Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики

ФУНКЦІЇ АКТИВАЦІЇ В АРХІТЕКТУРІ НЕЙРОННИХ МЕРЕЖ

Текстова частина до курсової роботи
за спеціальністю «Прикладна математика»

Керівник курсової роботи
ст. викладач Швай Н.О.

_____
(підпис)
“____” _____ 2022 р.

Виконав студент 1 р. н.
Мокрий М. В.
“____” _____ 2022 р.

Київ – 2022

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра математики

ЗАТВЕРДЖУЮ

Зав. кафедри математики

профессор, д.ф.-м.н.

Б. В. Олійник

_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Мокрому Михайлу Вікторовичу 1-го курсу факультету інформатики

ТЕМА Функції активації в архітектурі нейронних мереж

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

Загальні відомості про штучні нейронної мережі

Проблеми функцій активації

Методи ініціалізації ваг

Найбільш поширені функції активації, їх характеристики

Огляд датасету CIFAR-10 та архітектури нейронних мереж

Результати проведеного дослідження

Висновки

Список літератури

Дата видачі „___” _____ 2022 р. Керівник _____

Завдання отримав _____

Тема: Функції активації в архітектурі нейронних мереж

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу.	15.11.2021	
2.	Аналіз предметної області.	18.12.2021	
3.	Знайомство з функціями активації	Березень 2022	
3.	Написання практичної частини роботи.	Квітень 2022	
4.	Написання текстової частини роботи.	Травень 2022	
5.	Написання текстової частини роботи.	Березень 2022	
6.	Створення презентації та написання доповіді.	30.06.2022	
7.	Захист курсової роботи.	13.07.2022	

Студент Мокрий М. В.

Керівник Швай Н. О.

“_____”_____

Зміст

Анотація

Курсова робота присвячена дослідженню використання функцій активації в архітектурах нейроних мереж. Для практичної частини роботи була використана мова програмування Python, середовище розробки Google Colab, а також бібліотека Tensorflow.

Ключові слова: функції активації нейроних мереж, машинне навчання, ResNet, VGG, ViT, Google Colab, Python, CNN, Transformers, CIFAR-10, Image classification, TensorFlow, TensorFlow Keras.

Вступ

Штучні нейронні мережі з кожним роком набувають все більшої популярності. Вони можуть виконувати різноманітні задачі, такі як пошук і класифікація об’єктів, семантична сегментація, обробка природної мови та багато інших. Водночас з’являються нові моделі та нові архітектури штучних нейронних мереж, а також й нові задачі для яких ці моделі створюються. Проте все це не змінює початкової логіки нейронної мережі, в якій основним елементом є перцепрон. Зараз зазвичай достатньо написати декілька команд, щоб нейрона мережа з вже підготовленими вагами почала навчатися та після успішного навчання виконувати своє основне призначення. Тому в сучасному світі існує багато людей які лише поверхнево знайомляться з нейронними мережами, не заглиблюючись у їх будову, що в свою чергу призводить до відповідних неточних результатів.

Успішне тренування моделі залежить від великої кількості факторів, а саме: архітектури, глибини нейронної мережі, тобто кількості шарів, початкових параметрів моделі, методу ініціалізації ваг, датасету, який використовується для тренування моделі, а також функції активації. Про останній фактор, функцію активації, багато хто просто забуває, бо вона вже присутня в будь-якій стандартній моделі за замовчуванням. Тому ніхто й не думає, як можна покращити результати, змінивши стандартну функцію активації.

Головною метою цієї курсової роботи є аналіз функцій активації на різних архітектурах і моделях нейронної мережі та дослідження їх поведінки на них. Основними завданнями даного дослідження є:

- Опис різних функцій активації, їх особливостей та проблем;
- Дослідження роботи функцій активації на різних моделях та архітектурах штучних нейронних мереж;
- Аналіз кінцевих результатів та пошук закономірностей між ними;
- Визначення від яких параметрів залежить вибір функції активації.

Об’єктом дослідження є аналіз роботи функцій активації на різних архітектурах нейронних мереж.

Предметом дослідження є сам експеримент з функціями активації.

Курсова робота складається з семи розділів. Перший розділ присвячений опису загальних відомостей про роботу штучної нейронної мережі. У другому та третьому розділі описуються основні проблеми, з якими стикаються функції активації, а також існуючі методи ініціалізації ваг. Четвертий розділ присвячений виведенню основних характеристик функції активації. В п’ятому розділі детально описуються вибрані функції активації, їх особливості та проблеми, які притамані ним. В шостому розділі розповідається про датасет CIFAR-10. Сьомий розділ присвячений різним архітектурам та моделям штучних нейронних мереж, які були взяті для проведення дослідження. І останній розділ містить в собі реалізацію експерименту та наведення кінцевих результатів.

У результаті курсової роботи було проведено дослідження різних функцій активації на трьох моделях нейронних мереж і подальший аналіз отриманих результатів.

1 Загальні відомості про роботу нейронної мережі

Функції активації є найбільш важливим елементом будь-якої штучної нейронної мережі[1]. Це обумовлено однією дуже важливою причиною — функція активації використовується при зміні ваги перцептрону в процесі навчання нейронної мережі. Варто зазначити, що в реальних мережах кількість шарів набагато більше.

Першим елементом перцептрону є вхідний вектор, який можна записати як $x = [x_1 x_2 x_3 \dots x_m]$ розміром m. Другим елементом є вектор ваг, котрий позначається $w = [w_1 w_2 w_3 \dots w_m]$ розміром m. Вага або іншими словами синаптична вага відповідає за силу з'єднання з вхідними даними, на основі якої і будується вихідний результат перцептрона. Використовуючи початковий вхідний вектор і вектор ваг будується зважений вхідний результат цих обчислень, які позначається як $x^T w$ і називається зваженою сумою. Варто зазначити, що в процесі обчислення до вектора ваг додається нульовий елемент (w_0), а до вхідного вектора додається одиниця, які потрібні для подальших обчислень. Цей елемент прийнято називати біасом. Отже, результат зважених вхідних обчислень буде мати такий вигляд: $x^T w = w_0 + x_1 w_1 + x_2 w_2 + \dots + x_m w_m$. Наступним елементом перцептрону є функція активації, яка в якості параметра отримує на вхід зважену суму. Це можна відобразити за допомогою математичної формули $z = x^T w$ та $y = f(z)$, де z – зважена сума, а y – функція активації. Результат обчислення функції активації використовується для того, щоб нейронна мережа обчислила різницю між бажаним вихідним результатом і вихідним результатом роботи функції активації. Цей процес називається обчисленням рівеня втрат. Значення втрат використовуються при оновленні вектору ваг в процесі навчання нейронної мережі. Але для того, щоб мати змогу оновлювати вектор ваг під час навчання й успішно почати навчання нейронної мережі, потрібно мати початкові значення ваг. Процес присвоєння початкових

значень до вектору ваг називається ініціалізацією ваг і відіграє важливе значення в успішності і швидкості навчання. Отже, вірний метод ініціалізації ваг в поєднанням з правильно підібраною функцією активації може суттєво покращити точність та швидкість навчання. Результатом цієї роботи є дослідження того, яка функція активації є найкращою для нейронних мереж з різними архітектурами та різними методами ініціалізації ваг.

2 Проблеми, з якими стикаються функції активації

Функції активації є різноманітними. Кожна з них має власну швидкість обчислення, власну специфіку та особливості, а також власні проблеми. Найосновніші проблеми, які дуже часто зустрічаються в найбільш популярних і широко використовуваних функціях активації є проблема мертвого нейрона і проблема зникнення градієнту[2].

2.1 Проблема мертвого нейрона

Проблема мертвого нейрона з’являється коли нейрон отримує на вхід дуже багато нулів, і як результат функція активації отримує зважену суму близькою для нуля. Під час оновлення ваг є велика вірогідність того, що зважена сума більшої частини нейронної мережі буде завжди нульовою. Це спричинить проблему того, що більша частина нейронної мережі стане неактивна і не буде працювати, тобто в цих нейронах функція активації буде повертати нульові значення, що унеможливить коректне оновлення ваг і навчання мережі. Дана проблема називається проблемою мертвого нейрону і зустрічається в багатьох функціях активації, які не мають функціоналу боротися з великою кількістю нульових ваг.

2.2 Проблема зникнення градієнту

Не менш серйозною проблемою є проблема зникнення градієнту. Вона з’являється, коли великі зміни вхідних даних, після їх опрацювання функцією активації, призводять до значних змін на виході. Внаслідок цього зміни значення градієнту стають або дуже маленькі, або близькі до нуля, що негативно позначається на обчисленні втрат, значення яких перестає зменшуватись. Як результат оновлення ваг майже не відбувається, а нейрона мережа просто перестає навчатись і “застрягає” на одному місці. Цю проблему характеризують як проблему зникнення градієнту, а нейрони, які містять ваги, що не оновлюються, коректно називають насиченими нейронами. Дана проблема також зустрічається в багатьох функціях активації.

3 Методи ініціалізації ваг

Існує декілька різних методів активації нейронних ваг, починаючи від найбільш базових, таких як заповнення початкових значень нулями або випадкової ініціалізації, тобто заповнення початкових значень випадковими числами визначеного діапазону, що працює краще ніж заповнення нулями. Більш серйозними є методи ініціалізації Xavier та He, які почали активно використовуватись одразу після їх представлення. Базові методи ініціалізації ваг можуть використовуватися для деяких специфічних завдань, але їх використання для цього експерименту є сумнівним. Саме тому було вирішено використовувати найбільш популярні методи ініціалізації ваг такі як Xavier і He.

3.1 Метод ініціалізації Xavier

Метод Xavier[3], який дуже часто називається Glorot Normal – це метод ініціалізації ваг, основна ідея якого полягає в ініціалізації ваг випадковими значеннями, використовуючи нормальний розподіл з стандартним відхиленням

в межах одиниці і середнім значенням, що дорівнює нулю. Метод ініціалізації Xavier можна представити за допомогою формули $(-\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}})$, де n_{in} – це кількість вхідних з'єднань в шарі нейроної мережі, а n_{out} – кількість вихідних з'єднань.

Також було досліджено, що звичайний метод ініціалізації ваг за допомогою випадкових значень не дуже підходить для сигмоїдних функцій активації, тому ця функція активації найкраще підходить для них. Але це створює проблеми для функцій активаціїї сімейства ReLU, тому що ініціалізація Xavier була заснована на припущенні, що функція активація повинна бути функцією активації сімейства сигмоїдних функцій. Це твердження не підходить для функцій активації ReLU, через це було створено невелику модифікацію цього метода ініціалізації ваг, щоб зробити її придатною для використання. Метод Xavier для функцій активації сімейства ReLU можна представити за допомогою формули $(-\sqrt{2}\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}}, \sqrt{2}\frac{\sqrt{6}}{\sqrt{n_{in}+n_{out}}})$.

3.2 Метод ініціалізації He

Метод ініціалізації ваг He[4], або як його ще часто називають He Normal, працює трохи іншим чином. У цьому методі ваги відштовхуються від розміру попереднього шару нейронної мережі, що призводить до успішного досягнення глобальної мінімізації та покращенні ефективності функції. Метод He Normal можна представити за допомогою формули $(-\frac{\sqrt{6}}{\sqrt{n_{in}(1+a^2)}}, \frac{\sqrt{6}}{\sqrt{n_{in}(1+a^2)}})$, де n_{in}— кількість вхідних з'єднань в шарі нейроної мережі, а a – це параметр функції активації та нормального розподілу. Цей метод покращує швидкість тренування нейронної мережі і його рекомендують використовувати для функцій активації типу ReLU, але це твердження потребує додаткової перевірки.

4 Основні характеристики функцій активації

Функції повинні мати певні властивості задля того, щоб називатися функціями активації. Нижче показано основні властивості цих функцій, а саме:

1. Нелінійність.

   Закономірності реального світу з різних сфер життя найчастіше не можуть бути лінійними. Саме тому нелінійні функції дають більший простір для моделювання тієї чи іншої проблеми реального світу і подальшої її відображенні у вхідні дані нейронної мережі. Також нелінійність дає змогу використовувати глибину нейронної мережі, або кількість шарів, на відміну від лінійності, яку мої можна представити як сукупність лінійних комбінацій вхідного вектору.

2. Неперервність

   Властивість яка показує, що функція є неперервною на всій довжині її області визначення. Тобто в кожній точці на площині в області визначення функція має своє значення. Ця властивість тісно пов’язана з наступною властивістю – диференційованістю функції.

3. Диференційність

   Функція активації повинна бути диференційованою задля того, щоб рахувати градієнт.

4. Продуктивність

   Кожна функція активації має свою складність обчислення. Чим складніше функція, тим більше часу потрібно для того, щоб вона обчислила вхідні дані.

5. Центрованість

   Здатність функції мати як негативні, так і позитивні значення. Тобто функція не є завжди лише позитивною, або лише негативною. Це дає їй

певну гнучкість, зменшуючи кількість епох для успішного навчання нейронної мережі. Є не обов'язковою властивістю функції активації.

6. Обмеженість

Обмеженість зумовлена тим, що вхідні дані проходять через мережу перцептронів, кожен з яких має свою функцію активації. Тому функція повинна бути обмежена, щоб вихідний результат не був або завеликим, або дуже малим. Ця властивість також не є обов’язковою у функціях активації.

5 Найбільш розповсюджені функції активації

5.1 Сигмоїдна функція

Сигмоїдна або іншими словами логістична сигмоїдна функція[5] була однію з найбільш популярних функцій активацій, особливо на початкових стадіях розвитку нейронних мереж. Сигмоїдну функцію можна зобразити за допомогою формули $f(x) = \frac{1}{1+e^{-x}}$, де x – вхідні дані. Основними властивостями сигмоїдної функції є неперервність та обмеженість на проміжку від 0 до 1 як це добре видно на її графіку.

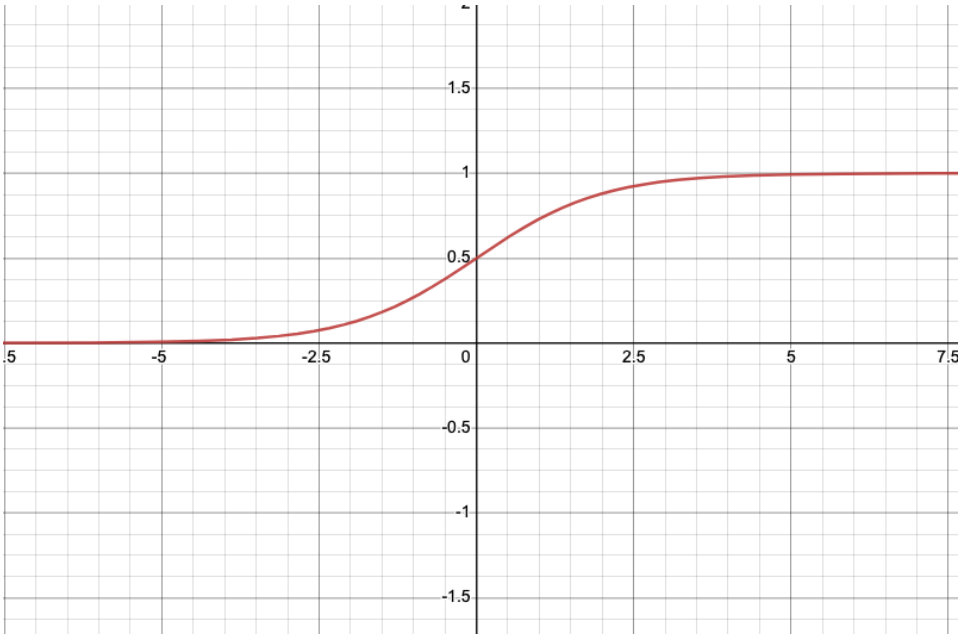

Рисунок 5.1.1 Графік сигмоїдної функції активації

Також варто зазначити, що ця функція відноситься до групи сигмоподібних функцій. Сигмоїдна функція є доволі ресурсозатратною тому, що в її основі є експоненційна функція, яка і є причиною такої складності обчислень. Незважаючи на таку популярність, функція активації має також і проблеми через те,що областю визначення функції є маленький інтервал (0, 1), функція завжди повертає невід'ємні значення і є нецентрованною. Маленькі дані на виході, які отримуються в результаті можливого надходження великих даних можуть спричинити проблему зникнення градієнту. Тож функція має два основних недоліки нецентрованість, а також проблему зникання градієнту.

5.2 Тангенсна функція

Класична функція активації, яка вирішує одну з проблем сигмоїдної функції, а саме відсутність центрованості. Тангенсна або іншими словами гіперболічна тангенсна функція[6] є популярнішою ніж сигмоїдна через те, що вона показує кращу продуктивність, особливо на багатошарових нейронних мережах. Функцію активації можна зобразити за формулою

$f(x) = \frac{1-e^{-x}}{1+e^{-x}}$, де x параметр функції, який отримує вхідні дані.

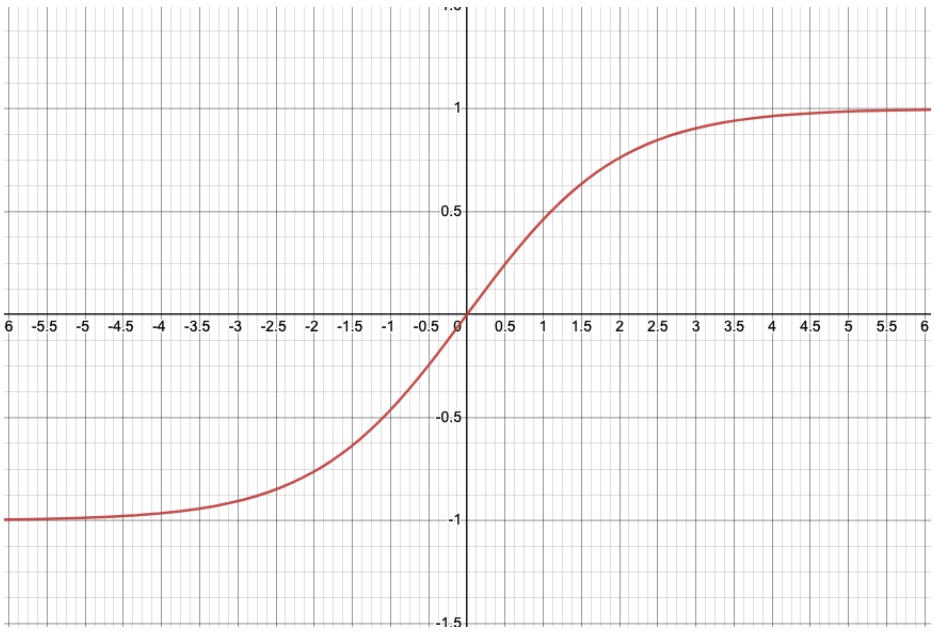

Рисунок 5.2.1 Графік тангенсної функції активації

По формулі видно, що вона не сильно відрізняється від сигмоїдної, тому що є певною її модифікацією. Отже, вона має ті самі властивості що і сигмоїдна функція, тому й відноситься до сигмоподібної групи функцій активації, але її область визначення збільшується в два рази і займає проміжок (-1, 1). Вихідними результатами можуть бути і негативні, і позитивні числа в цьому невеликому проміжку, а також число нуль, що надає їй властивості центрованої функції активації. Це покращує її характеристики і вирішує одну з проблем сигмоїдної функції. Але невеликий інтервал вихідних результатів спричиняє ту саму проблему, що і минула функція, а саме проблему зникнення градієнту, від якої вона і страждає.

5.3 Функція ReLU

Функція ReLU[7] (англ. Rectified linear unit) вирішує проблему зникнення градієнту, яка існує в тангенсній функції. Цю функцію можна записати за допомогою формули $f(x) = max(0, x)$, де x вхідні дані функції активації. Її можна зобразити графіком, поданому нижче.

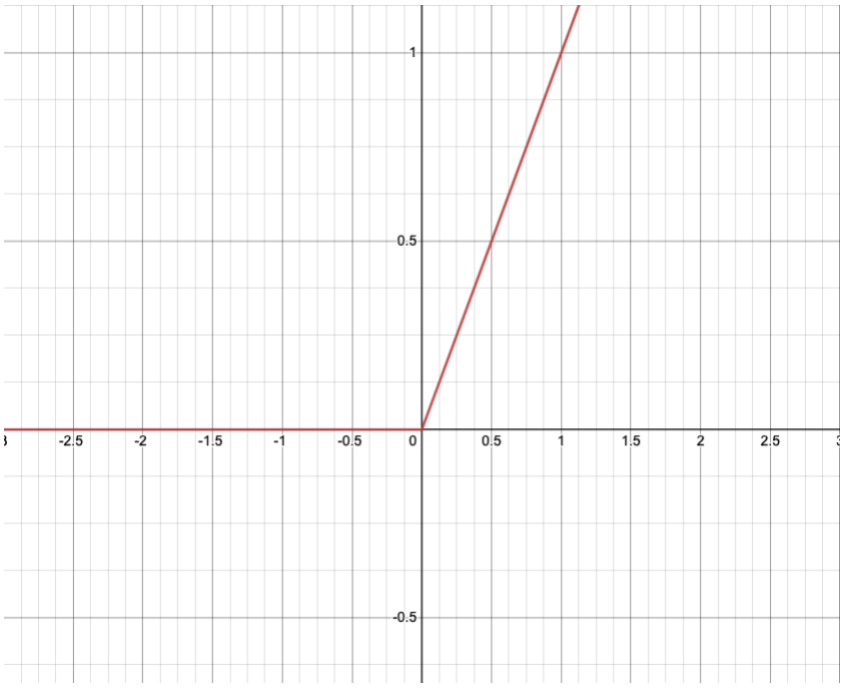

Рисунок 5.3.1 Графік функції активації ReLU

Функція ReLU стала найбільш вживаною функцією активації. Вона має всі поширені властивості функцій активації крім того, що функція не є відцентрованною та обмеженою. Не дивлячись на це, ReLU має одну значну перевагу — її продуктивність надзвичайно висока, тому що всі негативні числа прирівнюються до нуля. Тобто проміжок між отриманими вхідними даними і обробленим вихідним результатом досить короткий. Але це також спричиняє певні проблеми. Кожні негативні вхідні дані на виході перетворюється в нуль, що в майбутньому може призвести до того, що велика частина нейронів стане неактивною. Нейрона мережа стає дуже не стабільною, тому що її велика частина просто не буде працювати. Тобто функція ReLU страждає від проблеми зникаючого нейрона.

5.4 Функція LReLU

Функція активації LReLU[8] (англ. Leaky Rectified linear unit) є покращеною версією функції ReLU. Її можна представити за допомогою формули $f(x) = \begin{cases} 0.01x & x \leq 0 \\ x & x > 0 \end{cases}$

Графік функції має деякі відмінності від аналогічного графіка ReLU. Функція LReLU має такі саме властивості як ReLU, а саме неперервність і обмеженість, але до цього переліку додається ще й центрованість.

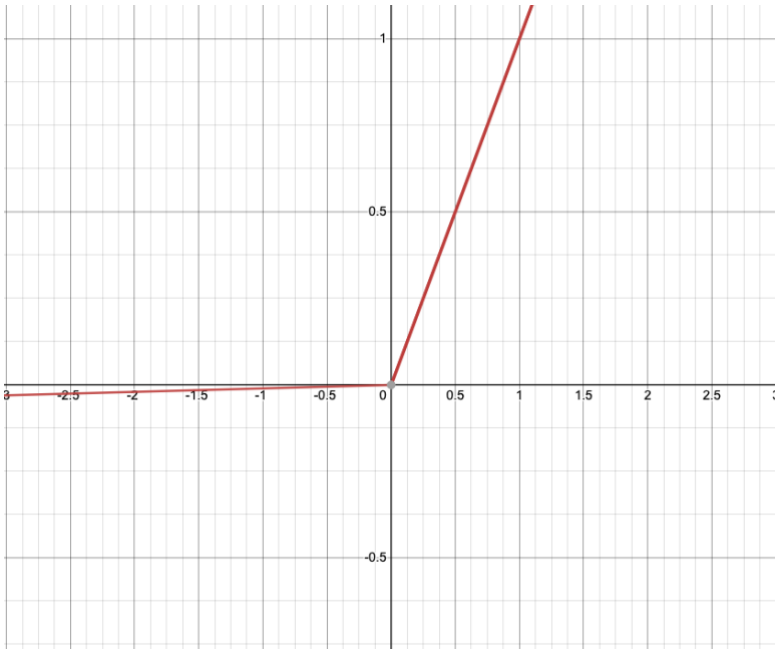

Рисунок 5.4.1 Графік функції активації LReLU

По графіку функції видно, що її область визначення містить невелику частину негативних значень, що дозволяє частково позбутись проблеми зникаючого нейрона, яка характерна для звичайної функції ReLU. Але функція на виході дає дуже маленьку від’ємну частину, а саме 0.01. Великі зміни у вхідних даних призводять до маленьких змін у вихідних даних від’ємної частини. Це може спричинити проблему зникаючого градієнту. Тому хоч дана функція активації і є покращеною версією ReLU, вона повністю не позбувається всіх її недоліків.

5.5 Функція GeLU

Функція GeLU[9] з сімейства Rectified linear unit, яка детально розшифровується як Gaussian Error Linear Unit, є відносно новою функцією, проте вона вже зарекомендовала себе як хороша функція активації. Саму функцію можна представити за формулою $f(x) = 0.5x(1 + \frac{2}{\sqrt{\pi}}\int_0^{\frac{x}{\sqrt{2}}} e^{-t^2} dt)$, і її можна зобразити за допомогою графіка.

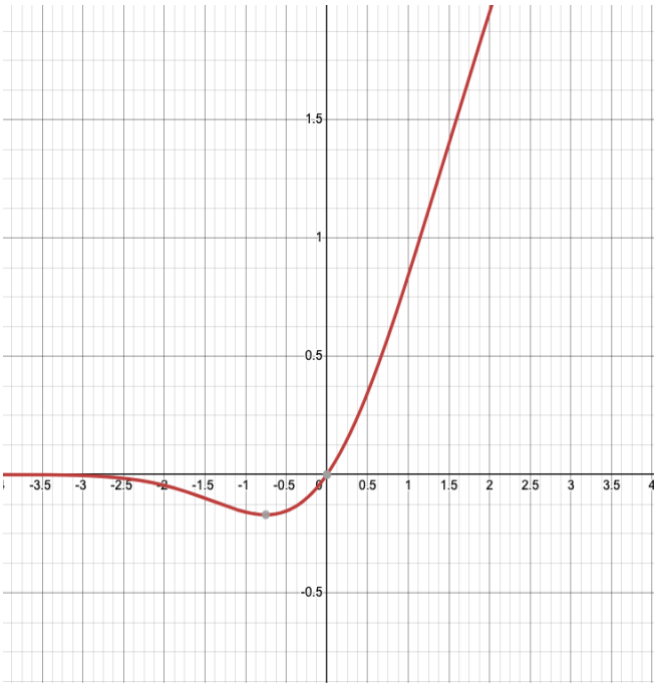

Рисунок 5.5.1 Графік функції активації GeLU

Візуально видно, що графік GeLU має заокруглення, на відміну від стандартної функції активації ReLU. Найбільшою відмінністю від ReLU є те, що функція активації може набувати негативних значень на виході. Функція є неперервною, необмеженою і центрованою. Вона частково стикається з тією самою проблемою, що і ReLU — проблемою зникнення градієнту.

5.6 Функція Swish

Функція SiLU, її найчастіше називають Swish[10] має суттєві відмінності від сімейства Rectified linear unit. Вона поєднує в собі sReLU і класичну сігмоїдну функцію активації і її можна зобразити за допомогою формули $f(x) = x \cdot sig\ (\beta x)$, де x - вхідні дані, β – або константа, або змінний параметр.

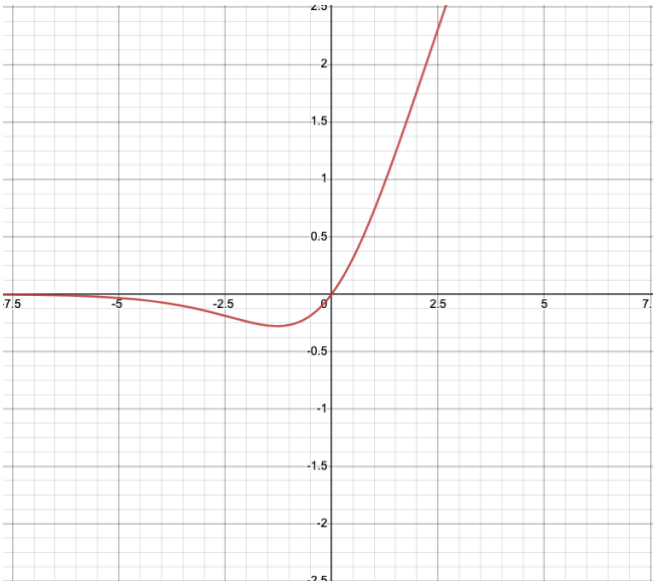

Рисунок 5.6.1 Графік функції активації Swish

Можливість мати змінний параметр, який може змінюватись під час тренування функції дає їй неабияку гнучкість. При β що дорівнює 0, функція стає звичайнісінькою лінійною функцією $f(x) = \frac{x}{2}$, а при $\beta \to \infty$, сигмоїдна частина функції повертає значення в проміжку від 0 до 1, що фактично перетворює її в ReLU. Функція є неперервною на всьому проміжку визначення, необмеженною зверху, але обмеженою знизу. Також вона є центрованою, бо повертає як від'ємні так і додатні значення. При змінному параметру β, Swish може контрольовано змінювати свою природу від лінійної функції до ReLU, що робить її дуже цікавою. Але вона також має свої недоліки, один з найбільших – це наявність в своїй основі сигмоїдної функції, яка має доволі низьку продуктивність через складність обчислень вхідних даних.

6 Датасет CIFAR-10

Датасет CIFAR-10[11] є одним з найбільш популярних датасетів, які використовуються для тренування та тестування нейронних мереж. Він був створений для тренувальних, навчальних та науково-дослідницьких проектів. Датасет представляє з себе набір з шістдесяти тисяч квадратних кольорових зображень розмірами тридцять два на тридцять два пікселя. Він розбитий у відношенні один до шести, тобто всього існує п’ятьдесят тисяч тренувальних зображень і десять тисяч тестових. Зображення згруповані на десять різних класів, пронумерованих відповідно від нуля до дев’яти. Вони є доволі стандартними, а саме: літак, автомобіль, птах, кіт, олень, пес, жаба, кінь,

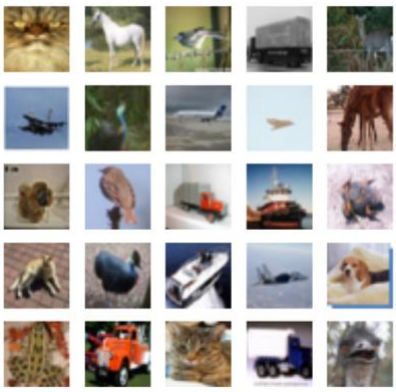

Рисунок 6.1 Приклад зображень датасету CIFAR-10

корабель та вантажівка. З переліку класів випливає, що датасет не дуже підходить для застосування у реальних проектах, де класів повинно бути набагато більше, але для експериментальних цілей даного експерименту його більш ніж достатньо. Саме тому цей датасет і був вибраний для дослідження роботи різних функцій активації в архітектурі нейронних мереж.

7 Різні архітектури нейронних мереж

Нейронні мережі є різноманітними і вони мають різне призначення і галузі використання. Саме тому існують відповідні архітектури для тієї або іншої задачі. Найбільш поширеними з них є згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), трансформерні мережі (ViT), нейроні мережі довгої короткострокової пам’яті (LSTM), нейронні мережі прямого поширення (Feedforward NN), тощо.

Для дослідження функцій активації на датасеті CIFAR-10 було прийнято рішення використовувати тільки архітекутри, які найкраще підходять для задачі класифікації графічних зображень, а саме: згорткову нейрону мережу та трансформерну мережу. Для цих нейронних мереж були підібрані моделі, які успішно зарекомендували себе і мають високу точність. Очевидно, що ці моделі не можна напряму порівнювати між собою, тому що кожна з них має власну архітектуру, складність та кількість шарів, тому було зроблено аналіз роботи функцій активації на кожній моделі нейронної мережі окремо, без їх загального порівняння.

7.1 Згорткові нейронні мережі

Згорткова нейронна мережа (англ. Convolution neural network) – нейрона мережа, основний принцип якої ґрунтується на дуже потужній математичній операції: згортці. Згорткою можна охарактеризувати доволі просту операцію, яка представляє з себе матрицю ваг, по якій проходиться ядро, поступово виконуючи операцію множення елементів матриці, які входять в область дії, над якою воно знаходиться. Після чого всі елементи сумуються і записуються в елемент нової матриці. Тим самим зменшується розмірність матриці, вона “згортається”. CNN добре працює з класифікацією зображень.

У якості моделі для тестування згорткової нейроної мережі було вирішено вибрати згорткову нейрону мережу архітектури VGG[12], а також архітектури

ResNet[13]. Не будемо поглиблюватись в детальну структуру цих мереж, вони підходходять для нашої задачі, а саме дослідження різних функцій активації.

7.2 Трансформерні мережі

Основна ідея трансформера полягає в утворенні взаємозв'язку з парами вхідних даних, в нашому випадку одиницею даних, яка передається на вхід, є зображення. Зорові трансформери, які ще скорочено називають ViT[14] (англ. Vision transformers), використовуються як аналог трансформерів для обробки природної мови або NLP (англ. Natural Language Processing).

ViT розбиває вхідне зображення на групу частинок оригінального зображення. Як приклад можна привести розбиття речення на групу слів, яке полегшує визначення його типу. Трансформери мають набагато більшу продуктивність й потребують менше вхідних даних для навчання, якщо порівнювати їх з CNN. Але на маленьких датасетах вони отримують слабші результати. Тому датасет повинен бути достатньо великим , що не дозволяє датасету CIFAR-10 показати явну перевагу візуальних трансформерів над згортковими нейронними мережами. В той час, як CNN використовує масив пікселів зображення, ViT розділяє зображення на маленькі частинки.

Для дослідження функцій активації було вирішено взяти нескладну модель архітектури ViT, яка повністю задовольняє усі наші потреби.

8 Результати дослідження

8.1 Модель VGG

У якості першої нейронної мережі була взята дев'ятишарова згорткова мережа з архітектурою VGG. Вона є класичною і чудово підходить для класифікації зображень, але все одно результати досліджень виявились доволі цікавими. Всього було проведено п'ятдесят епох навчання. Як оптимізатор

гіперпараметрів використовувався SGD з коефіцієнтом швидкості навчання рівним 0.001. Розмір тренувальних прикладів (англ. Batch size) становив шістдесят чотири одиниці. Також для цього експерименту були задіяні різні методи ініціалізації ваг, що робить його ще цікавішим. Варто додатково зазначити, що експеримент був обмежений у часі, тому кількість епох, які бралися для нього, не були достатніми для повноцінного тренування моделей, але їх результатів цілком достатньо, щоб побачити загальну динаміку кожної функції активації. Зазвичай архітектура VGG має функції активації ReLU, тому в прихованих шарах функції активації змінювалась.

Нижче приведені результати першого дослідження на архітектурі VGG.

На останньому місці сигмоїдна функція активації, яка показала, м’яко кажучи, жахливі результати. Точність моделі з двома методами ініціалізації ваг (Xavier та He) становить приблизно 10 відсотків. Нижче наведені графіки точності та втрат (англ. Cross-entropy loss), де червоним показано функцію тестового датасету, а блакитним — тестувального.

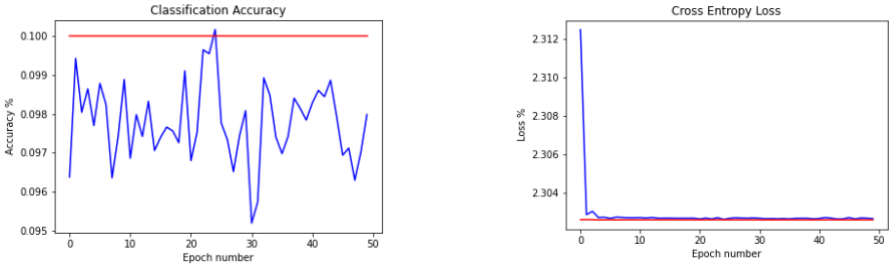

Рисунок 8.1.1 Графіки точності та втрат Sigmoid. Ініціалізація ваг He

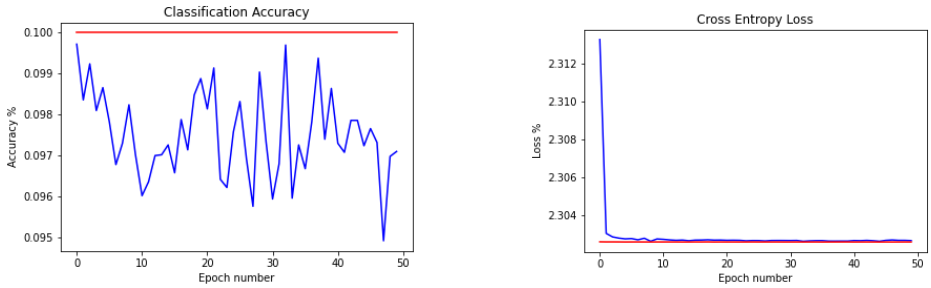

Рисунок 8.1.2 Графіки точності та втрат Sigmoid. Ініціалізація ваг Xavier

Можемо сказати, що сигмоїдна функція спричинила проблему зникнення градієнту, що в результаті призвело до неможливості нормального навчання моделі і її повної зупинки. Так як сигмоїдні функції все ще використовуються в згорткових нейронних мережах, можна припустити, що проблема більше в архітектурі VGG, яка не підходить для цієї функції активації.

Наступною функцією активації є Swish. Доволі цікавим є той факт, що ця функція активації є покращеною версією сигмоїдної функції, але її результати не є дуже поганими. Найгірше функція показала себе з методом ініціалізації ваг Xavier, в результаті навчання модель змогла показати лише 72.7 відсотка точності, що гірше за всі інші функції активації. Графіки зміни точності і функцію витрат наведено нижче.

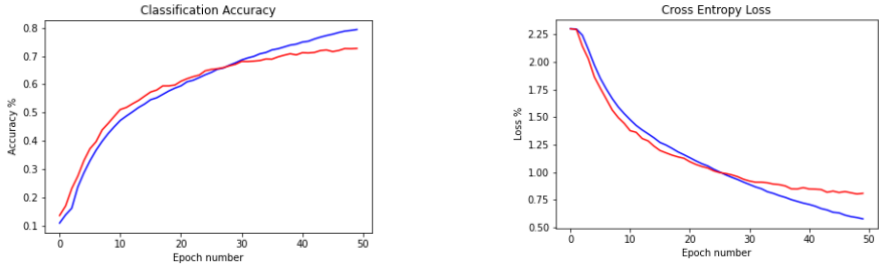

Рисунок 8.1.3 Графіки точності та втрат Swish. Ініціалізація ваг Xavier

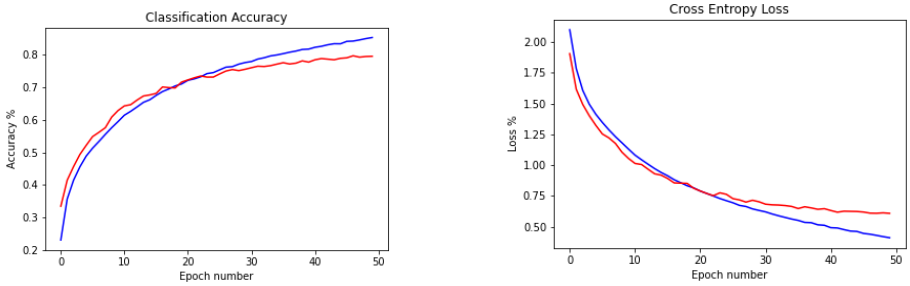

Рисунок 8.1.4 Графіки точності та втрат Swish. Ініціалізація ваг He

Куди кращі результати з методом ініціалізації ваг He. Він дозволив показали аж 79.5 відсотка точності, що є дуже гарним результатом.

Ще однією функцією, яка приємно вразила стала тагенсна функція активації. З її допомогою вийшло натренувати модель до 79 відсотків точності для метода ініціалізації ваг Xavier і 78.95 відсотків для He. Можемо припустити,

що обидва методи добре підходять для тангенсної функції активації. Графіки зміни точності і функції втрат наведено нижче.

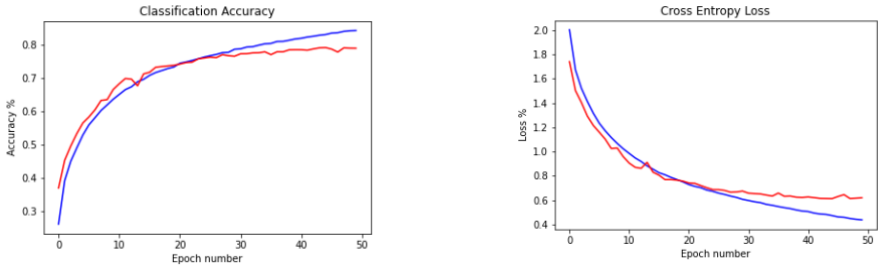

Рисунок 8.1.5 Графіки точності та втрат Tanh. Ініціалізація ваг Xavier

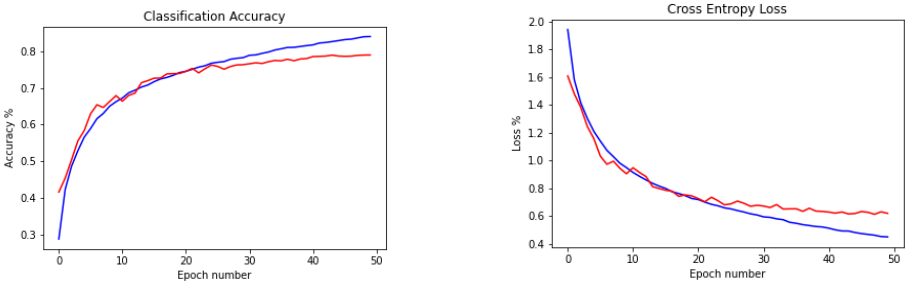

Рисунок 8.1.6 Графіки точності та втрат Tanh. Ініціалізація ваг He

Функція активації GeLU показала слабкіші результати ніж інші функції з сімейства Reclifier Linear Unit. З методом активації Xavier модель VGG змогла досягти точності лише 75,2 відсотка.

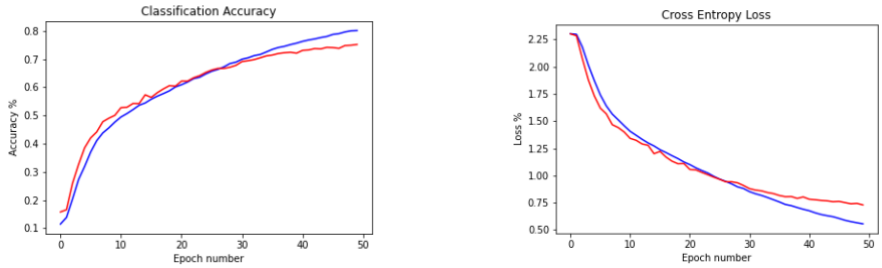

Рисунок 8.1.7 Графіки точності та втрат GeLU. Ініціалізація ваг Xavier

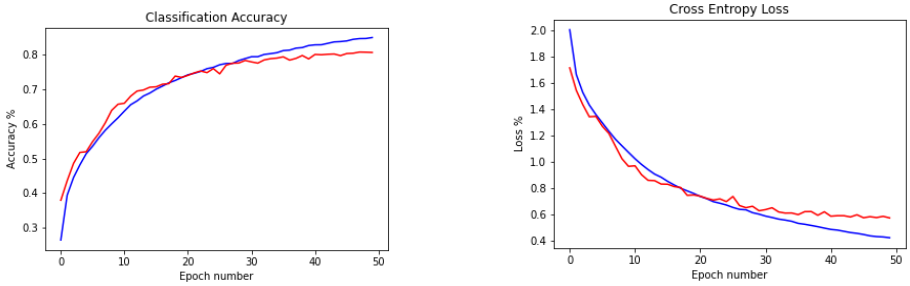

Рисунок 8.1.8 Графіки точності та втрат GeLU. Ініціалізація ваг He

А ось метод ініціалізації ваг He зміг покращити результат до 80.6 відсотка точності. Це підтверджує теорію, що метод початкової ініціалізації ваг He підходить для функцій сімейства Rectified Linear Unit набагато краще ніж метод Xavier.

Наступною є стандартна для архітектури VGG функція активації ReLU. Вона показала гарні і стабільні результати. З методом ініціалізації ваг Xavier було отримано 79.65 відсотків точності, в той час як з методом ініціалізації He – 80.15 відсотків точності. Ці результати ще раз підтверджують тезу, що He має кращу сумісність з функціями ReLU ніж Xavier. Графіки функції точності і функції втрат для кожного з методів ініціалізації наведено нижче.

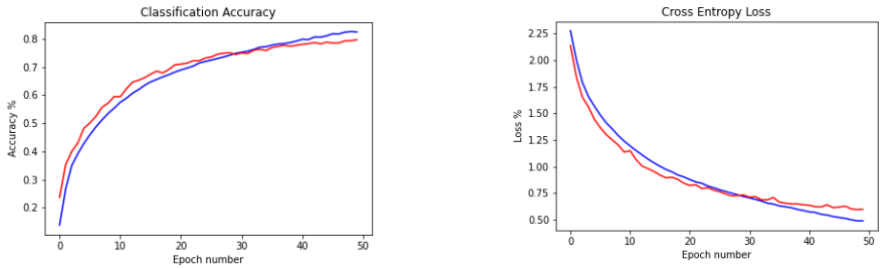

Рисунок 8.1.9 Графіки точності та втрат ReLU. Ініціалізація ваг Xavier

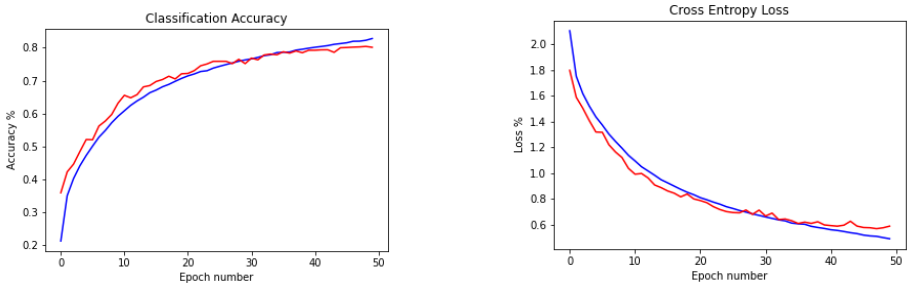

Рисунок 8.1.10 Графіки точності та втрат ReLU. Ініціалізація ваг He

Функція LReLU показала найкращі з усіх функцій активації результати. З методом ініціалізації ваг Xavier було досягнуто 80.6 відсотка точності, що є кращим результом з усіх функцій активації. Очевидно, що метод ініціалізації He дозволяє покращити цей результат до 81.84 відсотка точності.

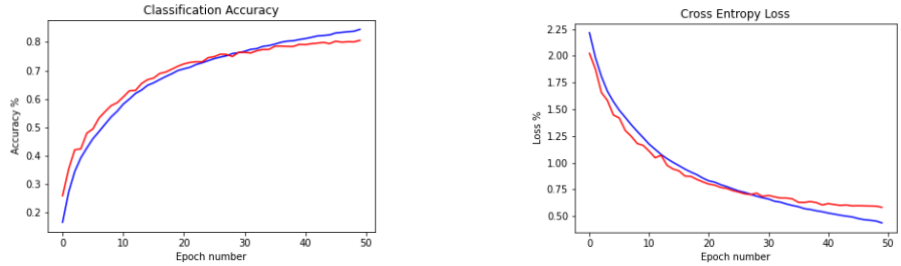

Рисунок 8.1.11 Графіки точності та втрат LReLU. Ініціалізація ваг Xavier

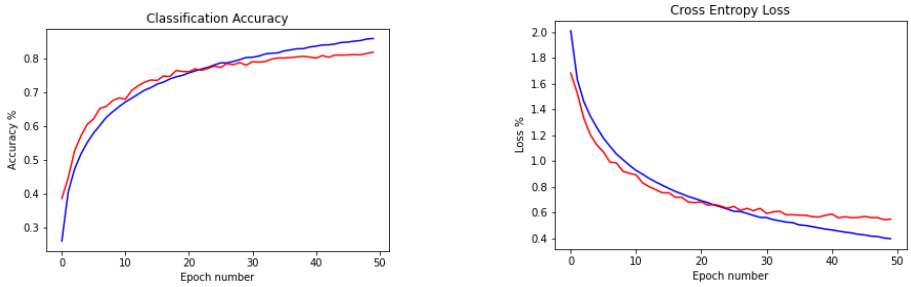

Рисунок 8.1.12 Графіки точності та втрат LReLU. Ініціалізація ваг He

Отже в результаті навчання згорткової нейронної мережі VGG на датасеті CIFAR-10, найкраще себе показала функція активації LReLU з методом ініціалізації ваг He. Водночас найгірший результат отримала сігмоїдна функція активації, можна навіть сказати, що її результат є повністю незадовільним для цього експерименту. Також доволі спірним рішенням є використання методу ініціалізації ваг Xavier для функцій сімейства Reclifier Linear Unit, а також для функції активації Swish. Результати є значно нижчими ніж при використанні метода He.

8.2 Модель ViT

Для другого експерименту була вибрана проста модель ViT з зовсім іншою, трансформерною архітектурою, яка повністю відрізняється від звичайної згорткової нейронної мережі. Модель є двошаровою, з восьми трансформерними шарами. Коефіцієнт швидкості навчання брався такий само, а саме 0.001. У якості оптимізатора гіперпараметрів був взятий AdamW з обрізанням ваг на рівні

0.0001. Розмір тренувальних прикладів становив тридцять дві одиниці. Для того щоь натренуватися, модель отримала тридцять епох навчання. Було вирішено взяти саме таку кількість епох, тому що дане дослідження не має мети натренувати модель до найбільшої точності, адже це займає багато часу. Метою цього дослідження є перевірка різних функцій активації і пошук залежностей між ними. Варто додатково зазначити, що результати цього експерименту не показують усіх можливостей моделі ViT, і якщо би модель мала, припустимо, сто епох навчання, то результати могли би бути набагато кращими. Як вже було зазначено, отримання максимального результату від моделі не є основною метою даного дослідження. В цьому дослідженні не розглядалися різні методи ініціалізації ваг, тому головний акцент ставився саме на функціях активації.

Як не дивно найгірший результат показала функція активації Swish. Модель змогла отримати лише 53.1 відсотка точності за тридцять епох навчання, що безумовно є слабким результатом в порівнянні з іншими моделями. Можливо, що функція активації Swish не дуже гарно вписується в архітектуру зорових трансформерів, або вона постраждала від проблеми зникаючого градієнта. Також можливо коефіцієнти навчання для цієї функції активації були не задовільними. Але факт залишається фактом, Swish отримала найгірший результат з-поміж усіх функцій активації, які бралися для цього експерименту.

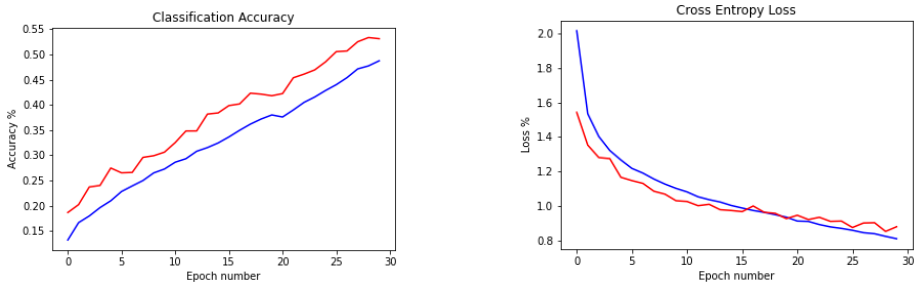

Рисунок 8.2.1 Графіки точності та втрат Swish.

Ще однією функцією активації, яка не змогла показати пристойний результат є LReLU з її 54.2 відсотка точності. Можемо зробити висновки, що функції активації LReLU і Swish не дуже підходять для цієї конкретної

архітектури ViT з поточним коефіцієнтом навчання. Графіки залежності, точності і втрат від епохи наведені нижче.

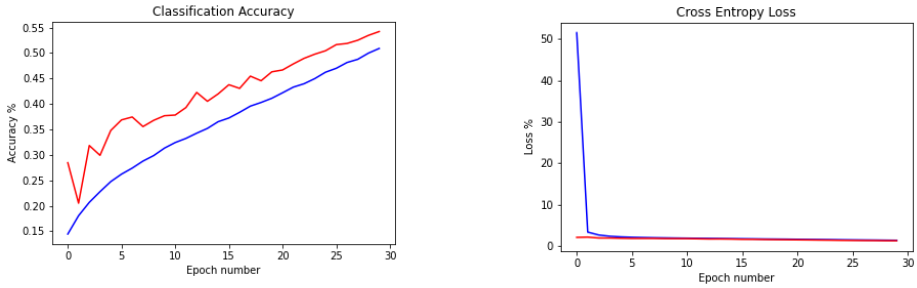

Рисунок 8.2.2 Графіки точності та втрат LReLU.

Доволі непоганий результат показала тангенсна функція активації. За тридцять епох навчання модель змогла натренуватися і отримати точність 62.9 відсотка, що безумовно не є слабким результатом в порівнянні з іншими моделями. Нижче наведено графіки точності і втрат, які показують, як саме змінювались ці значення на кожному етапі навчання моделі ViT.

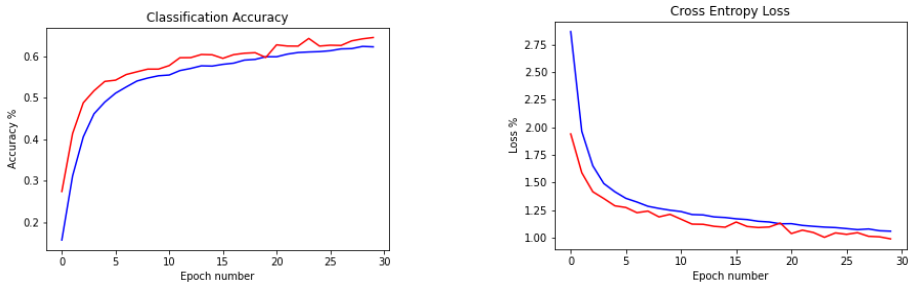

Рисунок 8.2.3 Графіки точності та втрат Tanh.

Сігмоїдна функція активації показала чудовий результат. Вона змогла покращити показники тангенсної функції активації і допомогла натренувати модель до 68.7 відсотків точності. Ця класична і проста функція обійшла по результатам такі функції як LReLU та Swish і змогла показати, що інколи навіть класичні функції активації на деяких архітектурах моделей можуть працювати набагато краще ніж більш сучасні і потужні функції.

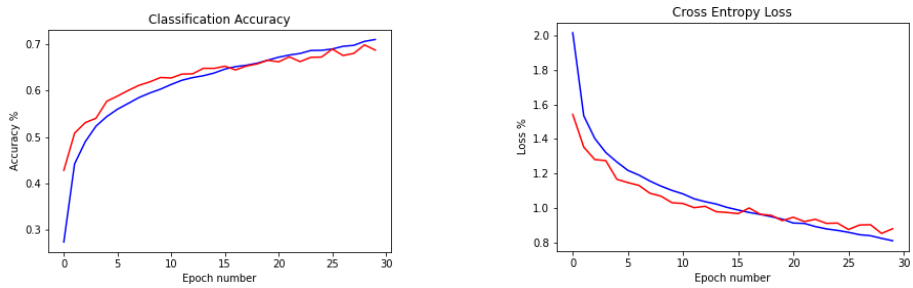

Рисунок 8.2.4 Графіки точності та втрат Sigmoid.

Наступною функцією активації є ReLU. Її результати є стабільними, вже другий експеримент поспіль. Маючи в запасі тридцять епох навчання, функція активації ReLU допомогла натренувати модель ViT до 71.75 відсотка точності. Якщо розглядати цей результат як результат робочої моделі, то безумовно він є слабким, але на меті не було показати найкращі можливості моделі, лише залежність цього результату від інших функції активації. Графік результатів наведений нижче.

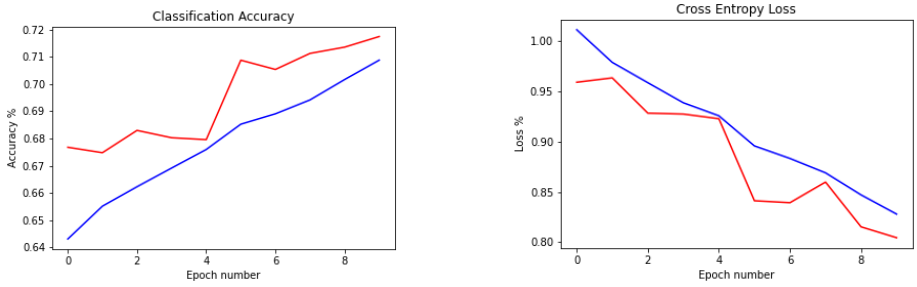

Рисунок 8.2.5 Графіки точності та втрат ReLU.

Функція активації GeLU показала найкращий результат з поміж усіх інших функцій, а саме 72.9 відсотка. Тому можемо константувати факт, що вона найкраще підходить для моделі ViT поданої архітектури.

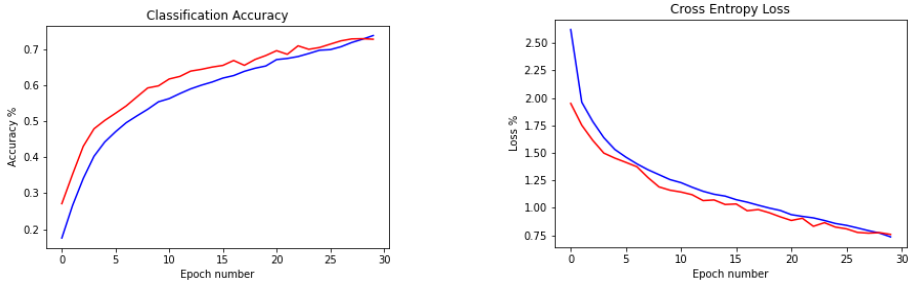

Рисунок 8.2.6 Графіки точності та втрат GeLU.

В результаті цього експерименту за допомогою моделі зорових трансформерів ViT та датасету CIFAR-10 було досліджено 6 функцій активації, які показали різні результати. Найгіршими функціями активації стали LReLU та Swish, що є доволі не очікуваним результатом, а найкращою – функція активації GeLU.

8.3 Модель ResNet

У якості фінального експерименту була взята п’ятдесяти шарова згорткова нейронна мережа ResNet50. На цей раз коефіцієнт швидкості навчання становив 0.0001, в ролі оптимізатора гіперпараметрів виступив оптимізатор Adam. Розмір тренувальних прикладів (англ. Batch size) був таким само як у згортковій нейронній мережі VGG і складав шістдесят чотири одиниці. В цей раз було прийнято рішення взяти сто епох для навчання моделі. Наголошується, що підібрані параметри, були далекі від ідеалу, а кількість епох навчання була недостатньою для такої глибокої нейронної мережі, тому це напряму вплинуло на загальні результати. Датасет CIFAR-10 з зображеннями розмірами тридцять два пікселя на тридцять також не дуже підходить для архітектури цієї моделі, в якій стандартною вхідною розмірністю є кортеж (224, 224, 3), якщо основною метою є отримання найбільшої точності. Але основною метою була перевірка можливостей і результатів кожної функції активації, тому загальна точність моделі не грає такої великої ролі. Варто зазначтити, що в цілому всі результати

були доволі низькими, але це не забороняє їй виконувати основну задачу — порівнювати функції активації між собою.

Провальні результати отримали класичні функції активації, а саме сигмоїдна та тангенсна функції активації. Ці функції не дозволили моделі нормально натренуватися та спричинили проблему зникнення градієнту. Маємо певну закономірність, що сігмоїдна функція активації вже вдруге майже не дає нормальних результатів саме на згортковій нейронній мережі. Графіки функцій наведені нижче і на них видно, що точність моделі ResNet варіюється в межах 10 відсотків, що є провальним результатом.

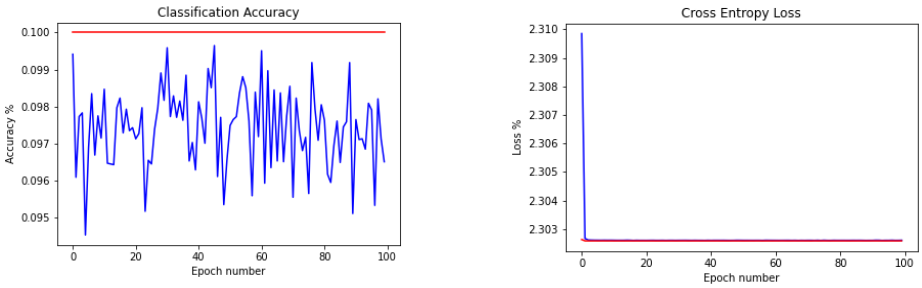

Рисунок 8.3.1 Графіки точності та втрат Sigmoid.

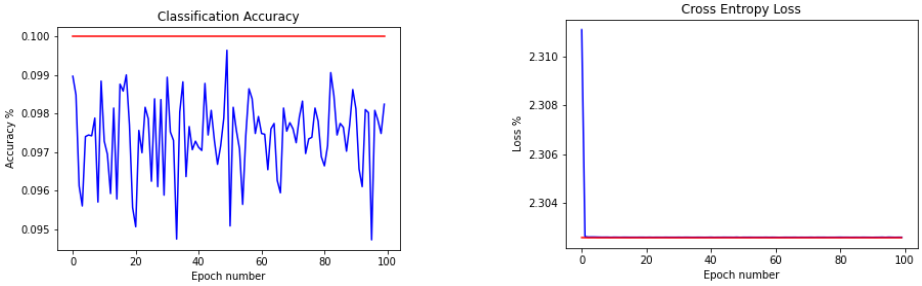

Рисунок 8.3.2 Графіки точності та втрат Tanh.

Наступною функцією активації є Swish. Після ста епох навчання, модель з цією функцією активації змогла отримати точність лише 54.7 відсотки. Цей результат можна вважати найгіршим, тому що класичні моделі взагалі не змогли натренувати модель. Нижче можна побачити графіки точності та втрат моделі.

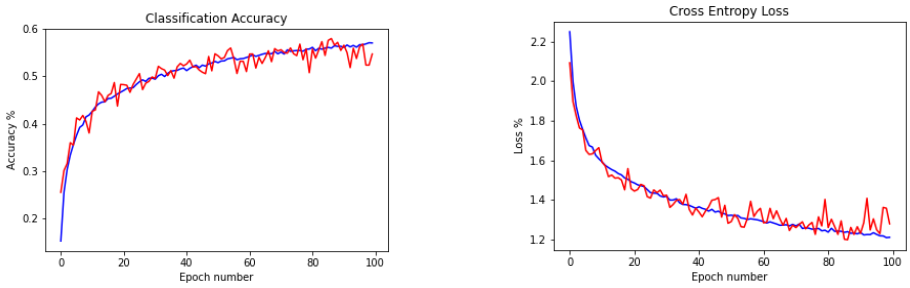

Рисунок 8.2.3 Графіки точності та втрат Swish.

Функція активації ReLU є стандартною функцією активації в згортковій нейронні мережі ResNet50, тому при проведенні експерименту не потрібно було її змінювати. Вона змогла показати результат на рівні 57.2 відсотка. Цей результат є доволі непоганим, враховуючи загальні результати різних функцій активації. Графіки зміни точності та втрат наведені знизу.

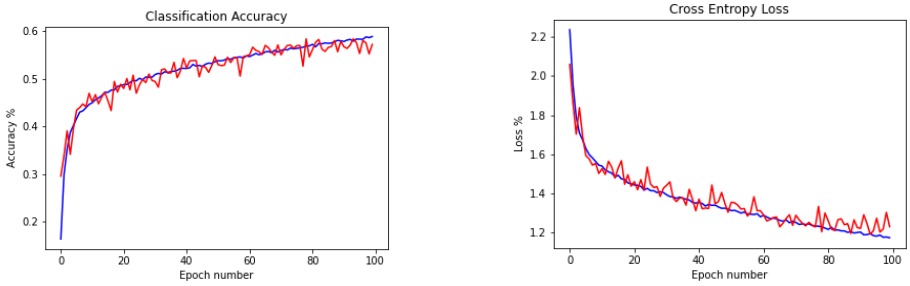

Рисунок 8.2.4 Графіки точності та втрат ReLU.

Більш точний результат отримала функція активації LReLU. З її допомогою за сто епох модель змогла натренуватися до 59.8 відсотків точності. Бачимо, що стандартна функція ReLU, яка по замовчуванню наявна в моделі ResNet50 програє іншим функціям активації з сімейства Rectified Linear Unit. Нижче можна побачити графіки точності та втрат.

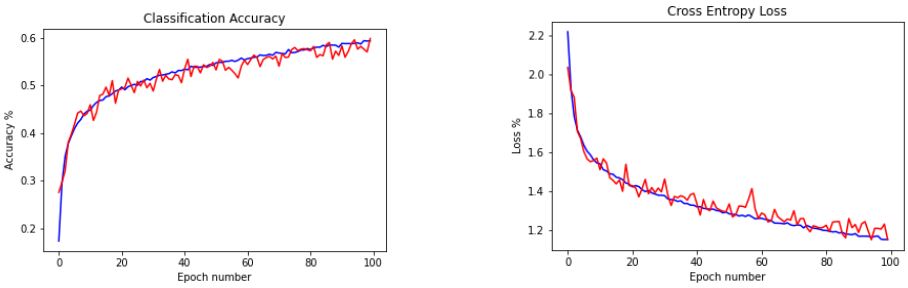

Рисунок 8.2.5 Графіки точності та втрат LReLU.

Найкращий результат вже у двох різних експериментах змогла отримати функція активації GeLU. Модель, яка містила цю функцію, за сто епох натренувалась до точності 60.2 відсотки, що не набагато краще за результати LReLU. Якщо дивитися на цю точність під призмою практичного використання моделі, то вона є дуже маленькою, але це не заважає робити певні висновки з цього експерименту. Графіки втрат та точності, які змінювались в моделі під час усього її навчання можна побачити нижче.

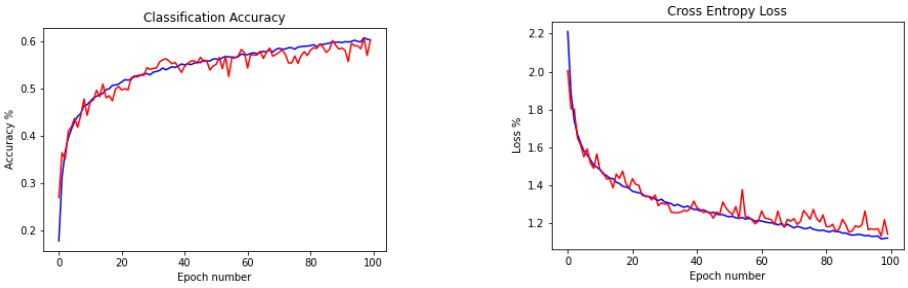

Рисунок 8.2.6 Графіки точності та втрат GeLU.

Тож класичні функції активації, а саме тангенсна і сигмоїдна, показали свою повну неспроможність натренувати модель. Найгірший після них результат отримала функція активації Swish, а найкраще показала себе LReLU та GeLU.

8.4 Загальні результати

Функція активації	VGG (He) Accuracy	VGG (Xavier) Accuracy	ViT Accuracy	ResNet Accuracy
Sigmoid	0.1000	0.1000	0.6872	0.1000
Tanh	0.7895	0.7903	0.6289	0.1000
Swish	0.7956	0.7273	0.5212	0.5473
ReLU	0.8015	0.7965	0.7174	0.5716
LReLU	0.8184	0.8060	0.5421	0.5980
GeLU	0.8066	0.7520	0.7290	0.6018

Висновки

Результатом виконаної роботи став огляд найбільш популярних функцій активації штучних нейронних мереж та аналіз їх ефективності на різних архітектурах нейронних мереж з різними методами ініціалізації ваг.

Було досліджено, що функції активації напряму залежать від архітектури штучної нейронної мережі, її конкретної моделі, а також метода ініціалізації ваг.

У ході проведеного експерименту було визначено, що найбільш стабільною функцією активації залишається ReLU, яка на всіх моделях показала хороший результат, а найбільш спірною є функція активації сигмоїд, яка аж на двох моделях не змогла допомогти натренувати модель. В той час найбільш результативною стала функція активації GeLU, показавши на двох різних моделях найкращий результат.

Якщо підбивати підсумки, то з отриманих результатів можна зробити висновки щодо функцій активації, розташувавши їх від найгіршої до найкращої.

Сигмоїдна функція активації показала найбільш слабкі результати і рекомендована для використання лише в специфічних випадках. Найкраще вона працює з трансформерною мережею моделі ViT, тоді як як з згортковими нейронними мережами моделей ResNet та VGG вона взагалі не працює.

Функція активації Swish, показала гарний результат лише на згортковій нейроній мережі моделі VGG з методом ініціалізації ваг He. На моделі трансформерної архітектури ViT, а також на моделі згорткової нейроної мережі ResNet функція показала слабші результати ніж інші функції активації. Тобто функція гарно спрацювала один раз із трьох, і то лише з одним методом активації ваг.

Тангенсна функція активації отримала стабільні результати на двох методах ініціалізації ваг моделі VGG, її результат на трансформерній мережі моделі ViT також є хорошим. Але функція активації повністю провалилась в

експерименті з моделю ResNet. Можна казати, що в цілому ця функція показує гарні результати на обох архітектурах штучних нейронних мереж, але існують моделі, окрім ResNet, з якими категорично не рекомендовано використовувати цю функцію активації.

Найбільш популярна функція активації ReLU отримала хороші результати у всіх дослідженнях, що робить її найбільш стабільною функцією активації. Саме тому вона найчастіше присутня в будь-якій стандартній моделі штучної нейронної мережі в якості основної функції активації. Вона рекомендована для використання, але наступні дві функції в більшості випадках можуть досягти кращих результатів ніж ReLU.

Функція активації LReLU отримала дуже найкращі результати на моделі згорткової штучної нейроної мережі VGG, а також на моделі ResNet отримала майже найкращі результати. В той час як з трансформерною архітектурою ця функція активації не змогла показати пристойний результат, саме тому її не рекомендовано до використання для тренування моделі ViT.

Найкращі результати в двох експериментах із трьох показала функція активації GeLU. Вона суттєво просіла лише на моделі VGG з методом ініціалізації ваг Xavier, який працює гірше з усіма функціями сімейства Rectified Linear Unit ніж He. Тому з вірним методом ініціалізації ваг цю функцію можна рекомендувати для використання з метою покращення загальних результатів моделі і використовувати її як заміну ReLU.

Перспективами розширення даного дослідження є пошук нових параметрів, від яких залежить вибір функції активації, а саме: датасет, складність задачі, розмір вхідних даних. А також збільшення загального переліку функцій активації, архітектур штучних нейроних мереж і самих моделей, на яких і досліджуються ці функції активації.

Список літератури

1. Simon Haykin. Neural Networks: A Comprehensive Foundation. Prentice Hall, 1999.

2. Bishop, C. (2006). Pattern Recognition and Machine Learning. Springer.

3. Y Bengio and X Glorot. Understanding the difficulty of training deep feed forward neural networks. International Conference on Artificial Intelligence and Statistics, pages 249–256, 01 2010.

4. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. IEEE International Conference on Computer Vision (ICCV 2015), 1502, 02 2015.

5. K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. Neural Netw., 2(5):359–366, July 1989.

6. A Vehbi Olgac and Bekir Karlik. Performance analysis of various activation

functions in generalized mlp architectures of neural networks. International

Journal of Artificial Intelligence and Expert Systems, 1:111–122, 02 2011.

7. Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2018.

8. Andrew Maas, Awni Hannun , Andrew Ng. Rectifier nonlinearities improve neural network acoustic models, 2013.

9. Dan Hendrycks, Kevin Gimpel. Gaussian Error Linear Units (GELUs), 07 2020.

10. Prajit Ramachandran, Barret Zoph, Quoc V. Le. Searching for Activation Functions, 10 2017.

11. The CIFAR-10 dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.cs.toronto.edu/~kriz/cifar.html.

12. Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 09 2014.

13. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, 12 2015.

14. Alexey Dosovitskiy, Lucas Beyer, Xiaohua Zhai, Dirk Weissenborn and Jian Sun. An image is worth 16x16 words: Transformers for image recognition at scale. International Conference on Learning (ICLR 2021), 2021.

Додаток А (обов’язковий). Модель VGG

```
def VGG3(func, initializer):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation=func, kernel_initializer=initializer, padding='same',
    input_shape=Config.input_shape)
    model.add(Conv2D(32, (3, 3), activation=func, kernel_initializer=initializer, padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(64, (3, 3), activation=func, kernel_initializer=initializer, padding='same'))
    model.add(Conv2D(64, (3, 3), activation=func, kernel_initializer=initializer, padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(128, (3, 3), activation=func, kernel_initializer=initializer, padding='same'))
    model.add(Conv2D(128, (3, 3), activation=func, kernel_initializer=initializer, padding='same'))
    model.add(MaxPooling2D((2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    opt = SGD(learning_rate=Config.learning_rate, momentum=0.9)
    model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

Додаток В (обов'язковий). Модель ViT

```
def ViT(func):

    with strategy.scope():

        inputs = layers.Input(shape=Config.input_shape)
        augmentation_layer = tf.keras.Sequential([
            keras.layers.Input(Config.input_shape),
            keras.layers.experimental.preprocessing.Normalization(),
            keras.layers.experimental.preprocessing.Resizing(Config.image_size, Config.image_size),
            keras.layers.experimental.preprocessing.RandomRotation(factor=0.02),
            keras.layers.experimental.preprocessing.RandomZoom(height_factor=0.2, width_factor=0.2),
        ])
        augmented = augmentation_layer(inputs)

        patches = Patches(Config.patch_size)(augmented)
        encoder_patches = PatchEncoder(Config.num_patches, Config.projection_dim)(patches)

        for _ in range(Config.transformer_layers):
            x1 = layers.LayerNormalization(epsilon=1e-6)(encoder_patches)
            attention_output = layers.MultiHeadAttention(
                num_heads=Config.num_heads,
                key_dim=Config.projection_dim,
                dropout=0.1
            )(x1, x1)
            x2 = layers.Add()([attention_output, encoder_patches])

            x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
            x3 = mlp(x3, func, hidden_units=Config.transformer_units, dropout_rate=0.1)
            encoder_patches = layers.Add()([x3, x2])

        representation = layers.LayerNormalization(epsilon=1e-6)(encoder_patches)
        representation = layers.Flatten()(representation)
        representation = layers.Dropout(0.5)(representation)

        features = mlp(representation, func, hidden_units=Config.mlp_head_units, dropout_rate=0.5)

        outputs = layers.Dense(Config.num_classes)(features)

        model = keras.Model(inputs=inputs, outputs=outputs)
    return model
```

Додаток С (обов’язковий). Модель ResNet50

```
def ResNet50():
  base_model = resnet50.ResNet50(include_top=False, weights='imagenet', input_shape=Config.input_shape,
  classes=Config.num_classes)
  base_model.trainable = False
  last_layer = base_model.get_layer('conv3_block1_3_conv')
  last_output = last_layer.output
  flat = tf.keras.layers.Flatten()(last_output)
  dense1 = tf.keras.layers.Dense(1024, activation='relu')(flat)
  dense2 = tf.keras.layers.Dense(512, activation='relu')(dense1)
  predictions = tf.keras.layers.Dense(10, activation='softmax')(dense2)
  model = Model(inputs = base_model.input, outputs = predictions, name="cifar10_model")
  optimizer = tf.keras.optimizers.Adam(learning_rate=Config.learning_rate)
  model.compile(
    optimizer=optimizer,
    loss='categorical_crossentropy',
    metrics=["accuracy"]
  )
  return  model
```