводиться огляд об’єкта дослідження. У другому розділі розглянуто ряд математичних моделей зготкових нейронних мереж, за допомогою яких відбувається класифікація. Третій розділ описує архітектуру розробленої програми, також у ньому аналізуються результати роботи алгоритму. Четвертий розділ являє собою економічну частину, в якій розробляється кошторис витрат на розробку.

1. РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ
   1. Актуальність роботи

Задача є актуальною, тому що страхова сфера в США є досить складною і з великою кількістю різноманітних нюансів. Також, страхові компанії досить закриті і користуються застарілим програмним забезпеченням або паперовим видом оформлення страхових випадків.

Однією з ключових особливостей програми є можливість автоматичного визначення категорії пошкодження для того, щоб додаток міг працювати незалежно від людей, які є носіями знань, а це принесе велику гнучкість в майбутньому масштабуванні на велику кількість страхових компаній та на посередників. Саме це ми намагаємося досягти використовуючи глибоке навчання і згорткові нейронні мережі.

В майбутньому після досягнення необхідної кількості метрик і даних додаток зможе досить точно визначати і прогнозувати нові метрики, які зможуть більш ефективно управляти страховими випадками.

* 1. Опис даних

Набір даних був сформований реальними користувачами додатка, розробленого працівниками компанії WebLegends у рамках контракту з компанією, що займається випадками автострахування, та попередньо відредагований мною.

Зображення не знаходяться у відкритому доступі та повністю належать компанії-розробнику. Самі зображення являють собою фото зроблені за допомогою смартфонів. На зображеннях знаходяться автомобільні засоби сфотографовані з різних сторін, а також ліцензійні номери цих авто. На більшості зображень присутнє скло з тим чи іншим пошкодженням. Щоб нейронна мережа була точнішою, я обрізав зображення, залишивши лише самі пошкодження. Таким чином була прибрана уся чутлива інформація і залишилося лише те, що дійсно необхідно для нейронної мережі.

Изображение выглядит как окно

Автоматически созданное описание

Рисунок 1.1 Зображення з набору даних з підписами

Усі зображення зберігаються у форматах .jpg, .jpeg та .png та мають розмір 224х224 пікселя та три канали R, G, B. На рівні датасету зображення не поділені на тренувальні, валідаційні та тестові – розділення буде здійснене в момент завантаження даних до середи програмування.

Дані про розмічені області зберігаються у файлі dataset\_metadata.json у форматі зображеному на рисунку 1.2

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.2 Фрагмент файла dataset\_metadata.json

* 1. Аналіз даних

Загальна кількість зображень в наборі даних 9748. На кожному з зображень крупним планом зображена одна з двох категорій: «Repair» або «Replacement». При цьому 7163 з них належать категорії «Replacement» та 2585 до категорії «Repair».

Таку перевагу одної категорії над іншою дуже легко пояснити: у сучасному світі пошкоджене скло більш вірогідно потрібно буде повністю замінювати, оскільки ремонт лобового скла – це або дуже дорого, або ж зовсім неможливо. Тому оскільки дані, які належать до вибірки, абсолютно реальні, в цій перевазі нема зовсім нічого дивного.

Поглянемо детальніше як виглядають пошкодження кожного із класів.

Изображение выглядит как окно

Автоматически созданное описание

Рисунок 1.3 Зображення класа Replacement (повна заміна)

Як бачимо, зображення з категорії Replacement представляють собою тріщини та дірки у склі. Однак є деякі тріщини, які майже не видно на зображені через кут зору і можна лише здогадуватися де вони є.

Изображение выглядит как седзи, окно, здание

Автоматически созданное описание

Рисунок 1.4 Зображення класа Repair (ремонт)

На відміну від минулої категорії, в категорії Repair майже усі зображення мають сколи.

Звичайній людині доволі складно з впевненістю сказати де яка категорія, але цьому розподілу можна довіряти, оскільки його робили спеціально навчені люди. Що ж, подивимося як з цим впораються алгоритми машинного навчання.

* 1. Постановка задачі

Спочатку необхідно визначитися з типом задачі, яку ми розв’язуємо. Потрібно віднести кожне з зображень до одного з двох класів: Repair або Replacement. Тобто перед нами стоїть задача класифікації зображення.

Задача класифікації зображень – це задача, у якій міститься множина об’єктів (фотографій, малюнків, тощо), розподілених на класи за деякою логікою. Задана скінченна множина об’єктів, про яких відомо до яких класів вони належать. Ця множина називається вибіркою. У свою чергу ця вибірка розподіляється на 3 множини: тренувальна, валідаційна та тестова вибірки [1].

Тренувальна вибірка – вибірка, завдяки якій відбувається налаштування (оптимізація) ваг і коефіцієнтів моделі.

Валідаційна вибірка – вибірка для підбору параметрів, ознак та прийняття рішень, що стосуються навчання моделі (такі, як перенавчання)

Тестова вибірка – вибірка для оцінки якості навчання моделі. Тестова вибірка не повинна залежати від тренувальної [2].

Необхідно побудувати алгоритм, який буде здатний класифікувати довільний об’єкт із початкової множини.

Класифікувати об’єкт – указати номер або назву класу, до якого відноситься даний об’єкт.

В машинному навчанні задачі класифікації зображень вирішується, як правило, за допомогою методів штучної згорткової нейронної мережі  у вигляді навчання з учителем [1].

* 1. Формальна постановка задачі

Нехай Х – множина опису об’єктів, Y – скінченна множина номерів (імен) класів. Існує невідома цільова залежність-відображення , значення якої відомі тільки на об’єктах скінченної вибірки . Необхідно побудувати алгоритм: , здатний класифікувати довільний об’єкт .

Отже, поставимо задачу.

Дано: розміченні зображення з пошкодженнями на склі автомобільних засобів.

Опис системи: необхідно розробити систему, яка отримувала б фотографію на вхід, після цього користувач міг би виділити зону, на якій є пошкодження, а система відповіла, до якої з категорій це пошкодження належить.

Вимоги до системи: система повинна працювати на сервері, щоб користувачу не було потреби завантажувати будь які додаткові модулі та бібліотеки для роботи системи. Також система повинна мати зрозумілий програмний інтерфейс, щоб у майбутньому її можна було легко інтегрувати в іншу систему, що вимагає розпізнання пошкоджень.

Результат роботи: результатом роботи буде виведена на екран назва категорії.

* 1. Висновки до розділу 1

В даному розділі спершу була розглядана актуальність задачі класифікації зображень з пошкодженнями на склі авто.

Було проведено опис даних, з поясненням їх походження та методом збереження, а також з. поясненням форматів файлів., типів кодування, проведено аналіз предметної області.

Провели попередній аналіз вихідних даних, подивилися на те, як виглядають екземпляри кожного з класів.

В кінці розділу розглянули поняття класифікації зображень, поняття тренувальної, валідаційної та тестової вибірок, а також поставили задачу, вимоги та обмеження до системи, яка розроблюється.

1. РОЗДІЛ 2 МАТЕМАТИЧНІ ОСНОВИ РОБОТИ
   1. Огляд згорткових нейронних мереж

Нейронні мережі складаються з окремих одиниць, які називаються нейронами. Нейрони розташовані в ряді груп - шарів (див. Рисунок 2.1). Нейрони в кожному шарі з'єднані з нейронами наступного шару. Дані надходять від вхідного шару до вихідного шару вздовж цих сполук. Кожен окремий вузол виконує простий математичний розрахунок. Потім він передає свої дані у всі вузли, до яких він підключений.



Рисунок 2.1 Архітектура нейронної мережі.

На Рисунку 2.1 зображена архітектура нейронної мережі . Вона складається з вхідного шару, прихованих шарів та вихідного шару. Кількість прихованих шарів визначають глибину нейронної мережі. Вхідні дані поступають у вхідний шар і потім поширюються у прихований шар під номером один. В цьому шарі виконується лінійне перетворення вхідних даних, до яких потім застосовуються певна функція активації, яка додає нелінійності у модель. Аналогічним чином дані поширюються від одного до іншого прихованого шару. У вихідному шарі вже будується прогнозне значення , яке має бути максимально близьким до оригінального.

Якщо позначити – номер шару нейронної мережі, а – нейрон прихованого шару, то тоді значення, яке надходить до функції активації можна представити наступним чином:

|  |  |
| --- | --- |
|  | (2.1) |

де – ваги, зміщення та результат лінійного перетворення відповідно.

Функція активації, в свою чергу обчислюється як:

|  |  |
| --- | --- |
|  | (2.2) |

де – результат застосування функції активації;

– функція активації. [5]

* + 1. Згорткові нейронні мережі

Згорткові нейронні мережі (CNN) - це спеціальна архітектура штучних нейронних мереж, запропонована Яном ЛеКуном у 1988 р. Цей алгоритм може приймати вхідне зображення, призначати важливість різним аспектам / об'єктам на зображенні та мати можливість диференціювати один об’єкт від іншого.

Попередня обробка, необхідна в CNN, набагато нижча порівняно з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри розробляються вручну, при достатній підготовці, CNN має можливість вивчати ці фільтри / характеристики.

Архітектура CNN аналогічна структурі зв’язку нейронів в мозку людини і натхненна організацією зорової кори. Окремі нейрони реагують на подразники лише в обмеженій області зорового поля, відомому як рецептивне поле. Колекція таких полів накладається на всю зорову зону.[4]

Один із найпопулярніших напрямків використання цієї архітектури - класифікація зображень. Наприклад, Facebook використовує CNN для алгоритмів автоматичного позначення, Amazon - для генерації рекомендацій щодо продуктів, а Google - для пошуку серед фотографій користувачів. [3]

Операція згортки зазвичай позначають зірочкою ( [6]:

|  |  |
| --- | --- |
|  | (2.3) |

Якщо розглядати дану операцію з точки зору нейронних мереж, то функція – є вхідним значенням, а – ядром. Результат операції іноді називають картою характеристик.[5]

* + - 1. Представлення зображення у нейронній мережі

Розглянемо використання CNN для класифікації зображень більш детально. Основним завданням класифікації зображень є прийняття вхідного зображення та наступне визначення його класу. Це навичка, яку люди вивчають з народження і здатні легко визначити, що зображення на картині - це слон. Але комп'ютер бачить зображення зовсім інакше:

Изображение выглядит как текст, млекопитающее

Автоматически созданное описание

Рисунок 2.2 Як комп’ютер бачить зображення

Замість зображення комп’ютер бачить масив пікселів. Наприклад, якщо розмір зображення становить 300 x 300. У цьому випадку розмір масиву буде 300x300x3. Де 300 - це ширина, наступні 300 - це висота, а 3 - значення каналів RGB. Кожному з цих номерів комп’ютеру присвоюється значення від 0 до 255. Це значення описує інтенсивність пікселя в кожній точці. [3]

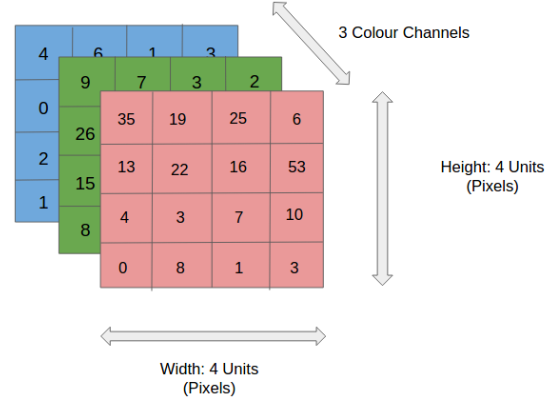


Рисунок 2.3 Матриця пікселів 4х4 в представленні комп’ютера

Ви можете собі уявити, наскільки обчислювальними будуть речі, як тільки зображення досягнуть розмірів, скажімо, 8K (7680 × 4320). Роль конволюційних шарів полягає у зменшенні зображень у форму, яку легше обробити, не втрачаючи особливостей, які є критично важливими для отримання хорошого прогнозу. Це важливо, коли ми хочемо розробити архітектуру, яка не тільки добре володіє функціями навчання, але й масштабована до масивних наборів даних.[4]

* 1. Архітектура нейронної мережі

Архітектури CNN складаються з окремих шарів. У всіх випадках шари приймають як вхідний 3D-об'єм, перетворюють цей об'єм за допомогою диференціальних рівнянь і виводять 3D-об'єм. Деякі шари вимагають налаштування гіперпараметрів, а інші - ні.

* + 1. Input layer:
* Вихідні значення пікселів зображення, представленого у вигляді 3D-матриці
* Розміри Ш х В х Г, де глибина відповідає кількості кольорових каналів на зображенні.
  + 1. Convolution layer:
* Конволюційні шари обчислюють вихідні дані вузлів, підключених до локальних областей вхідної матриці.
* Точкові добутки обчислюються між набором вагових коефіцієнтів (який зазвичай називають фільтрами) та значеннями, пов’язаними з локальною областю введення.
  + 1. ReLu layer (активація):
* Вихідний обсяг конволюційних шарів подається на поелементну функцію активації, як правило, ReLu.
* Шар ReLu визначатиме, чи буде вхідний вузол "працювати" з урахуванням вхідних даних.
* Якщо вузол "спрацює", це буде сигналізувати, що фільтри шару згортки виявили візуальну особливість.
* Функція ReLu застосовуватиме функцію max (0, x) з пороговим значенням 0.
  + 1. Pooling layer (об'єднання):
* Для зменшення ширини та висоти вихідного обсягу застосовується об’єднання пікселів за допомогою згортки.
  + 1. Fully connected layer
* Як і у звичайних нейронних мереж, кожен вузол у цьому рівні пов'язаний з кожним вузлом в обсязі об'єктів, що подаються далі.
* Імовірності класів обчислюються і виводяться в 3D-масиві з розмірами: [1x1xK], де K - кількість класів.

Далі детальніше пройдемося по кожному з шарів по черзі.[6]

* 1. Види шарів



Рисунок 2.3 Описана вище архітектура CNN з підписаними назвами шарів

* + 1. Convolution

Шар згортки завжди йде першим. У нього вводиться зображення (матриця зі значеннями пікселів). Зчитування вхідної матриці пікселів починається у верхньому лівому куті зображення. Далі програмне забезпечення вибирає там меншу матрицю, яка називається фільтром (також називають нейроном або ядром). Потім за допомогою фільтра відбувається згортка, тобто рух вздовж вхідного зображення.

Завдання фільтра – помножити його значення на початкові значення пікселів. Всі ці множення підсумовуються. В результаті виходить одне число. Оскільки фільтр зчитував зображення лише у верхньому лівому куті, він рухається все далі вправо на 1 одиницю, виконуючи таку саму операцію. Після роботи фільтра по всіх положеннях, виходиться матриця, але менша за вхідну. [3]

Згорткові шари – це будівельні блоки CNN. Ці шари зроблені з багатьох фільтрів, які визначаються їх шириною, висотою та глибиною. На відміну від щільних шарів регулярних нейронних мереж, згорткові шари побудовані з нейронів у 3-вимірах. Завдяки цій характеристиці, згорткові нейронні мережі є розумним рішенням для класифікації зображень. [6]

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2.4 Приклад роботи шару згортки

* + 1. Шари активації

Після згортки вхідне зображення було перетворено в набір карт ознак зображення. Кожна карта ознаки відповідає візуальному об’єкту, який видно у певних місцях на вхідному зображенні. Розмір цього набору ознак дорівнює кількості вузлів (тобто фільтрів) у згортковому шарі.

Функції активації визначають релевантність даного вузла в нейронній мережі. Вузол, що відповідає прогнозуванню моделі, буде спрацьовувати після проходження функції активації. [6]

Функції активації мають довгу історію. На початку, використовувалася сигмоїда через її похідну, діапазон значень від 0 до 1 та згладжену вірогіднісну форму. Її вигляд був наступний:

|  |  |
| --- | --- |
|  | (2.4) |

Її альтернативою був гіперболічний тангенс, значення якого лежали в інтервалі в -1 до 1:

|  |  |
| --- | --- |
|  | (2.5) |

Але класичні функції активації замінила ReLU (Рисунок 2.5). Її простота і ефективність підштовхнула її розвиток і пов’язаних з нею Leaky ReLU та параметризованою ReLU. [5]

Однією з найбільших переваг ReLU перед іншими функціями активації є те, що він не активує всі нейрони одночасно. З зображення для функції ReLU вище ми помітимо, що воно перетворює всі негативні входи в нуль, і нейрон не активується. Це робить його дуже обчислювальним, оскільки за раз активується мало нейронів.

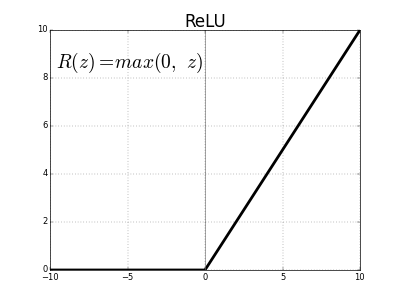


Рисунок 2.5 Графік функції ReLU

Деяким недоліком ReLU є те, що він насичений в негативній області, тобто градієнт у цій області дорівнює нулю. З градієнтом, рівним нулю, під час зворотного розповсюдження всі ваги не оновлюватимуться, щоб виправити це, ми використовуємо Leaky ReLU.

Крім того, функції ReLU не мають нульового центру. Це означає, що для того, щоб він дійшов до своєї оптимальної точки, йому доведеться використовувати зигзагоподібний шлях, який може бути довшим. [7]

* + 1. Pooling

Шар об'єднання можна побачити між шарами Convolution в архітектурі CNN. Цей рівень в основному зменшує кількість параметрів та обчислень у мережі, контролюючи надмірність шляхом поступового зменшення просторового розміру мережі.

Це має зменшити обчислювальну потужність, необхідну для обробки даних, завдяки зменшенню розмірності. Крім того, це корисно для вилучення домінантних ознак, які є інваріантними щодо обертання та позиції, таким чином підтримуючи процес ефективного навчання моделі.

Існує два типи об’єднання: Average Pooling та Max Pooling. Max Pooling повертає максимальне значення частини зображення, охопленої ядром. З іншого боку, Average Pooling повертає середнє значення всіх значень із частини зображення, охопленої ядром (Рисунок 2.6). [4]



Рисунок 2.6 Приклад роботи шару об’єднання

* + 1. Fully connected layer

У цьому шарі нейрони мають повний зв’язок з усіма активаціями попередніх шарів. Отже, їх активації можуть бути обчислені за допомогою матричного множення з подальшим зміщенням зміщення. Це остання фаза для мережі CNN. [7]

Метою повно зв’язного шару є прогнозування класів. Повністю зв’язаний шар прийме в якості вхідного сигналу сплощений вектор вузлів, які були активовані в попередніх шарах згортки. Вибір правильної функції активації залежить від типу проблеми класифікації:

* Сигмоїд зазвичай використовують для двійкових задач класифікації, оскільки він є логістичною функцією
* Softmax гарантує, що сума значень у вихідному рівні складає 1, і може використовуватися як для двійкових, так і для багатокласних задач класифікації. [6]
  + 1. Dropout

Також часто застосовується форма регуляризації, яка називається випадання. Випадання – це простий спосіб запобігти перенапруженню нейронних мереж.

Коли ми вводимо цю регуляризацію, ми випадковим чином відбираємо нейрони, які ігноруються під час навчання. Це тимчасово усуває внески в активацію нейронів, а також оновлення ваг (що відбуватиметься під час зворотного розмноження) також не застосовується. Це покращує узагальнення моделі та перешкоджає певним наборам ваг спеціалізуватися на конкретних особливостях, що може призвести до перенавчання, якщо не обмежено.

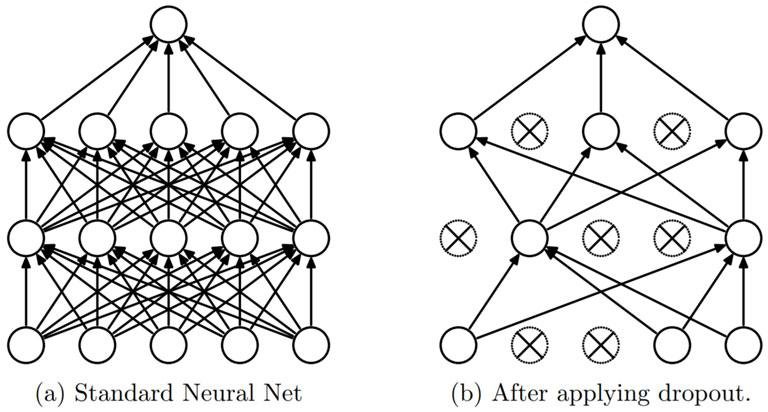


Рисунок 2.7 Приклад роботи шара випадання

* 1. Оптимізатори

Оптимізатори – це алгоритми або методи, що використовуються для зміни атрибутів вашої нейронної мережі, таких як ваги та швидкість навчання, щоб зменшити втрати.

Алгоритми та стратегії оптимізації відповідають за зменшення втрат і забезпечують максимально точні результати роботи. [8]

* + 1. Градієнтний спуск

Градієнтний спуск – це найосновніший, але найбільш часто використовуваний алгоритм оптимізації. Він широко використовується в алгоритмах лінійної регресії та класифікації. Зворотне поширення в нейронних мережах також використовує алгоритм градієнтного спуску.

Градієнтний спуск – це алгоритм оптимізації першого порядку, який залежить від похідної першого порядку функції втрат. Він обчислює, яким чином слід змінити ваги, щоб функція могла досягти мінімумів.

Алгоритм:

|  |  |
| --- | --- |
|  | (2.6) |

де – ваги нейронної мережі;

– гіперпараметр швидкості навчання;

– градієнт функції втрат.

**Переваги:**

* Простота обчислень.
* Легко реалізувати.
* Легко зрозуміти.

**Недоліки:**

* Ваги змінюються після розрахунку градієнта для всього набору даних. Отже, якщо набір даних занадто великий, час роботи буде дуже великий.
* Потрібна велика кількість пам’яті для обчислення градієнта для всього набору даних.
  + 1. Стохастичний градієнтний спуск

Це варіант градієнтного спуску. Він намагається частіше оновлювати параметри моделі. У цьому параметри моделі змінюються після обчислення втрат на кожному навчальному прикладі. Отже, якщо набір даних містить 1000 рядків, SGD буде оновлювати параметри моделі 1000 разів за один цикл набору даних, а не один раз, як у випадку стандартного градієнтного спуску.

Алгоритм:

|  |  |
| --- | --- |
|  | (2.7) |

де – тренувальні приклади.

Оскільки параметри моделі часто оновлюються, параметри будуть мати велику дисперсію та коливання функцій втрат при різній інтенсивності.

**Переваги**:

* Часті оновлення параметрів моделі сходяться за менший час.
* Потрібно менше пам'яті, оскільки немає необхідності зберігати значення функцій втрат.
* Може отримати нові мінімуми.

**Недоліки**:

* Висока різниця в параметрах моделі.
* Може стріляти навіть після досягнення глобальних мінімумів.
* Щоб отримати таку саму збіжність, як градієнтний спуск, потрібно повільно зменшувати значення рівня навчання.
  + 1. Градієнтний спуск з моментом

Градієнтний спуск з моментом був винайдений для зменшення великої дисперсії SGD та пом'якшення конвергенцію. Це прискорює зближення до відповідного напрямку і зменшує коливання до нерелевантного напрямку. У цьому методі використовується ще один гіперпараметр, відомий як імпульс, що символізується "γ".

|  |  |
| --- | --- |
|  | (2.8) |

де – гіперпараметр «момент».

Тоді правило оновлення приймає наступного вигляду [8]:

|  |  |
| --- | --- |
|  | (2.9) |

Термін імпульсу γ зазвичай встановлюється на 0,9 або подібне значення.

**Переваги**:

* Зменшує коливання і велику дисперсію параметрів.
* Збігається швидше, ніж градієнтний спуск.

**Недоліки**:

* Додано ще один гіперпараметр, який потрібно вибрати вручну.
  + 1. Adagrad

Одним з недоліків усіх пояснених оптимізаторів є те, що швидкість навчання є постійною для всіх параметрів і для кожного циклу. Однак Adagrad змінює швидкість навчання «η» для кожного параметра та на кожному кроці часу «t». Це алгоритм оптимізації другого порядку. Він працює над похідною функції помилки.

|  |  |
| --- | --- |
|  | (2.10) |

Тоді оновлення параметрів виконується за наступним правилом:

|  |  |
| --- | --- |
|  | (2.11) |

де – швидкість навчання, яка змінюється в залежності від параметру на основі минулих градієнтів;

– сума квадратів градієнтів ;

– елемент, який дозволяє уникати ділення ну нуль (зазвичай близький до ).

**Переваги**:

* Швидкість навчання змінюється для кожного навчального параметра.
* Не потрібно налаштовувати швидкість навчання вручну.
* Здатний тренуватися на обмежених даних.

**Недоліки**:

* Вимагає багато обчислень, оскільки потрібно обчислювати похідну другого порядку.
* Швидкість навчання завжди зменшується в результаті повільного навчання.
  + 1. AdaDelta

Це розширення AdaGrad, яке прагне усунути проблему затухаючої швидкості навчання. Замість того, щоб акумулювати квадрати минулих градієнтів, AdaDelta обмежує вікно накопичених минулих градієнтів певним фіксованим розміром w. При цьому використовується зважене середнє замість суми всіх градієнтів.

|  |  |
| --- | --- |
|  | (2.12) |

|  |  |
| --- | --- |
|  | (2.13) |

**Переваги**:

* Рівень навчання не занепадає, а навчання не припиняється.

**Недоліки**:

* Вимагає багато обчислень.
  + 1. Adam

Адам (Адаптивна оцінка моменту) працює з імпульсами першого та другого порядку. Його ідея полягає в наступному: ми не хочемо дуже швидко рухатися, оскільки можемо перестрибнути мінімум, ми хочемо зменшувати швидкість трохи, щоб проводити більш уважний пошук. На додаток до того, що Адам зберігає експоненційно зважене середнє квадратів минулих градієнтів, як і AdaDelta, Adam також зберігає експоненційно зважене середнє минулих значень градієнтів моменту.

|  |  |
| --- | --- |
|  | (2.14) |
|  | (2.15) |

де – гіперпараметри методу;

і – це значення першого і другого моментів, які ,в свою чергу, є відповідно середнім і невідцентрованою дисперсією градієнтів.

Оновлення параметрів тепер виглядає таким чином:

|  |  |
| --- | --- |
|  | (2.16) |

**Переваги**:

* Метод занадто швидкий і швидко зближується.
* Виправляє зникаючу швидкість навчання, велику дисперсію.

**Недоліки:**

* Вимагає багато обчислень.
  1. Preprocessing Image Data

Згадаємо, як зображення інтерпретується комп’ютером. Комп’ютер переводить зображення у тривимірний масив значень від 0 до 255.

Перш ніж ми зможемо навчити модель розпізнавати та класифікувати зображення, дані потрібно певним чином підготувати. Залежно від завдання комп'ютерного зору, деякі кроки попередньої обробки можуть не потребувати реалізації, але майже завжди потрібно буде виконувати нормалізацію та аугментацію.

* + 1. Нормалізація

Одною з найбільш поширених практик є нормалізація діапазону значень вхідних зображень перед подачею їх у модель. На цьому етапі попередньої обробки ми будемо масштабувати всі числові значення на наших зображеннях до значення в діапазоні від 0 до 1.

Це дуже важливо, оскільки деякі зображення можуть мати дуже високі значення пікселів, тоді як інші мають нижчі значення пікселів. Під час масштабування всіх зображень ви гарантуєте, що кожне зображення рівномірно сприяє функції втрати моделі.

Іншими словами, масштабування даних зображення гарантує, що всі зображення розглядаються однаково, коли модель тренується та оновлює свої ваги.

* + 1. Аугментація

Класифікація зображень за об’єктами повинна бути статистично незмінною. Нагадаємо, що всі зображення представлені у вигляді тривимірних масивів значень пікселів. Однак слід врахувати шанс того, що не всі зображення будуть виглядати однаково.

Необхідно щоб алгоритм був незмінним, щоб покращити його здатність узагальнювати дані зображення. Це може бути зроблено шляхом введення збільшення кількості зображень в етапі попередньої обробки. Збільшення кількості вхідних зображень збільшує дисперсію навчальних даних різними способами, такими як: включення випадкового обертання, збільшення або зменшення яскравості, зміщення позицій об'єкта та горизонтальне або вертикальне гортання зображень. [6]

* 1. Відомі архітектури

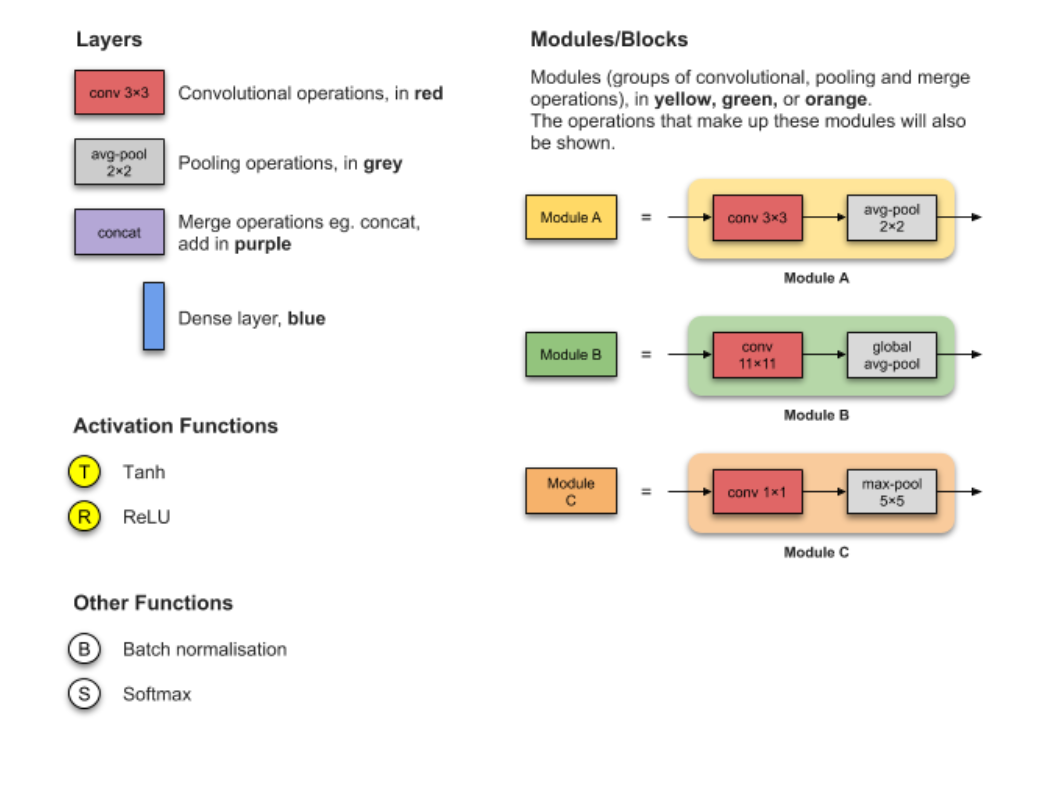


Рисунок 2.8 Умовні позначення шарів моделі

* + 1. MobileNet

Як підказує назва, модель MobileNet розроблена для використання в мобільних додатках, і це перша модель мобільного комп'ютерного зору TensorFlow.

MobileNet використовує відокремлені поглибленні шари згортки. Це значно зменшує кількість параметрів у порівнянні з мережею зі звичайними шарами згортки з однаковою глибиною в сітках. Це призводить до полегшення глибоких нейронних мереж.

Згортка, що відокремлюється по глибині, складається з двох операцій.

1. Глибока згортка.
2. Точкова згортка.

MobileNet - це невелика модель з низькою затримкою та малою потужністю, параметризовані для задоволення обмежень ресурсів у різних випадках використання. Вони можуть використовуватися для класифікації, виявлення, вбудовування та сегментації.

MobileNet - це клас CNN, який був відкритий для Google, і тому це дає нам чудову відправну точку для навчання наших класифікаторів, які мають бути доволі малими, але водночас швидкими. [10]

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 2.9 Архітектура MobileNet

* + 1. EfficientNet

EfficientNet - це згорткова архітектура нейронної мережі та метод масштабування, який рівномірно масштабує всі розміри глибини/ширини/роздільної здатності, використовуючи складений коефіцієнт. На відміну від звичайної практики, яка довільно масштабує ці фактори, метод масштабування EfficientNet рівномірно масштабує ширину, глибину та роздільну здатність мережі за допомогою набору фіксованих коефіцієнтів масштабування. [11]

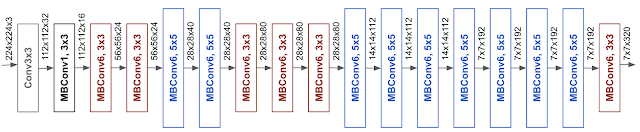


Рисунок 2.10 Архітектура EfficientNet [12]

Складений метод масштабування виправдовується інтуїцією того, що якщо вхідне зображення більше, то мережі потрібно більше шарів для збільшення сприйнятливого поля та більше каналів для захоплення більш дрібнозернистих візерунків на більшому зображенні.

* + 1. VGG

Люди з Visual Geometry Group (VGG) винайшли VGG-16, який має 13 згорткових та 3 повністю зв’язані шари. Першою важливою відмінністю, яка стала фактичним стандартом, є використання великої кількості маленьких фільтрів. Зокрема, фільтри розміром 3 × 3 та 1 × 1 з кроком один (Рисунок 2.11), відрізняються від великих розмірів фільтрів у LeNet-5 та менших, але все ще відносно великих фільтрів та великого кроку чотирьох в AlexNet.

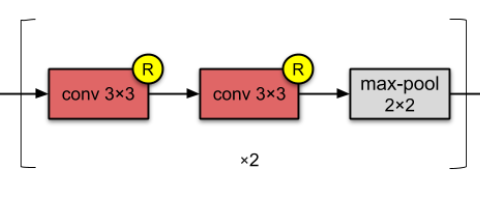


Рисунок 2.11 Група шарів згортки та об’єднання

Шари Max Pooling використовуються після більшості, але не всіх згорткових шарів, виходячи з прикладу в AlexNet, але все об'єднання виконується з розміром 2 × 2 і тим самим кроком, який теж став фактичним стандартом.

Зокрема, мережі VGG використовують приклади двох, трьох і навіть чотирьох згорткових шарів, складених разом перед використанням максимального шару об'єднання. Обгрунтуванням було те, що укладені згорткові шари з меншими фільтрами наближають ефект одного згорткового шару з фільтром більшого розміру, наприклад три складені згорткові шари з фільтрами 3 × 3 наближаються до одного згорткового шару з фільтром 7 × 7.

Інша важлива відмінність - дуже велика кількість використовуваних фільтрів. Кількість фільтрів збільшується із глибиною моделі, хоча починається з відносно великої кількості 64 і збільшується через 128, 256 та 512 фільтрів в кінці частини видобування особливостей моделі.[13]

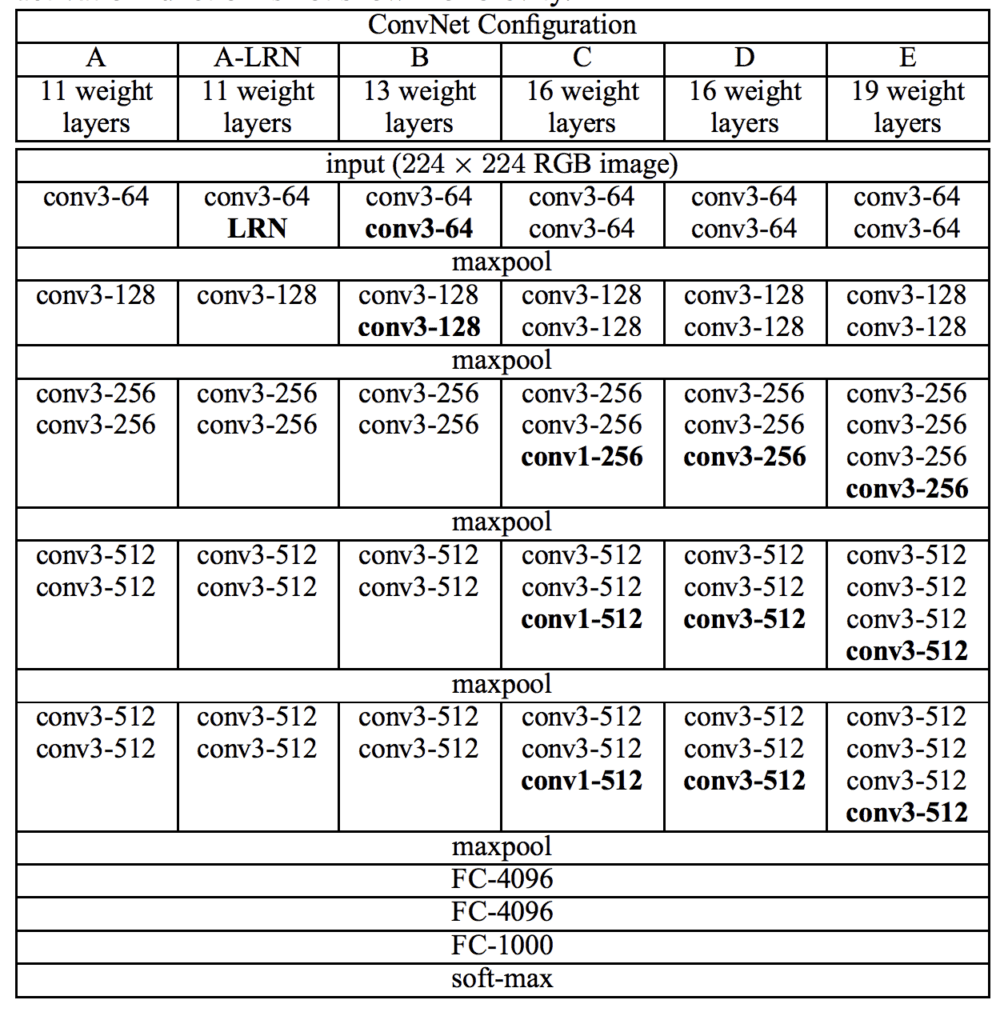


Рисунок 2.12 Архітектура мережVGG

* + 1. Inception

Автори пропонують архітектуру, що називається початком. Ключовим нововведенням у цих моделях є початковий модуль. Це блок паралельних згорткових шарів з фільтрами різного розміру (наприклад, 1 × 1, 3 × 3, 5 × 5) та шаром об’єднання 3 × 3 max, результати якого потім об’єднуються (Рисунок 2.13).

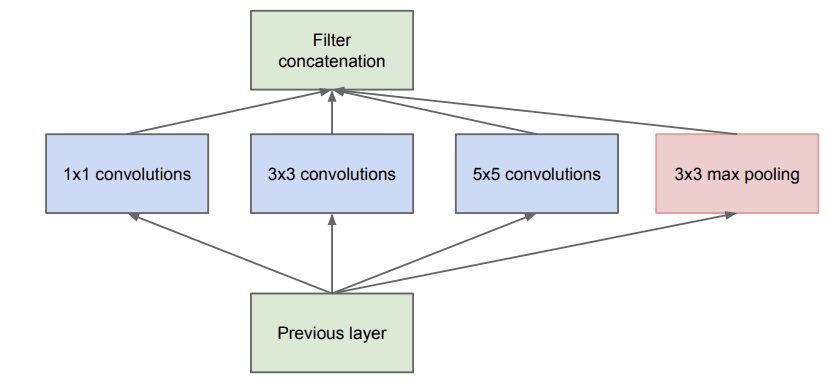


Рисунок 2.13 Модуль, запропонований в моделі Inception

Другим важливим дизайнерським рішенням у моделі Inception було підключення виходу в різних точках моделі. Це було досягнуто шляхом створення невеликих вихідних мереж з основної мережі, які були навчені робити прогнози. Метою було надати додатковий сигнал помилки із завдання класифікації в різних точках глибинної моделі для вирішення проблеми зникаючих градієнтів. Потім ці невеликі вихідні мережі були видалені після навчання.

Нижче (Рисунок 2.14) наведена повернута версія (зліва направо для вводу-виводу) архітектури моделі Inception. Початок від входу ліворуч до класифікації виходів праворуч і два додаткові вихідні мережі, які використовувались лише під час навчання.[13]

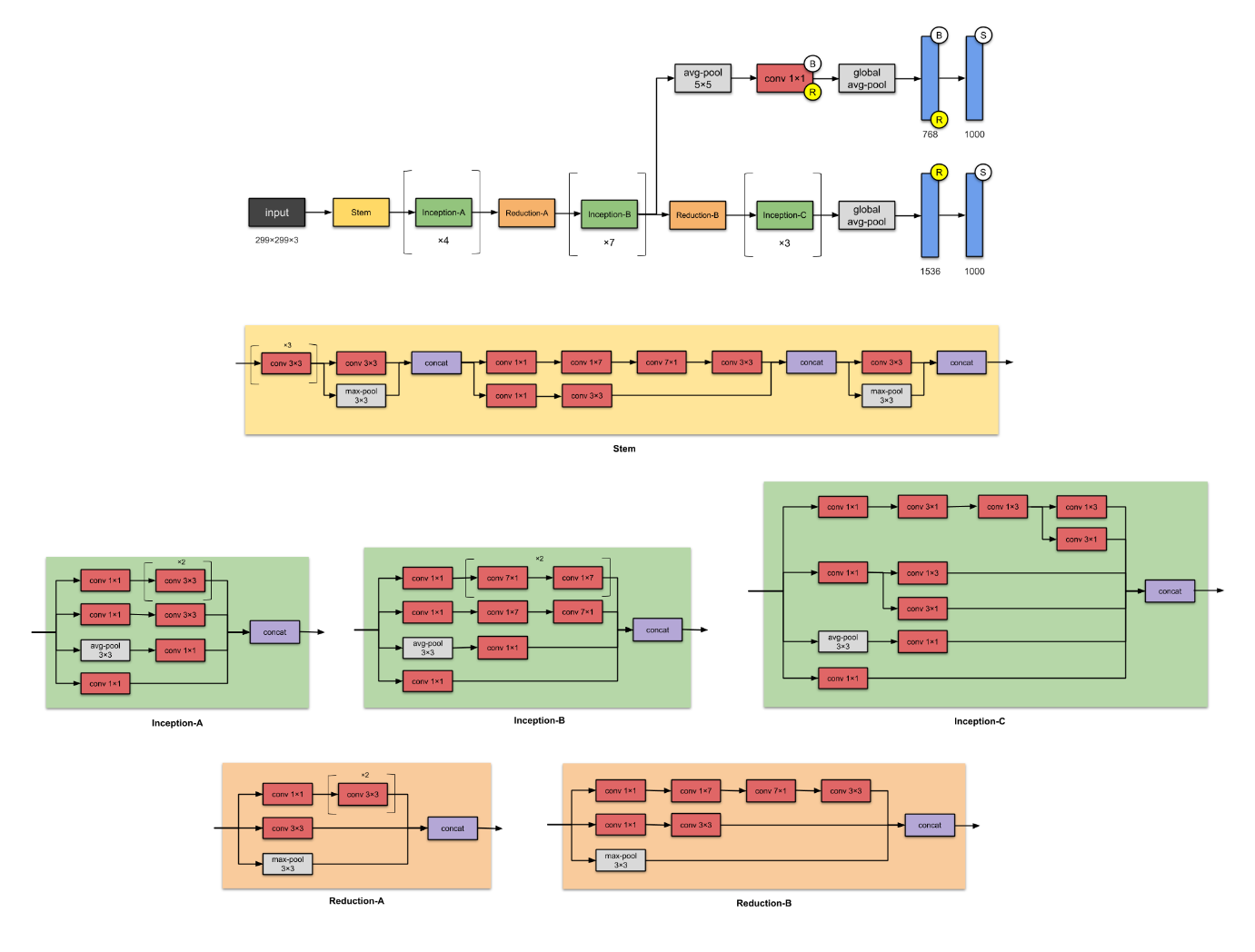


Рисунок 2.14 Архітектура моделі Inception

* + 1. ResNet

Коли кількість мережевих рівнів CNN зросла до певного числа, натомість продуктивність CNN знизилася. Причина полягає в тому, що глибші мережі набагато важче оптимізувати через горезвісну проблему зникнення / вибуху градієнтів.

Чи є спосіб збільшити кількість згорткових шарів, уникаючи проблеми вибуху / зникнення градієнта? У 2015 році для вирішення цієї проблеми було винайдено ResNet. [14]

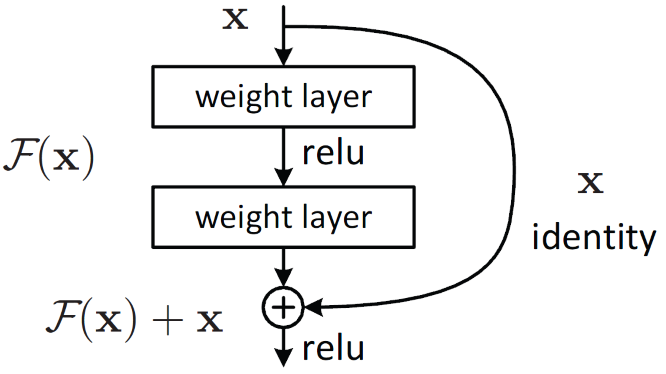


Рисунок 2.15 Залишкове навчання

ResNet представив концепцію, яка називається Залишкове навчання. Інтуїтивно висновок кожного шару згортки є принаймні таким же гарним, як і вхідний сигнал. тобто F (x) + x ≥ x. Доведено, що ця архітектура добре вирішує проблему градієнта. Найбільша кількість згорткових шарів ResNet може бути більше 1000.

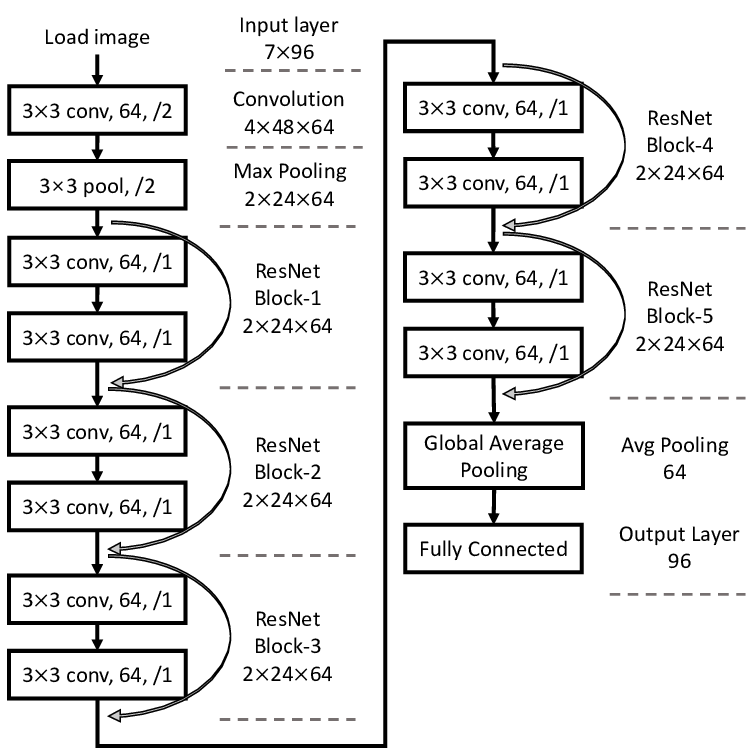


Рисунок 2.16 Архітектура ResNet

* 1. Висновки до розділу 2

У другому розділі ми досить детально з’ясували основи нейронних мереж. Особлива увага була саме до зготкових нейронних мереж, тому що саме вони використовуються для задач класифікації зображень. Також з’ясували як саме комп’ютер інтерпретує вхідне зображення.

Було викладено базові поняття по архітектуру зготкових мереж, а також про шари згортки, об’єднання, активації, повно зв’язні шари і шари викидання.

Після цього були розглянуті базові принципи попередньої обробки зображень, які необхідно робити при роботі з CNN.

Також було викладено міркування стосовно оптимізаторів, які широко використовуються у глибинному навчанні. До кожного з них було наведено переваги та недоліки.

Наприкінці були наведені найбільш поширені архітектури згорткових нейронних мереж, а саме MobileNet, EfficientNet, VGG, Inception та ResNet. До кожного з них було наведено зображення з архітектурою, а також були пояснені деякі найбільш цікаві та незвичні модулі, завдяки яким ці мережі стали такими відомими.

1. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ
2. Классификация. URL:

<http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%81%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F>

1. Как сформировать датасет для машинного обучения URL: <https://www.bigdataschool.ru/blog/dataset-data-preparation.html>
2. Image Classification with Convolutional Neural Networks URL: <https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8#:~:text=Convolutional%20neural%20networks%20(CNN)%20is,this%20architecture%20is%20image%20classification.&text=Instead%20of%20the%20image%2C%20the%20computer%20sees%20an%20array%20of%20pixels>
3. A Comprehensive Guide to Convolutional Neural Networks URL: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
4. Недашківська Н. І. курс лекцій
5. Convolutional Neural Networks — Image Classification w. Keras URL: <https://www.learndatasci.com/tutorials/convolutional-neural-networks-image-classification/>
6. Basic Overview of Convolutional Neural Network (CNN) URL: <https://medium.com/dataseries/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17#:~:text=The%20activation%20function%20is%20a,neuron%20would%20fire%20or%20not.&text=We%20have%20different%20types%20of,Rectified%20Linear%20Unit%20(ReLU)>
7. Various Optimization Algorithms For Training Neural Network URL: <https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6>
8. Illustrated: 10 CNN Architectures URL: <https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>
9. Image Classification With MobileNet URL: <https://medium.com/analytics-vidhya/image-classification-with-mobilenet-cc6fbb2cd470>
10. EfficientNet URL: <https://paperswithcode.com/method/efficientnet#:~:text=EfficientNet%20is%20a%20convolutional%20neural,resolution%20using%20a%20compound%20coefficient>
11. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling URL: <https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html>
12. Convolutional Neural Network Model Innovations for Image Classification URL: <https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/>
13. From AlexNet to NASNet: A Brief History and Introduction of Convolutional Neural Networks URL: <https://towardsdatascience.com/from-alexnet-to-nasnet-a-brief-history-and-introduction-of-convolutional-neural-networks-cf63bf3320e1>