ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

МОДЕЛІ ТА МЕТОДИ ПРЕДСТАВЛЕННЯ ЗНАНЬ І ШТУЧНОГО ІНТЕЛЕКТУ:

методичні рекомендації до виконання індивідуального розрахункового завдання за освітньо-професійною програмою другого (магістерського) рівня спеціальності 122 «Комп'ютерні науки»

Рекомендовано Науково-методичною радою ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
(протокол № 6 від «24» травня 2024 р.)
Обов'язково до розміщення в репозитарії

Запоріжжя 2024

metinvest polytechnic

Моделі та методи представлення знань і штучного інтелекту: методичні рекомендації до виконання індивідуального розрахункового завдання за освітньо-професійною програмою другого (магістерського) рівня спеціальності 122 «Комп'ютерні науки» / Уклад. Шматко О.В., Горбач Т.В., Держевецька М.А. Запоріжжя, ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА», 2024. 50 с.

Методичні вказівки включають тематику індивідуальних завдань, методичні пояснення щодо порядку їх виконання, критерії оцінювання виконаного індивідуального завдання, вимоги до його оформлення, включаючи зразок звіту.

Рекомендовано для студентів спеціальності 122 «Комп'ютерні науки» другого (магістерського) рівня освіти.

Самостійне електронне текстове мережеве видання

Затверджено на засіданні кафедри
цифрових технологій та проєктно-
аналітичних рішень
Протокол № 6 від «06» лютого 2024р.

Узгоджено:
Секретар Редакційної ради

_____ Малій Х. В.
«08» травня 2024 р.

ЗМІСТ

ВСТУП

Індивідуальне розрахункове завдання (надалі - ІНДРЗ) виконується з актуальних проблем пов'язаних з методами та моделями штучного інтелекту та являє собою самостійне дослідження, яке є важливою ланкою у системі опанування загальних та фахових компетенцій здобувачами ОПП «Комп'ютерні науки та цифровий інтелект» спеціальності 122 Комп'ютерні науки усіх форм навчання.

Методичні вказівки укладено на підставі Стандарту вищої освіти за спеціальністю 122 Комп'ютерні науки галузі знань 12 Інформаційні технології для другого (магістерського) рівня вищої освіти, затвердженого Наказом Міністерства освіти і науки України від 28.04.2022 р. № 393.

В процесі виконання ІНДРЗ передбачається поєднання теоретичних знань і практичних умінь, набутих здобувачами освіти в результаті вивчення кількох дисциплін професійної підготовки магістра, а саме:

1) Інтелектуальний аналіз даних;
2) Основи штучних нейронних мереж;

ІНДРЗ – це індивідуальне завдання, яке є творчим та аналітичним рішенням конкретних завдань з використанням комплексу методів та моделей штучного інтелекту, алгоритмів машинного навчання, методів глибинного навчання, виконане здобувачем вищої освіти самостійно під керівництвом викладача згідно із поставленими завданнями.

Виконання ІНДРЗ сприяє розширенню та поглибленню теоретичних знань, розвитку навичок їх практичного використання, формує вміння самостійного розв'язання конкретних професійних завдань, створює наукове підґрунтя для виконання кваліфікаційної роботи магістра.

Мета індивідуального розрахункового завдання – поглиблення теоретичних знань та закріплення практичних навичок самостійного аналізу проблем та задач машинного та глибинного навчання з метою створення, навчання та оптимізації моделей нейронних мереж для вирішення задач класифікації, кластеризації та регресійного аналізу ІНДРЗ спрямована на розвиток аналітичного мислення та навичок використання моделей та методів штучного інтелекту для вирішення інтелектуальних задач для окремих сфер діяльності та підприємства в цілому.

Для досягнення цієї мети необхідно поставити та вирішити такі завдання:

– сформулювати постановку задачі з дослідження проблемної ситуації (практичного завдання) відповідно до обраної теми ІНДРЗ;

– сформувати інформаційно-аналітичну базу, на основі якої і з застосуванням сучасних методів і алгоритмів штучного інтелекту, дослідити теоретичні передумови розв'язання проблемної ситуації (практичного завдання), виконати аналіз наукової літератури з обраної тематики; вивчити основні теоретичні поняття, методи та моделі, які використовуються у вашій обраній області.

– зібрати відповідний набір даних для дослідження; провести попередню обробку та очищення даних: нормалізацію, заповнення пропусків, видалення шумів тощо.);

– за результатами дослідження обрати тип нейронної мережі, яка найкраще підходить для поставленої задачі (CNN, RNN, LSTM, GAN тощо); розробити архітектуру моделі, визначивши кількість шарів, нейронів, типи активаційних функцій; підібрати гіперпараметри моделі (швидкість навчання, розмір пакету, кількість епох тощо); навчити модель на підготовлених даних, використовуючи крос-валідацію або розділ на тренувальний та тестовий набори; оцінити продуктивність моделі за допомогою відповідних метрик (точність, відгук, F1-оцінка, AUC тощо); проаналізувати отримані результати, виявити можливі недоліки та шляхи їх усунення;

– представити керівнику у встановлений термін результати індивідуального розрахункового завдання, у якому у логічній послідовності відобразити основні етапи і результати дослідження, обґрунтувати методи, засоби, інструменти вирішення проблемної ситуації (практичного завдання), а також очікувані результати і рекомендації і пропозиції вирішення проблемної ситуації (практичного завдання) із застосуванням методів штучного інтелекту;

– підготувати презентацію результатів виконання індивідуального розрахункового завдання у вигляді проєкту і продемонструвати вміння обґрунтовано і коректно викладати та відстоювати власну позицію перед професійною аудиторією та експертами з інших галузей знань під час захисту.

Дисципліна спрямована на отримання здобувачами наступних загальних та спеціальних (фахових) компетентностей:

- ЗК01. Здатність до абстрактного мислення, аналізу та синтезу.
- ЗК02. Здатність застосовувати знання у практичних ситуаціях.
- ЗК05. Здатність вчитися й оволодівати сучасними знаннями.
- ЗК06. Здатність бути критичним і самокритичним.
- СК01. Усвідомлення теоретичних засад комп'ютерних наук.

- СК02. Здатність формалізувати предметну область певного проєкту у вигляді відповідної інформаційної моделі.
- СК03. Здатність використовувати математичні методи для аналізу формалізованих моделей предметної області.
- СК06. Здатність застосовувати існуючі і розробляти нові алгоритми розв'язування задач у галузі комп'ютерних наук.
- СК09. Здатність розробляти та адмініструвати бази даних та знань.
- СК13. Здатність до аналітичного мислення та проведення досліджень у сфері розробки, удосконалення та впровадження цифрового інтелекту у систему управління підприємством, бізнес-процесами й виробничими процесами.

У результаті виконання індивідуального розрахункового завдання здобувач вищої освіти повинен продемонструвати достатній рівень сформованості наступних програмних результатів навчання:

- РН1. Мати спеціалізовані концептуальні знання, що включають сучасні наукові здобутки у сфері комп'ютерних наук і є основою для оригінального мислення та проведення досліджень, критичне осмислення проблем у сфері комп'ютерних наук та на межі галузей знань.
- РН2. Мати спеціалізовані уміння/навички розв'язання проблем комп'ютерних наук, необхідні для проведення досліджень та/або провадження інноваційної діяльності з метою розвитку нових знань та процедур.
- РН6. Розробляти концептуальну модель інформаційної або комп'ютерної системи.
- РН7. Розробляти та застосовувати математичні методи для аналізу інформаційних моделей.
- РН11. Створювати нові алгоритми розв'язування задач у сфері комп'ютерних наук, оцінювати їх ефективність та обмеження на їх застосування.
- РН12. Проектувати та супроводжувати бази даних та знань.
- РН21. Аналізувати існуючи цифрові технології, проектувати, розробляти та впроваджувати на підприємствах різних галузей економіки систем цифрового інтелекту, використовуючи сучасні знання бізнес-аналізу, методів інтелектуальної обробки даних, моделей та технологій видобування знань предметної області.

1. ЗАГАЛЬНІ МЕТОДИЧНІ РЕКОМЕНДАЦІЇ З ВИКОНАННЯ ІНДИВІДУАЛЬНОГО РОЗРАХУНКОВОГО ЗАВДАННЯ СТУДЕНТА

Самостійна робота студентів (СРС) займає провідне місце у системі сучасної вищої освіти. З усіх видів навчальної діяльності СРС значною мірою забезпечує формування самостійності як провідної риси особистості студента.

Самостійна робота завершує завдання усіх інших видів навчальної діяльності. Адже знання, що не стали об'єктом власної діяльності, не можуть вважатися дійсним надбанням людини. Тому СРС має навчальне, особисте та суспільне значення.

СРС – це багатоаспектне та поліфункціональне явище з двоєдиністю цілей:

- формування самостійності студента;
- розвиток здібностей, вмінь, знань та навичок студентів.

Завдяки СРС відбувається перехід від переважно виконавчої репродуктивної діяльності студентів до пошукового, творчого начала на всіх етапах навчання у ВНЗ.

Індивідуальне розрахункове завдання (ІНДРЗ) з дисципліни «Моделі та методи представлення знань і штучного інтелекту» припускає її здійснення в наступних видах: самостійне вивчення теоретичного матеріалу, самостійне виконання практичних завдань для більш глибокого засвоєння матеріалу.

Метою виконання ІНДРЗ є більше глибоке вивчення сфери застосування та можливостей, які надають моделі та методи штучного інтелекту та набуття практичних навичок роботи з моделями нейронних мереж.

Правильна організація самостійної роботи необхідна для більш повного оволодіння дисципліною та визначає успішність здачі заліку й наступної практичної діяльності.

Самостійна робота виконується студентами під керівництвом викладача, який здійснює аудиторну роботу в навчальній групі.

Самостійна робота студентів повинна мати такі головні ознаки:

- бути виконаною особисто студентом;
- бути закінченою розробкою, де розкриваються й аналізуються актуальні проблеми з певної теми або її окремих аспектів;

- демонструвати достатню компетентність автора в розкритті питань, що досліджуються;
- мати навчальну, наукову, й/або практичну спрямованість і значимість;
- містити певні елементи новизни;
- самостійна індивідуальна розрахункова робота оформляється відповідно до вимог кафедри.

1.1 Основні вимоги до виконання індивідуального розрахункового завдання

Перед виконанням самостійної роботи потрібно повністю ознайомитися зі змістом завдання, підібрати потрібну літературу, визначити усі параметри виконання індивідуального завдання.

Результатом виконання самостійної роботи є звіт, який виконується з використанням комп'ютерної техніки та надрукований на папері формату А4. Оформлення звіту: шрифт − Arial; розмір шрифту − 14 кегель; інтервал між рядками − півтора; абзац − 12,5 мм, поля: верхнє і нижнє – 20 мм, ліве – 25 мм, праве – 15 мм; нумерація сторінок − по центру нижнього поля. Зразок оформлення звіту наведено у додатку А.

Після перевірки кожного завдання викладачем студент зобов'язаний усунути допущені помилки, інакше він не допускається до виконання наступного завдання.

Усі види самостійної роботи повинні бути здані у встановлений графіком термін. Викладач фіксує факт здачі кожної роботи та виставляє оцінку в журнал.

Поради із планування й організації часу, необхідного для виконання самостійної роботи

Раціональне планування і організація самостійної роботи студентів є найважливішою умовою її ефективності.

Планування самостійної роботи направлено на формування логічно вибудуваної, прозорої, зрозумілої, доступної і ефективної системи організації самостійної роботи та її оцінки.

При цьому необхідно пам'ятати, що самостійна робота студентів виконує в навчальному процесі кілька функцій:

− розвиваючу (підвищення культури розумової праці, привчання до творчих видів діяльності, вдосконалення інтелектуальних здібностей студентів);

− інформаційно-навчальну (навчальна діяльність на аудиторних заняттях, непідкріплена самостійною роботою, стає мало результативною);

− орієнтуючу і стимулюючу (процесу навчання надається прискорення і мотивація);

− виховну (формуються і розвиваються професійні якості фахівця);

− дослідницьку (новий рівень професійно-творчого мислення).

В основі самостійної роботи студентів лежать наступні принципи: розвиток творчої діяльності, цільове планування, особистісно-діяльнісний підхід.

Самостійну роботу можна назвати ефективною тільки в тому випадку, якщо вона організована і реалізується в освітньому процесі як цілісна система на всіх етапах навчання.

Можна виділити кілька об'єктивних закономірностей організації самостійної роботи студентів:

− творча складова самостійної роботи зростає в міру навчання;

− в процесі організації самостійної роботи виникає потреба в методичному забезпеченні;

− застосування інформаційних технологій стає частиною організації і моніторингу самостійної роботи студентів на всіх її етапах.

У процесі самостійної роботи студент набуває навиків самоорганізації, самоконтролю, самоврядування, саморефлексії і стає активним самостійним суб'єктом навчальної діяльності.

Самостійна робота повинна давати важливий вплив на формування особистості майбутнього фахівця. Кожен, хто навчається самостійно планує режим своєї роботи з урахуванням часу роботи бібліотеки, профільних лабораторій, комп'ютерних класів і т.п. Він виконує самостійну роботу за особистим індивідуальним планом, в залежності від його підготовки, часу та інших умов.

Першим завданням в організації позааудиторної самостійної роботи є складання розкладу, що відображає час занять і їх характер, перерви на обід, вечеря, відпочинок, сон, проїзд і т.п. Із самого початку студенту не потрібно прагнути робити відразу найважчу її частину. Доцільно вибрати щось середнє за складністю. Після цього, перейти до більш важкої роботи, легке залишивши наостанок. Розумову працю необхідно не тільки правильно організувати, а й стимулювати. Важливо вміти підтримувати стійку увагу до досліджуваного матеріалу.

Вироблення уваги вимагає значних вольових зусиль від студента. Стійка увага з'являється тоді, коли людина ставиться до справи з інтересом.

Слід правильно організувати свої заняття за часом: 50 хвилин − робота, 5-10 хвилин − перерва, після 3 годин роботи перерва − 20-25 хвилин. Інакше наростаюча втома спричинить нестійкість уваги. Організація активного відпочинку передбачає чергування розумової та фізичної діяльності, що відновлює працездатність людини.

1.2 Опис послідовності дій студента при виконанні самостійної роботи

Організацію самостійної роботи можна умовно розділити на три етапи:

- планування навчальної діяльності та її методична підготовка;
- здійснення цієї діяльності та її супровід;
- контроль, аналіз результатів (з можливими змінами в плануванні самостійної роботи).

Рекомендації щодо використання матеріалів навчально-методичного комплексу навчальної дисципліни

Зміст вивчення дисципліни "Комп`ютерні мережі" визначено її робочою програмою.

Інформативну частину навчання складають навчальні посібники, конспекти лекцій у паперовій та електронній формі, план, зміст та методичні рекомендації до проведення лабораторних занять, методичні рекомендації до виконання самостійної роботи, перелік рекомендованої до вивчення літератури, ресурси мережі Інтернет.

У рекомендаціях до проведення лабораторних занять з дисципліни "МОДЕЛІ ТА МЕТОДИ ПРЕДСТАВЛЕННЯ ЗНАНЬ І ШТУЧНОГО ІНТЕЛЕКТУ" міститься план занять, індивідуальні завдання та перелік питань для самостійного опрацювання матеріалу. Також зазначається короткий теоретичний коментар до кожної теми, що допомагає студентові ознайомитися із сутністю питань, на основі яких базується виконання завдань. Окрім цього у даних методичних рекомендаціях можна ознайомитися з питаннями, що виносяться на обговорення та списком літератури, необхідної для цілеспрямованої роботи студента при підготовці до наступного лабораторного заняття.

1.3 Рекомендації щодо роботи з літературою

Найважливішим інформаційним джерелом вивчення навчальної дисципліни «Моделі та методи представлення знань і штучного інтелекту» є ресурси мережі Інтернет. Основна частина матеріалу в Інтернеті розрахована на професіоналів, тому при вивченні навчальної дисципліни спочатку необхідно користуватися літературою навчального характеру.

При опрацюванні матеріалу потрібно дотримуватись таких правил:

1. Зосередитися на тому, що читаєш.
2. Виділити головну думку автора.
3. Виділити основні питання тексту від другорядних.
4. Зрозуміти думку автора чітко і ясно, що допоможе виробити власну думку.
5. Уявити ясно те, що читаєш.

У процесі роботи над темою тлумачення незнайомих слів і спеціальних термінів слід знаходити у фаховій літературі, термінологічних словниках. Незрозумілі місця, фрази, вирази доречно перечитувати декілька разів, щоб зрозуміти їх зміст.

Після прочитання тексту необхідно:

1. Усвідомити зв’язок між теоретичними положеннями і практикою.
2. Закріпити прочитане у свідомості.
3. Пов’язати нові знання з попередніми у даній галузі.
4. Перейти до заключного етапу засвоєння і опрацювання – записам.

Записи необхідно починати з назви теми та посібника, прізвища автора, року видання та назви видавництва. Якщо це журнал, то рік і номер видання, заголовок статті. Після чого скласти план, тобто короткий перелік основних питань тексту в логічній послідовності теми.

Складання плану, або тез логічно закінченого за змістом уривка тексту, сприяє кращому його розумінню. План може бути простий або розгорнутий, тобто більш поглиблений, особливо при опрацюванні додаткової літератури за даною темою. Записи необхідно вести розбірливо і чітко. Вони можуть бути короткі або розгорнуті залежно від рівня знань студента, багатства його літературної і професійної лексики, навичок самостійної роботи з книгою.

Для зручності користування записами необхідно залишати поля для заміток і вільні рядки для доповнень. Записи не повинні бути

одноманітними. В них необхідно виділяти важливі місця, головні слова, які акцентуються різним шрифтом або різним кольором шрифтів, підкреслюванням, замітками на полях, рамками, стовпчиками тощо. Записи можуть бути у вигляді конспекту, простих або розгорнутих тез, цитат, виписок, систематизованих таблиць, графіків, діаграм, схем.

1.4 Поради із підготовки до поточного, проміжного та підсумкового контролю

Контрольні заходи включають поточний і підсумковий контроль знань студентів. Поточний контроль є органічною частиною навчального процесу і проводиться під час лекцій та лабораторних занять.

Форми поточного контролю:

- усна співбесіда за матеріалами розглянутої теми на початку лабораторного заняття з оцінкою відповідей студентів (5-10 хв.);
- письмове фронтальне опитування студентів на початку чи в кінці лабораторного заняття (5-10 хв.). Відповіді перевіряються і оцінюються викладачем у поза аудиторний час;
- перевірка виконання завдань лабораторних робіт;
- тестова перевірка знань студентів;
- модульний контроль;
- інші форми.

При кредитно-модульній системі навчання теми самостійної роботи входять у модуль, який контролюються після закінчення логічно завершеної частини лекцій та інших видів занять з дисципліни та їх результати враховуються при виставленні підсумкової оцінки.

2. ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ ЩОДО ВИКОНАННЯ ІНДИВІДУАЛЬНОГО РОЗРАХУНКОВОГО ЗАВДАННЯ

2.1 Методи та функції PyTorch

(Документація: https://pytorch.org/docs/stable/index.html)

2.1.1 Бібліотеки:

np - бібліотека NumPy для роботи з багатовимірними масивами даних

pickle - бібліотека Pickle для серіалізації та десеріалізації структур даних ЯП Python

sklearn - бібліотека, що реалізує в основному методи класичного машинного навчання та інструменти для роботи з ними

PIL - легковага бібліотека Pillow для роботи із зображеннями та виведення графічних елементів безпосередньо в Jupyter Notebook

matplotlib - бібліотека для побудови графіків, здебільшого повторює API Matlab'a

torch - бібліотека Pytorch для глибокого навчання нейронних мереж

2.1.2 Ужиті скорочення:

torch.nn - nn
torch.nn.functional - F
torch.optim - optim

Методи:

torch.Tensor - створює тензор із багатовимірного масиву Numpy і успадковує його тип даних. За замовчуванням пам'ять під тензори виділяється на CPU. При виставленні прапора *requires_grad* автоматично відстежує градієнти за допомогою рушія *autograd*, який будує динамічний обчислювальний граф. Увімкнути відстеження тензора t можна так само за допомогою методу **t.requires_grad_(True).** У такому разі після виклику методу *backward*, в поле *grad* будуть записані похідні. Похідні тензора t можна очистити викликом методу t**.grad.zero_().** Для того щоб відсікти непотрібні обчислення похідних,

використовується метод detach, який створює копію тензора, при цьому прапор requires_grad знімається і відстеження рушієм *autograd* припиняється.

torch.numpy - створює багатовимірний NumPy масив даних із тензоа

torch.item - повертає число, але тільки якщо ранг тензора 0. В іншому випадку видає помилку і слід використовувати **torch.numpy**

torch.uint8, torch.int16, torch.int64, torch.float32 - приведення масиву до нового типу, аналогічно NumPy. Для приведення використовується метод .to (наприклад t.to(torch.int64)). За замовчуванням всі обчислення на графі проводяться в float64, є також можливість використання mixed precision (щось у float16, щось у float64), але це вважається просунутою технікою.

torch.ones, torch.zeros, torch.transpose, torch.reshape - API схожий, як у NumPy

torch.rand - створення випадкового тензора з числами в діапазоні від 0 до 1. Розмірність перераховується через кому

torch.t - транспонування тензора, схоже на розглянутий раніше **numpy.transpose**. Якщо дано тензор X, то можна його транспонувати за допомогою X.t()

torch.sum - підсумовування елементів тензора вздовж зазначеної осі axis. Якщо підсумовування проводиться вздовж останньої осі, то дозволяється вказати замість номера -1. Для збереження вихідної розмірності тензора, необхідно виставити прапор keepdims.

torch.maximum - здійснює поелементне порівняння тензорів і повертає максимальний з елементів. На практиці використовується для реалізації деяких функцій активації нейронної мережі

torch.mm - добуток тензорів. Для 2 двовимірних матриць із розмірностями (M, N) і (N, K) результатом цього методу буде двовимірна матриця розмірністю (M, K)

torch.exp - повторює функціонал numpy.exp - поелементне піднесення тензора до степеня експоненти

torch.log - поелементна операція логарифмування тензора - взяття натурального логарифма, зворотна операція потенціювання

torch.flatten - аналогічно NumPy .reshape(-1), якщо зазначено параметр start_dim, то починає "випрямлення" масиву, починаючи із зазначеного номера. Тобто для того, щоб перевести тензор t з формою (100, 32, 32, 32, 3) у форму (100, 3072), достатньо написати torch.flatten(t, start_dim=1)

F.one_hot - один із багатьох способів отримати гаряче кодування класу у вигляді PyTorch тензора. Наприклад, для 5 класів, гаряче кодування класу "4" буде [0, 0, 0, 0, 1, 0]

torch.utils.data.TensorDataset - створення пов'язаних тензорів, наприклад навчальних прикладів і відповідних міток. Як аргумент передаються тензори. Прийнятний спосіб створення набору даних, коли навчальна вибірка невелика і повністю поміщається в оперативній пам'яті.

torch.utils.data.DataLoader - В основі утиліти завантаження даних PyTorch лежить клас DataLoader. Він являє собою Python об'єкт, що повторюється за набором даних, з підтримкою набору даних у стилі map та ітератора; налаштування порядку завантаження даних; автоматичного розбиття на мінібатчі; завантаження даних в один і кілька процесів/потоків. Найкорисніші аргументи в конструкторі - розмір міні-батча batch_size і кількість паралельних процесів num_workers. Щоб перемішати дані (для кращої збіжності), слід виставити прапор shuffle в True

torch.save - збереження параметрів моделі на постійний носій інформації. Для цього першим аргументом передається model.state_dict(), де model - навчена нейромережева модель, а другим аргументом передається шлях з ім'ям файлу.

2.1.3 Створення моделей:

Створення моделей здійснюється за допомогою модуля nn, при цьому в модулі вже реалізовані найпопулярніші блоки нейронних мереж або шари, такі як:

- повнозв'язний шар Linear
- згортковий шар Conv2d
- шар пулінгу MaxPool2d
- нормалізація BatchNorm2d
- безліч активаційних функцій ReLU, Softmax, Tanh
- шари-регуляризатори, наприклад Dropout

У цій роботі ми розглянемо лише 2 блоки нейронної мережі з вище наведеного списку, а саме **Linear** і **ReLU**.

Задати модель можна 2 способами:

- за допомогою **nn.Sequential**
- за допомогою успадкування від класу **nn.Module**

Перший спосіб підходить для створення простих моделей без відгалужень. По суті їх можна уявити як конвеєр, де вхідний тензор передається низці послідовно наведених трансформацій для отримання вихідного тензора.

Якщо необхідно застосовувати складніші архітектури, де конвеєрні доріжки можуть розгалужуватися на кілька частин, то використовується **nn.Module**. Цей підхід дає змогу реалізувати найрізноманітніші архітектури.

Для створення простого багатошарового перцептрона з одним прихованим шаром і функцією нелінійності, відповідно до першого способу, достатньо написати такий код:

```
    model = nn.Sequential(
      nn.Linear(input_dims, hidden_dims),
      nn.ReLU(),
      nn.Linear(hidden_dims, num_classes)
    )
```

Для створення простого багатошарового перцептрона з одним прихованим шаром і функцією нелінійності, згідно з другим способом, необхідно створити клас і модель як екземпляр цього класу:

```
    class MLP(nn.Module):
        def   __init__(self,   input_dims,   hidden_dims,
num_classes,
                       *args, **kwargs):
            super(MLP, self).__init__()
            self.fc1 = Linear(input_dims, hidden_dims)
            self.fc2 = Linear(hidden_dims, num_classes)

        def forward(self, input):
             x = self.fc1(input)
             x = F.relu(x)
             x = self.fc2(x)
             return x

    model = MLP(input_dims, hidden_dims, num_classes)
```

При цьому допускається вкладати **nn.Module** і **nn.Sequential** усередині інших модулів, що дає змогу створювати дуже складні архітектури моделей.

2.1.4 Навчання моделей:

Перед навчанням моделей необхідно вибрати функцію втрат і оптимізатор. Різні функції втрат подано також у модулі nn:

- **nn.MSELoss** - середньоквадратична помилка (*y_true-y_pred)**2*
- **nn.BCEWithLogitsLoss** - бінарна перехресна ентропія для задач бінарної класифікації
- **nn.CrossEntropyLoss** - категоріальна перехресна ентропія для задач багатокласової класифікації

Як альтернативу можна власноруч реалізувати функцію втрат, наприклад для **MSELoss**:

```
inputs, y = batch
...
output = model(inputs)
loss = ((output - y)**2).sum()
...
```

Оптимізатори містяться в модулі **torch.optim**. Існує безліч оптимізаторів цільової функції, класичним є стохастичний градієнтний спуск *Stochastic Gradient Descent* або *SGD*. У конструктор класу необхідно передати ваги моделі, а також вказати крок навчання або *learning rate.*

Для переведення моделі в стан навчання необхідно викликати метод **train**. Після чого модель готова для навчання.

Для навчання нейромережевих моделей використовується градієнтний спуск і його різновиди, в основі яких лежить метод послідовних наближень.

За одну епоху умовно обирають проходження ітератора через увесь набір даних, за одну ітерацію - оптимізація параметрів моделі за допомогою поточного міні-батча. PyTorch автоматично рахує похідні під час виклику методу *backward*, застосованого до функції втрат.

При цьому при повторному виклику, значення нових градієнтів додадуться до попередніх розрахованих. Тому, для уникнення небажаних ефектів заведено очищати минулі значення градієнтів на кожній ітерації за допомогою методу *zero_grad*, застосованого до екземпляра класу оптимізатора.

Для оновлення параметрів нейронної мережі використовується метод *step*, застосований до екземпляра класу оптимізатора.

2.1.5 Перевірка якості моделей

Для переведення моделі в стан перевірки необхідно викликати метод *eval*. Після чого модель готова для перевірки.

Вихідний тензор передбачень моделі необхідно відсікти від обчислювального графа. Для цього використовується метод *detach*, застосований до вихідного тензора моделі. В іншому разі можливі витоки пам'яті. Метод *numpy* конвертує тензор у багатовимірний масив NumPy.

За замовчуванням модель виводить так звані логіти класів, а не їхні ймовірності. Для отримання ймовірностей необхідно застосувати функцію активації Softmax. Однак на практиці це необов'язково, оскільки величина логітів узгоджується з імовірністю класів, і для отримання номера найімовірнішого класу цей етап можна опустити. Номер класу отримують за допомогою або методу *argmax*, або методу *argsort*, причому останній дає змогу зчитати такі метрики, як Accuracy@5 і метрики ранжування.

2.2 Імпортування необхідних бібліотек

```
    import numpy as np
    import torch
    import torch.optim as optim
    import torch.nn as nn
    import torch.nn.functional as F
    from  torch.utils.data  import  TensorDataset,
DataLoader
    import pickle
    from sklearn.metrics import classification_report
    from  sklearn.datasets  import  make_circles,
make_moons
    from PIL import Image
```

2.3 Опис методів бібліотек

2.3.1 Методи та функції NumPy:

(Подробиці в документації https://numpy.org/doc/1.22/reference/index.html)

np.array - створення масиву зі списку або іншого масиву

np.shape - виводить розмірність багатовимірного масиву (тобто для масиву 2x2 буде виведено кортеж (2, 2))

np.size - виводить число елементів у масиві (тобто для масиву 2x2 буде виведено число 4)

np.uint8, np.int16, np.int64, np.float32 - приведення масиву до нового типу, при цьому в пам'яті виділяється місце під новий масив обраного типу. Число після типу позначить, скільки біт даних використовується для зберігання одного елемента масиву. Для зберігання картинок найчастіше використовується економний uint8 - беззнаковий 8-бітний цілочисельний тип даних (діапазон чисел 0-255)

np.ones, np.zeros - створення вже заповнених масивів або одиницями, або нулями. Як аргумент передається список або кортеж з необхідною розмірністю. Наприклад np.ones((10,)) створить вектор із 10 одиничок. А np.zeros((32, 32, 3)) створить двовимірний масив роздільною здатністю 32 на 32 пікселі з 3 каналами. На практиці використовується для перевірки архітектури моделі в прямому напрямку

np.arange - створення вже заповненого масиву у вигляді зростаючої арифметичної прогресії від першого аргументу до другого аргументу не включно з кроком, який задаєтесь третім аргументом. Перший і третій аргументи можна опускати, у такому разі виходить компактний запис np.arange(3) => [0, 1, 2]

np.repeat - дублювання елементів масиву на кількість, вказану першим аргументом. Таким чином, для масиву arr = [0, 1] arr.repeat(2) поверне [0, 0, 1, 1]

np.exp - застосування поелементної операції потенціювання до масиву

np.random.normal - генерація масиву, заповненого випадковими нормальними величинами зі стандартним відхиленням, заданим через аргумент scale, і із середнім значенням, що дорівнює аргументу mean. Число елементів у масиві задається числом або списком, переданим аргументу size.

np.random.randint - генерація масиву, заповненого випадковими цілими числами в діапазоні, що задається аналогічно np.arange. Число елементів у масиві задається числом або списком, переданим аргументу size.

np.reshape - буквально зміна розмірності багатовимірного масиву з урахуванням числа елементів. Як аргумент передається багатовимірний масив, а також список або кортеж з новою розмірністю. Наприклад np.reshape([0, 1, 2, 3], (2,2)) створить двовимірний масив розміром 2x2. При цьому в пам'яті новий масив не виділяється, а змінюється лише спосіб обходу по ньому. Дозволяється також і такий спосіб виклику методу: arr.reshape(2, 2). Зверніть увагу на відсутність додаткових дужок. Якщо замість конкретного числа підставити -1, то розмірність буде підраховано автоматично. На практиці використовується для випрямлення картинок у вигляді одновимірного масиву: X.reshape(-1, 3072)# [100, 32, 32, 3] -> [100, 3072]

np.transpose - перейменування осей багатовимірного масиву. Для роботи із зображеннями прийнято два формати NHWC і NCHW (N - число картинок у масиві, C - число каналів, H - висота, W - ширина). Як аргумент передається багатовимірний масив, а також список або кортеж з новою розстановкою осей. Наприклад np.transpose([[0, 1, 2, 3]], (1,0)) створить двовимірний вектор-стовпець [[[0], [1], [2], [3]]. Зауважте, що відлік осей починається з 0. На практиці використовується для переведення NHWC в NCHW і назад. У першому випадку 0 вісь N залишається на своєму першому місці, перша і друга осі H і W зсуваються на одну позицію вправо, а 3 вісь - C ставиться на друге місце. Тобто отримаємо таку перестановку: [0, 3, 1, 2]

np.isin - аналог SQL оператора IN, поелементна перевірка входження масиву в колекцію. np.isin([0, 2, 1], [2, 3]) поверне [False, True, False]

індексування - вибір підмасиву або зрізу масиву здійснюється за допомогою квадратних дужок []. Якщо arr = np.array([2, 1, 0]), то arr[0] поверне перший елемент. arr[[0, 1]] - звернення за індексом, arr[[True, False, True]] - звернення за булевою маскою. Зауважте, що звернення за індексом необов'язково має збігатися з розмірністю масиву, на відміну від звернення за маскою. На практиці зручно записувати значення маски в окрему змінну. Для вибору конкретного стовпця в багатовимірному масиві використовується синтаксис зрізів [:, k], де k - номер стовпця. Якщо k дорівнює -1, то використовується останній стовпець або елемент. Так, наприклад, для масиву arr = np.array([[0, 1],

[2, 3], [4, 5]) вираз arr[:, 0] поверне масив [0, 2, 4]. Оскільки використовується індекс зрізів (стандартний синтаксис Python), то можна також виконувати зрізи багатовимірних масивів. Для попереднього прикладу arr[1:2, 0:1] поверне [[2]]

np.unique - аналог SELECT DISTINCT у SQL. За стандартних параметрів повертає одновимірний підмасив, що містить унікальні елементи. Якщо вказати виставити прапор return_inverse, то повернеться масив із номерами відліків масиву з унікальними елементами. По суті виконується Label Encoding

np.concatenate - конкатенація багатовимірного масиву вздовж вказаної осі. Номер осі вказується через аргумент axis. Наприклад може бути використаний для об'єднання декількох ознак або декількох наборів даних. У контексті зображень може використовуватися для об'єднання або склеювання кількох зображень в одне як вертикально, так і горизонтально. У контексті звуку - склеювання двох аудіодоріжок.

np.max, np.min - повертає максимальний і мінімальний елементи масиву вздовж вказаної осі, відповідно. Якщо номер осі не вказано, то повертається число. Номер осі вказується через аргумент axis. Якщо вказується -1, то вважається, що використовується останній номер осі. Дозволяється також і виклик функції як методу багатовимірного масиву: arr.max()

np.argmax - повертає індекс максимального елемента масиву вздовж вказаної осі. Якщо номер осі не вказано, то повертається перший індекс, що відповідає максимальному значенню в масиві, тобто одне число. Номер осі вказується через аргумент axis. Якщо вказується -1, то вважається, що використовується останній номер осі. На практиці використовується для розрахунку метрики частки правильних відповідей моделі (Accuracy). Дозволяється також і виклик функції як методу багатовимірного масиву: arr.argmax(axis=-1).

2.3.2 Методи та функції Pickle

(Документація: https://docs.python.org/3/library/pickle.html)

pickle.dump - серіалізація структури даних Python. Першим аргументом йде сама структура, а другим FileObject. При цьому FileObject має бути відкритий у режимі запису байт (wb). Можна вказати кодування байт (big endian/ little endian). Тим самим можна зберігати на постійному носії стандартні структури даних, у тому числі NumPy масиви.

pickle.load - десеріалізація структури даних Python. Першим аргументом йде FileObject. При цьому FileObject має бути відкритий у режимі читання байт (rb). Можна вказати кодування байт (big endian/ little endian). Тим самим можна завантажувати раніше збережені структури даних, що може бути корисно, якщо для їх створення потрібен тривалий час (наприклад, параметри моделі глибокого навчання).

2.3.3 Методи та функції Sklearn

(Документація: https://scikit-learn.org/stable/modules/classes.html)

datasets.make_circles, datasets.make_moons - генерація синтетичної навчальної вибірки для задачі класифікації, повертає X - двовимірний масив із числом прикладів і числом ознак (ознак 2), а також одновимірний масив із мітками класів (0 або 1)

metrics.classification_report - створює текстовий звіт, що показує основні метрики класифікації (частка правильних відповідей, повнота, точність, f1-мера). Як перший аргумент передаються справжні мітки класу, як другий - мітки класу, передбачені моделлю. Додаткові корисні аргументи: digits - число виведених знаків після коми (за замовчуванням 2), output_dict - повертає словник із розрахованими метриками замість рядка, sample_weight - розраховує зважені метрики на основі ваги кожного прикладу

metrics.confusion_matrix - обчислює матрицю помилок моделі для оцінки точності класифікації. Матриця помилок ідеальної моделі має значення тільки на головній діагоналі. Може бути використана для підрахунку всіх класичних метрик класифікації (частка правильних відповідей, повнота, точність, специфічність, f1-мера).

2.3.4 Методи та функції PIL

(Документація: https://pillow.readthedocs.io/en/stable/)

Image.fromarray - створює об'єкт Image на основі двовимірного масиву або двовимірного масиву з каналами. Часто лається, якщо тип даних не uint8. Часто лається, якщо робиться спроба створити чорно-біле зображення з картинки розмірністю (W, H, 1). Для того, щоб отримати назад масив з об'єкта Image, досить привести його до NumPy масиву, наприклад np.array(img)

Image.resize - змінює роздільну здатність зображення за допомогою інтерполяції. Першим аргументом вказується список з новою

шириною і висотою зображення. За бажання можна вказати тип інтерполяції через аргумент resample. Підтримувані значення: PIL.Image.NEAREST, PIL.Image.BOX, PIL.Image.BILINEAR, PIL.Image.HAMMING, PIL.Image.BICUBIC, PIL.Image.LANCZOS. За замовчуванням використовується бікубічна інтерполяція.

Image.convert - переводить зображення з однієї колірної схеми в іншу. Нова колірна схема передається рядком, L - чорно-біла, LA - чорно-біла з прозорістю, RGB - стандартна колірна схема з 3 каналами, RGBA - стандартна колірна схема з 3 каналами кольору і одним каналом прозорості, HSV - альтернативне колірне представлення і т.д.

Image.open - зчитує зображення за вказаним шляхом у вигляді рядка або FileObject. При створенні набору даних може неправильно визначити формат (наприклад L замість RGB), тому рекомендується відразу після open приводити до потрібного формату за допомогою методу convert

Image.save - зберігає зображення за вказаним шляхом у вигляді рядка або FileObject. Якщо вказується FileObject, то потрібно також вказати формат зображення в аргументі format, наприклад 'PNG' або 'JPEG'

2.3.5 Методи та функції Matplotlib

(Документація: https://matplotlib.org/stable/api/index.html)

Прийняті скорочення: matplotlib.pyplot - plt

Методи:

plt.plot - малює графік по точках і з'єднує їх лінією. Першим аргументом передаються x-координати, другим - y-координати. Якщо не передавати другий аргумент, x-координати будуть прийняті за y, а як x будуть використані відліки масиву. Додаткові корисні аргументи: linestyle - тип відображуваної лінії ('--', '-', '-.' тощо), color - колір лінії ('k' - чорний, 'r' - червоний, 'white' - білий тощо), alpha - прозорість лінії, число від 0 (лінію не видно) до 1 (немає прозорості), label - текстова мітка цього графіка.

plt.scatter - малює графік по точках без з'єднання лініями. Першим аргументом передаються x-координати, другим - y-координати. Якщо не передавати другий аргумент, x-координати будуть прийняті за y, а як x будуть використані відліки масиву. Додаткові корисні аргументи: s - розмір точок, color - колір точок ('k' - чорний, 'r' - червоний, 'white' - білий

тощо), alpha - прозорість точок, число від 0 (лінію не видно) до 1 (немає прозорості), label - текстова мітка цього графіка.

plt.contourf - малює заповнені контурні лінії, що розмежовують межі.

plt.show - примусове відтворення графіка, може використовуватися для виведення декількох графіків в одному блоці коду.

plt.legend - відображає раніше зазначені мітки графіків

plt.xlim - обмежує діапазон x-координат від першого до другого аргументу. За замовчуванням діапазон горизонтальної осі підбирається автоматично на основі використовуваних даних. Для завдання діапазону значень горизонтальної осі вручну і використовується цей метод

plt.ylim - аналогічно plt.xlim, але для вертикальної осі.

3. ЗМІСТ ІНДИВІДУАЛЬНОЇ РОЗРАХУНКОВОЇ РОБОТИ СТУДЕНТА І МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ЇЇ ВИКОНАННЯ

3.1 Завдання для самостійної роботи

Необхідно познайомитися з фреймворком машинного навчання PyTorch і виконати три завдання:

1. Побудувати регресійну модель за теоремою універсальної апроксимації, використовуючи ручне диференціювання
2. Побудувати та досліди модель багатошарового перцептрона для бінарної класифікації за допомогою PyTorch
3. Навчити повнозв'язну нейронну мережу класифікації 3 класів зображень із набору даних CIFAR100 за варіантом із прикладу і потім підвищити точність на тестовій вибірці.

Для завдання 3 потрібно сформувати свою підвибірку CIFAR100 за варіантом. Завдання виконуються на платформі Google Colab.

Звіт має містити: титульний аркуш, завдання з варіантом, скріншоти та короткі пояснення щодо кожного етапу роботи, підсумкову таблицю з результатами для всіх варіантів навчання.

3.2 Індивідуальне розрахункове завдання

1. Проаналізуйте результати навчання вашої моделі. Що говорить про неї точність на навчальній і тестовій вибірці? З якими класами модель справляється краще і чому?
2. Проаналізуйте результати навчання. Чи виникає перенавчання вашої моделі? Що необхідно зробити, щоб нівелювати це (не використовуючи регуляризацію)?
3. Змінити розмір батча, але зберігаючи загальну кількість ітерацій. Проаналізувати результати навчання з новими гіперпараметрами. Що змінилося і чому?
4. Зменшити швидкість навчання і збільшити загальну кількість ітерацій, щоб підвищити точність моделі.
5. Змінити модель: кількість нейронів і шарів. Проаналізувати результати навчання нової моделі. Знайти найкращі гіперпараметри для цієї моделі.
6. Проаналізувати які дії допомогли підвищити точність вашої моделі та пояснити чому.

3.3 Коментарі до самостійного завдання

Зміну швидкості навчання і розміру батча робимо тільки після знаходження моменту перенавчання - епохи, де помилка на тестовій вибірці найменша. Інакше поліпшенням кроків ви можете посилити перенавчання (поліпшуватися під час навчання, погіршуватися для тесту).

Збільшення розміру батча за тієї самої кількості ітерацій має хоч трохи (у середньому) поліпшити навчання.

У скільки разів змінюєте швидкість навчання або розмір батча - у стільки ж змінюєте кількість епох для збереження ітерацій. На практиці ця умова не потрібна, бо під час нового навчання змінюють одразу багато гіперпараметрів. Але ми вчимося, тому досліджуємо поведінку кожного гіперпараметра окремо: ми не можемо порівняти 5 великих кроків і 7 маленьких, можемо порівняти 5 великих і 5 маленьких.

При зміні моделі потрібно шукати гіперпараметри заново - це нова модель. Не потрібно розписувати знову всі кроки, просто зробіть це самі - знайдіть перенавчання та пошукайте швидкість і розмір батча, які будуть кращими.

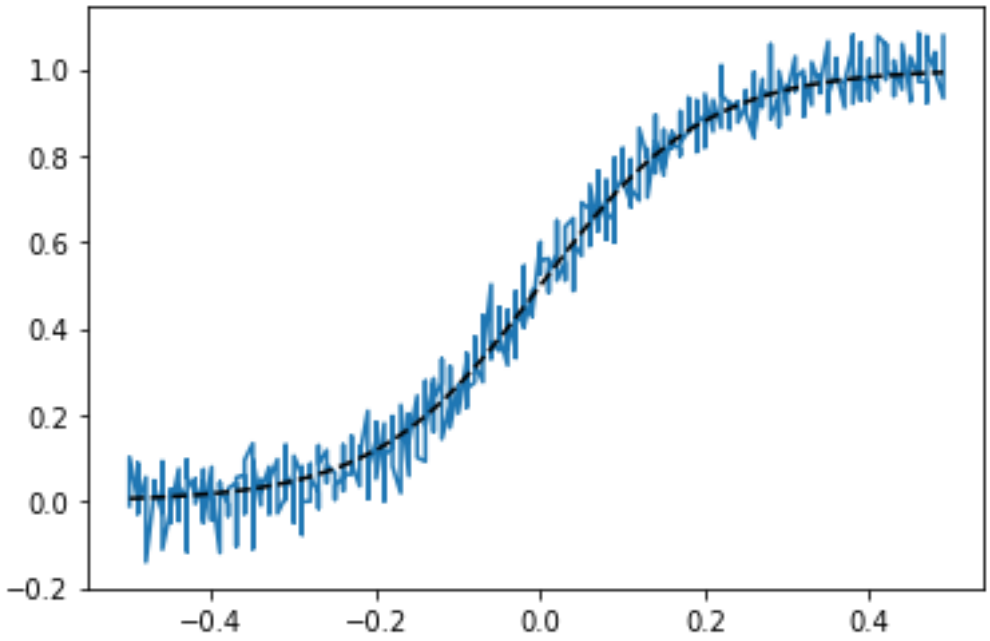
3.4 Методичні рекомендації щодо виконання роботи

3.4.1 Завдання 1. Задача регресії за теоремою універсальної апроксимації, ручне диференціювання

Генерація вибірки та ініціалізація параметрів нейронної мережі

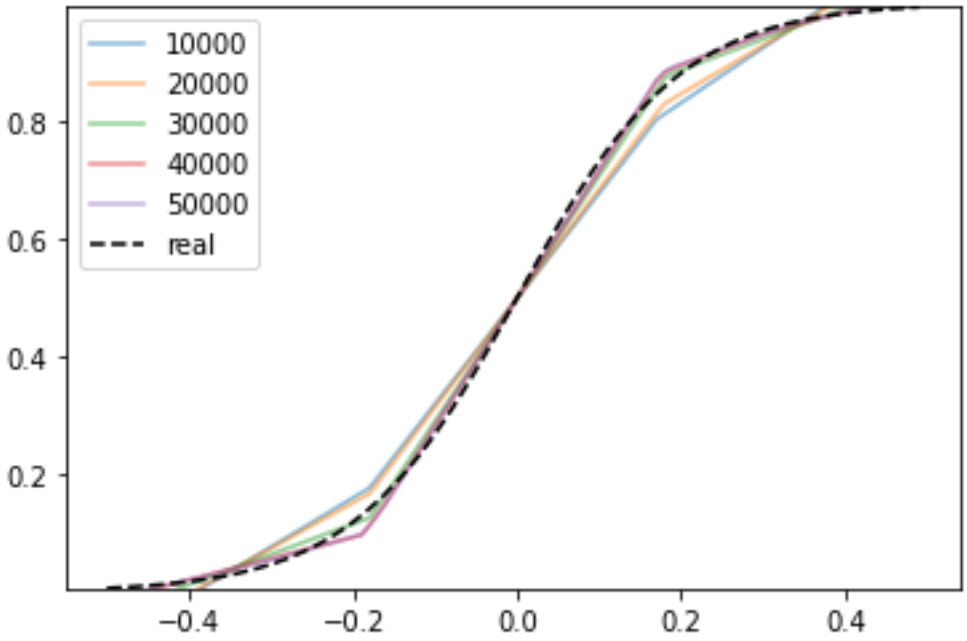
```
X = (np.arange(100)/100 - 0.5).repeat(5)

y = 1/(1+np.exp(-10*X))
yn = np.random.normal(scale=0.05, size=y.size)+y

plt.plot(X, yn)
plt.plot(X, y, linestyle='--', c='k')
##############################################
tensor_X = torch.Tensor(X.reshape(-1, 1))
tensor_y = torch.Tensor(yn.reshape(-1, 1))
```

```
    HIDDEN_SIZE = 64
    # Ініціалізація ваг MLP з одним прихованим шаром
    weights_1 = (torch.rand(1, HIDDEN_SIZE)-.5)/10
    bias_1 = torch.zeros(HIDDEN_SIZE)

    weights_2 = (torch.rand(HIDDEN_SIZE, 1)-.5)/10
    bias_2 = torch.zeros(1)
```



Навчання нейронної мережі для задачі регресії

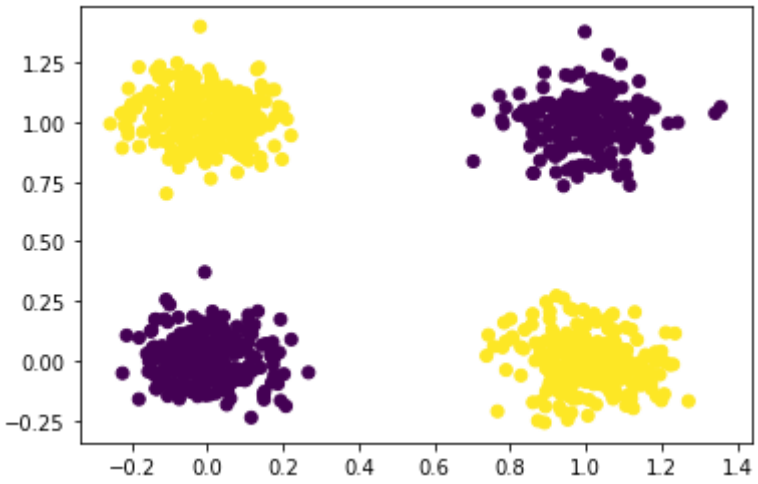
```
    # Визначаємо функцію нелінійності
    relu = lambda x: torch.maximum(x, torch.Tensor([0]))
    # Прямий прохід
    forward = lambda x:
(weights_2.t()*relu((weights_1*x) + bias_1)
                          ).sum(axis=-1,keepdims=True)
+ bias_2
    loss = lambda y, y_: ((y-y_)**2).sum(axis=-1)

    # # зворотній прохід
    def backward(X, y, y_pred):
        # похідна функції втрат за y_pred
```



3.4.2 Завдання 2. Бінарна класифікація за допомогою автодиференціювання PyTorch

Генерація вибірки та ініціалізація параметрів нейронної мережі

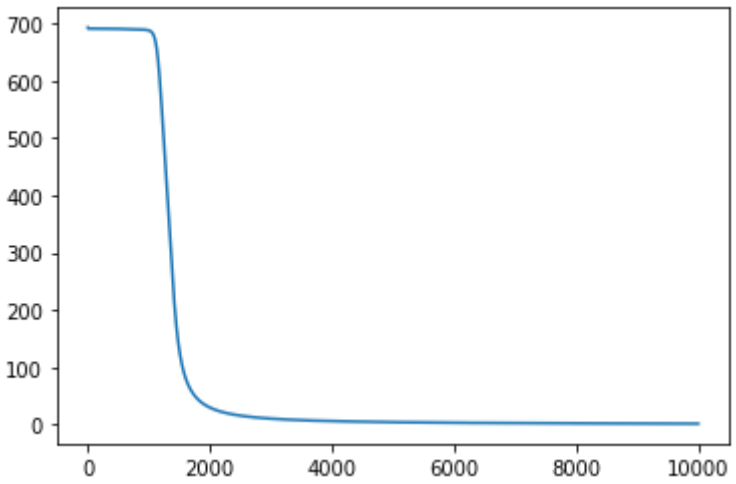
```
    X = np.random.randint(2, size=(1000, 2))

    y = (X[:, 0] + X[:, 1]) % 2 # XOR
    X = X + np.random.normal(0, scale=0.1, size=X.shape)
    #X, y = make_circles(n_samples=1000, noise=0.025)
    #X, y = make_moons(n_samples=1000, noise=0.025)
    plt.scatter(X[:, 0], X[:, 1], c=y)
    ##################################################
#
    tensor_X = torch.Tensor(X.reshape(-1, 2))
    tensor_y = torch.Tensor(y.reshape(-1, 1))

    HIDDEN_SIZE = 16
    # Ініціалізація ваг MLP з одним прихованим шаром
    weights_1     =     ((torch.rand(2,     HIDDEN_SIZE)-
.5)/10).detach().requires_grad_(True)
    bias_1          =          torch.zeros(HIDDEN_SIZE,
requires_grad=True)

    weights_2     =     ((torch.rand(HIDDEN_SIZE,     1)-
.5)/10).detach().requires_grad_(True)
    bias_2 = torch.zeros(1, requires_grad=True)
```



Навчання нейронної мережі задачі класифікації

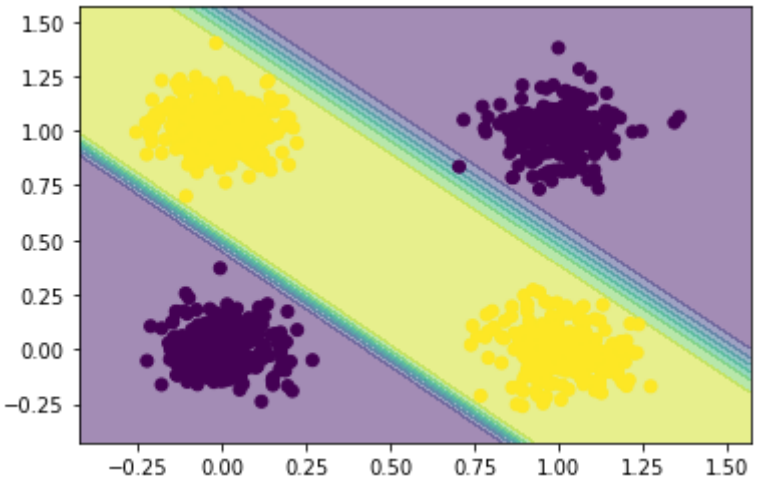
```
    # Визначаємо функцію нелінійності
    def sigmoid(x):
        return 1/(1+torch.exp(-x))

    # Прямий прохід
    def forward(x):
        hidden = torch.mm(x, weights_1) + bias_1
        hidden_nonlin = sigmoid(hidden)
        output                                          =
(weights_2.t()*hidden_nonlin).sum(axis=-
1,keepdims=True) + bias_2
        return sigmoid(output)

    # Logloss
    def loss(y_true, y_pred):
        return          -1*(y_true*torch.log(y_pred)+(1-
y_true)*torch.log(1-y_pred)).sum()

    # задаємо шаг навчання
    lr = 1e-3
    # задаємо число ітерацій
    iters = 10000
    params = [weights_1, bias_1, weights_2, bias_2]
    losses = []
    for i in range(iters):
        output = forward(tensor_X)
        lossval = loss(tensor_y, output)
        lossval.backward()  # тут включається в роботу
autograd
        for w in params:
```

```
            with torch.no_grad():
                w -= w.grad*lr # оновлюємо ваги
            w.grad.zero_() # зануляємо градієнти, щоб не
накопичувалися за ітерації
        losses.append(lossval.item())
    # виводимо історію функції втрат за ітераціями
    plt.plot(losses)
```



Перевірка результатів навчання

```
    X_diff = X.max() - X.min()
    X_left = X.min() - 0.1*X_diff
    X_right = X.max() + 0.1*X_diff
    grid = np.arange(X_left, X_right, 0.01)
    grid_width = grid.size
    surface = []
    # створюємо точки по сітці
    for x1 in grid:
        for x2 in grid:
            surface.append((x1, x2))
```



3.4.3 Завдання 3. Класифікація зображень CIFAR100

Варіанти для Завдання 3

Ви повинні використовувати наступні класи із набору даних CIFAR100:

- Номер групи + 15
- Номер варіанту + 56
- Номер варіанту + 21

Завантаження та розпакування набору даних CIFAR100

```
    !wget    https://www.cs.toronto.edu/~kriz/cifar-100-
python.tar.gz
    !tar -xvzf cifar-100-python.tar.gz
    --2022-02-10                                  21:29:10--
https://www.cs.toronto.edu/~kriz/cifar-100-
python.tar.gz
    Resolving                          www.cs.toronto.edu
(www.cs.toronto.edu)... 128.100.3.30
    Connecting           to            www.cs.toronto.edu
(www.cs.toronto.edu)|128.100.3.30|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 169001437 (161M) [application/x-gzip]
    Saving to: ‘cifar-100-python.tar.gz’

    cifar-100-python.ta       100%[===================>]
161.17M  46.1MB/s    in 3.9s

    2022-02-10  21:29:14  (41.2  MB/s)  -  ‘cifar-100-
python.tar.gz’ saved [169001437/169001437]
```

Читання тренувальної та тестової вибірки

```
    with open('cifar-100-python/train', 'rb') as f:
        data_train = pickle.load(f, encoding='latin1')
    with open('cifar-100-python/test', 'rb') as f:
        data_test = pickle.load(f, encoding='latin1')

    # Тут вказати класи згідно варіанту!!!
    CLASSES = [0, 55, 58]

    train_X = data_train['data'].reshape(-1, 3, 32, 32)
    train_X = np.transpose(train_X, [0, 2, 3, 1]) # NCHW
-> NHWC
    train_y = np.array(data_train['fine_labels'])
    mask = np.isin(train_y, CLASSES)
    train_X = train_X[mask].copy()
    train_y = train_y[mask].copy()
    train_y = np.unique(train_y, return_inverse=1)[1]
    del data_train

    test_X = data_test['data'].reshape(-1, 3, 32, 32)
    test_X = np.transpose(test_X, [0, 2, 3, 1])
    test_y = np.array(data_test['fine_labels'])
    mask = np.isin(test_y, CLASSES)
    test_X = test_X[mask].copy()
    test_y = test_y[mask].copy()
    test_y = np.unique(test_y, return_inverse=1)[1]
    del data_test
    Image.fromarray(train_X[50]).resize((256,256))
```

Створення Pytorch DataLoader'a

```
    batch_size = 128
    dataloader = {}
    for (X, y), part in zip([(train_X, train_y), (test_X,
test_y)],
                            ['train', 'test']):
        tensor_x = torch.Tensor(X)
        tensor_y                                        =
F.one_hot(torch.Tensor(y).to(torch.int64),

num_classes=len(CLASSES))/1.
        dataset = TensorDataset(tensor_x, tensor_y) #
створення об'єкту датасета
        dataloader[part]      =      DataLoader(dataset,
batch_size=batch_size,    shuffle=True)    #    ствpення
екземпляра классу DataLoader
    dataloader
    {'test': <torch.utils.data.dataloader.DataLoader at
0x7ff2823c59d0>,
     'train':   <torch.utils.data.dataloader.DataLoader
at 0x7ff1706411d0>}
```

Створення Pytorch моделі багатошарового перцептрона з одним прихованим шаром

```
    class Normalize(nn.Module):
        def __init__(self, mean, std):
            super(Normalize, self).__init__()
            self.mean = torch.tensor(mean)
            self.std = torch.tensor(std)

        def forward(self, input):
            x = input / 255.0
            x = x - self.mean
            x = x / self.std
            return torch.flatten(x, start_dim=1) # nhwc
-> nm

    class Cifar100_MLP(nn.Module):
        def __init__(self, hidden_size=32, classes=100):
            super(Cifar100_MLP, self).__init__()
            #             https://blog.jovian.ai/image-
classification-of-cifar100-dataset-using-pytorch-
8b7145242df1
```

```
            self.norm                                          =
Normalize([0.5074,0.4867,0.4411],[0.2011,0.1987,0.2025]
)
            self.seq = nn.Sequential(
                nn.Linear(32*32*3, hidden_size),
                nn.ReLU(),
                nn.Linear(hidden_size, classes),
            )

        def forward(self, input):
            x = self.norm(input)
            return self.seq(x)

    HIDDEN_SIZE = 10
    model    =    Cifar100_MLP(hidden_size=HIDDEN_SIZE,
classes=len(CLASSES))
    model
    Cifar100_MLP(
      (norm): Normalize()
      (seq): Sequential(
        (0):  Linear(in_features=3072,  out_features=10,
bias=True)
        (1): ReLU()
        (2):   Linear(in_features=10,   out_features=3,
bias=True)
      )
    )
```

Вибір функції втрат і оптимізатора градієнтного спуску

```
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.SGD(model.parameters(), lr=0.005)
    Навчання моделі за епохами
    [ ]
    EPOCHS = 250
    steps_per_epoch = len(dataloader['train'])
    steps_per_epoch_val = len(dataloader['test'])
    for epoch in range(EPOCHS):  # прохід по набору даних
декілька разів
        running_loss = 0.0
        model.train()
        for  i,  batch  in  enumerate(dataloader['train'],
0):
            #  отримання  одного  мінібатча;  batch  це
двоелементний список з [inputs, labels]
            inputs, labels = batch
```

Перевірка якості моделі за класами на навчальній і тестовій вибірках

```
    for part in ['train', 'test']:
        y_pred = []
        y_true = []
        with torch.no_grad(): # вимкнення автоматичного
диференціювання
            for i, data in enumerate(dataloader[part],
0):
                inputs, labels = data

                outputs                                 =
model(inputs).detach().numpy()
                y_pred.append(outputs)
                y_true.append(labels.numpy())
            y_true = np.concatenate(y_true)
            y_pred = np.concatenate(y_pred)
            print(part)

print(classification_report(y_true.argmax(axis=-1),
y_pred.argmax(axis=-1),
                                        digits=4,
target_names=list(map(str, CLASSES))))
            print('-'*50)
    train
                  precision    recall  f1-score   support

               0     1.0000    0.9900    0.9950       500
              55     0.9881    1.0000    0.9940       500
              58     0.9980    0.9960    0.9970       500

        accuracy                         0.9953      1500
       macro avg     0.9954    0.9953    0.9953      1500
    weighted avg     0.9954    0.9953    0.9953      1500
```

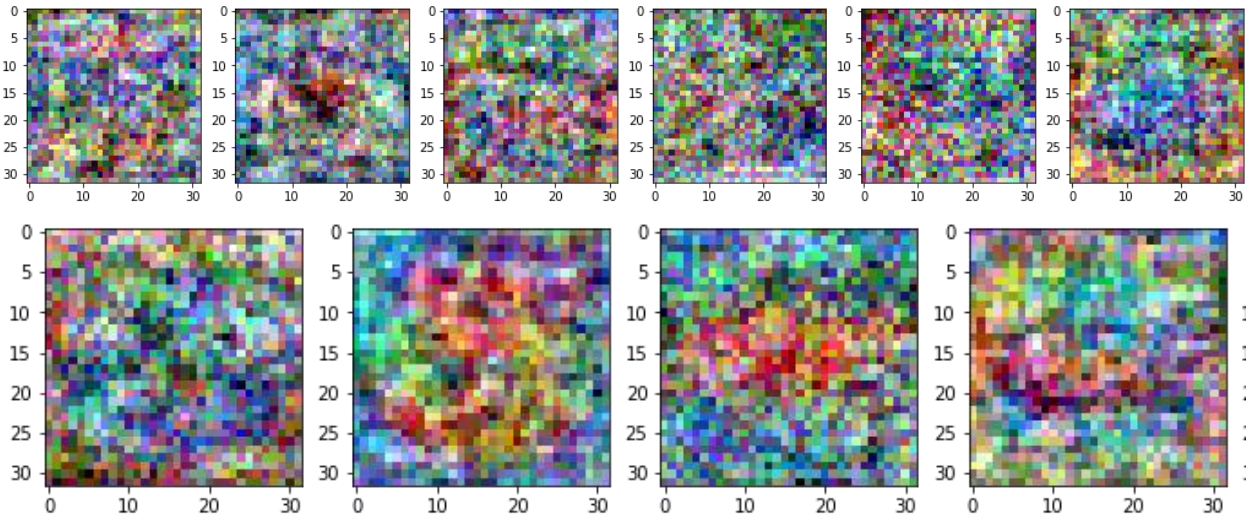
---

test

	precision	recall	f1-score	support
0	0.8174	0.9400	0.8744	100
55	0.7158	0.6800	0.6974	100
58	0.6889	0.6200	0.6526	100
accuracy			0.7467	300
macro avg	0.7407	0.7467	0.7415	300
weighted avg	0.7407	0.7467	0.7415	300

Візуалізація ваг

```
    weights =
list(model.parameters())[0].detach().numpy()
    print(weights.shape)
    fig, ax = plt.subplots(1, weights.shape[0],
figsize=(3*weights.shape[0], 3))
    for i, ω in enumerate(weights):
        ω = ω.reshape(32, 32, 3)
        ω -= np.percentile(ω, 1, axis=[0, 1])
        ω /= np.percentile(ω, 99, axis=[0, 1])
        ω = np.clip(ω, 0, 1)
        ax[i].imshow(ω)
```



4. КОНТРОЛЬНІ ПИТАННЯ

1. Повнозв'язна нейронна мережа, поясніть структуру, обчислення та призначення шарів і складових нейронів.
2. Укажіть кількість нейронів, зв'язків і ваг у повнозв'язній нейронній мережі 3 Опишіть завдання регресії та класифікації. Які функції втрат застосовуються в цих задачах?
3. опишіть структуру набору даних, призначення його частин
4. Опишіть алгоритм стохастичного градієнтного спуску. Укажіть призначення гіперпараметрів. У чому відмінність пакетного та стохастичного спуску?
5. Що таке епоха, ітерація, батч навчання. Як вони взаємопов'язані?
6. Що таке навчання з учителем, без учителя, з підкріпленням? Наведіть приклади методів і завдань для кожного виду навчання.

5. КРИТЕРІЇ ОЦІНЮВАННЯ

Критерії оцінювання знань студентів	Максимальна к-сть балів
Індивідуальне розрахункове завдання (ІНДРЗ)	30
робота виконана у повному обсязі з використанням усіх перелічених елементів	25-20
відсутність окремих з перелічених елементів	19-10
робота виконана без дотримання більшості з визначених вимог	9-8
робота не виконана або не зарахована	0-7

6. СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Базові

1 Методи та системи штучного інтелекту: навч. посіб. / укл. Д.В. Лубко, С.В. Шаров. – Мелітополь: ФОП Однорог Т.В., 2019.– 264 с.

2 Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2020.

3 Глибовець А.М., Глибовець М.М., Поляков М.В. Інтелектуальні мережі // НаУКМА, Дніпропетровськ, 2014.

4 Прикладні системи штучного інтелекту / Бібліотека підручників та статтей Posibniki (2022). URL: https://posibniki.com.ua/catalog-prikladni-sistemi-shtuchnogo-intelektu

5 Зайченко Ю.П. Основи проектування інтелектуальних систем. Навч. посібник. – Київ: Видавничий дім «Слово», 2004. – 352с.

6 Руденко О. Г., Бодянський Є. В. Штучні нейронні мережі: Навчальний посібник. — Харків: ТОВ "Компанія СМІТ", 2006. — 404 с.

7 Іванченко Г. Ф. Прикладні системи штучного інтелекту. Навч.посібник. - К.:КНЕУ, 2014.-630 с.

8 Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник; за заг. ред. С. О. Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375 с.

Додаткові

9 Бондаренко М. Ф. Мозгоподобные структуры : справочное пособие. / М. Ф. Бондаренко, Ю. П. Шабанов-Кушнаренко. – Т. 1. – Київ : Наукова думка, 2011. – 460 с.

10 Alex Smola and S.V.N. Vishwanathan. Introduction to Machine Learning. Cambridge University Press 2008. https://alex.smola.org/drafts/thebook.pdf

11 Shai Shalev-Shwartz, Shai Ben-David. UNDERSTANDING MACHINE LEARNING. From Theory to Algorithms. Cambridge University Press 2014. https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

12 Методичні вказівки для самостійної роботи з курсу «Теорія інтелекту» : для студ. спец. 121 «Інженерія програмного забезпечення»

/ О. Ю. Чередніченко, Д. Л. Орловський, А. М. Копп ; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків : НТУ «ХПІ», 2017. – 76 с.

13 Козуля Т.В. Формування знання-орієнтованого інформаційного забезпечення досліджень складних систем" / Т.В. Козуля, Н.В. Шаронова , М.М. Козуля // Системні дослідження та інформаційні технології – 2017, №3 – С.37-46.

14 Дорошенко А.Ю. Розробка програмних компонентів інформаційної системи екстракції фактографічних даних з веб-ресурсів / А.Ю.Дорошенко, Н.В. Шаронова, Б.О. Єна, О.В. Янголенко // Проблеми інформаційних технологій. – 2018. – №1 (023). – С.27-35.

ДОДАТОК А. ПРИКЛАД ОФОРМЛЕННЯ ЗВІТУ

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«МЕТІНВЕСТ ПОЛІТЕХНІКА»
Факультет автоматизації виробництва
та цифрових технологій
Кафедра цифрових технологій
та проєктно-аналітичних рішень

ЗВІТ З ІНДИВІДУАЛЬНОГО ЗАВДАННЯ

з дисципліни:
«МОДЕЛІ ТА МЕТОДИ ПРЕДСТАВЛЕННЯ
ЗНАНЬ І ШТУЧНОГО ІНТЕЛЕКТУ»

на тему
«Дослідження та налагодження
гіперпараметрів нейроних мереж»

Роботу виконав Іван ФЕЩЕНКО

Роботу перевірив Олександр ШМАТКО

ЗАПОРІЖЖЯ 2024

metinvest polytechnic

РЕФЕРАТ

Пояснювальна записка: 24 сторінок, 5 рисунків, 2 таблиць, 3 додатки, 13 використаних джерел.

Мета проектування – знайомство з фреймворком машинного навчання PyTorch та моделям нейронних мереж для вирішення задач регресії, бінарної класифікації та розпізнавання образів.

Робота складається з чотирьох розділів.

Перший розділ присвячений опису процесу створення регресійної моделі за теоремою універсальної апроксимаці.

У другому розділі побудовано та досліджено нейрону мережу для бінарної класифікації за допомогою автодиференціювання PyTorch.

У третьому розділі розглянуто процес побудови нейронної мережі для класифікації 3 класів зображень із набору даних CIFAR100 за індивідуальним варіантом та настройка гіперпараметрів мережі на тестовій вибірці.

У четвертому розділі наведено відповіді на контрольні питання.

Ключові слова: РЕГРЕСІЙНА МОДЕЛЬ, БІНАРНА КЛАСИФІКАЦІЯ, НЕЙРОННА МЕРЕЖА ГЛИБОКОГО НАВЧАННЯ, ГІПЕРПАРАМЕТРИ МЕРЕЖІ

ABSTRACT

Explanatory note: 24 pages, 5 figures, 2 tables, 3 appendices, 13 references.

Explanatory note: 24 pages, 5 figures, 2 tables, 3 appendices, 13 references.

The purpose of the project is to introduce the PyTorch machine learning framework and neural network models for solving regression, binary classification, and pattern recognition problems.

The paper consists of four chapters.

The first section describes the process of creating a regression model based on the universal approximation theorem.

The second section builds and studies a neural network for binary classification using PyTorch autodifferentiation.

The third section describes the process of building a neural network for classifying 3 classes of images from the CIFAR100 dataset using an individualized option and tuning the network hyperparameters on a test sample.

The fourth section provides answers to the control questions.

Keywords: REGRESSION MODEL, BINARY CLASSIFICATION, DEEP LEARNING NEURAL NETWORK, NETWORK HYPERPARAMETERS

ЗМІСТ

ВСТУП

В сучасному світі штучний інтелект (ШІ) набуває все більшої популярності та значення у різноманітних сферах людської діяльності, від автоматизації виробничих процесів до розробки персоналізованих медичних рішень. Одним із ключових елементів ефективних систем ШІ є нейронні мережі, які здатні моделювати складні залежності та взаємозв'язки великих обсягів даних. Однак, для досягнення оптимальної продуктивності нейронних мереж необхідно правильно вибрати та налаштувати їх гіперпараметри, процес, який вимагає глибокого розуміння як самої моделі, так і специфіки задачі, що вирішується.

Робота присвячена детальному дослідженню та налагодженню гіперпараметрів нейронних мереж у контексті дисципліни "Моделі та методи представлення знань і штучного інтелекту". Метою цієї роботи є аналіз впливу різних гіперпараметрів на ефективність нейронних мереж, включаючи розмір пакету, швидкість навчання, кількість шарів та нейронів в них, функції активації, а також методи оптимізації. Через експериментальний підхід та аналіз результатів ми прагнемо виявити оптимальні конфігурації для різних типів задач, які можуть бути вирішені за допомогою нейронних мереж.

У цій роботі особлива увага приділяється практичним аспектам налагодження гіперпараметрів, включаючи використання методів автоматизованого пошуку та оцінки моделей, що дозволяє значно підвищити ефективність процесу розробки та впровадження нейронних мереж у реальних умовах. Таким чином, дана робота не лише сприяє поглибленому розумінню студентами основ штучного інтелекту та нейронних мереж, а й формує необхідні навички для їх ефективного застосування в майбутніх дослідницьких та професійних проектах.

1 ДОСЛІДЖЕННЯ РЕГРЕСІЙНОЇ МОДЕЛІ ЗА ТЕОРЕМОЮ УНІВЕРСАЛЬНОЇ АПРОКСИМАЦІ

2. РОЗРОБКА ТА ДОСЛІДЖЕННЯ МОДЕЛІ БІНАРНОЇ КЛАСИФІКАЦІЇ ЗА ДОПОМОГОЮ АВТОДИФЕРЕНЦІЮВАННЯ PYTORCH

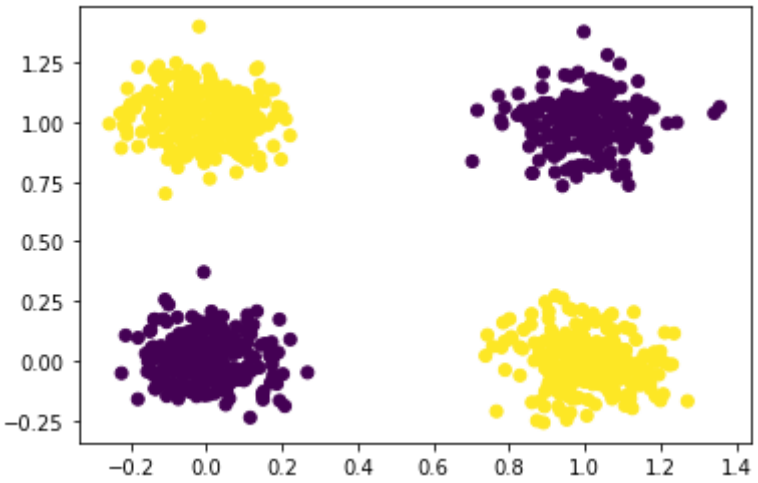


Рисунок 2.1 – Навчальні та тестові набори даних

Таблиця 2.1 – Підсумкова таблиця

Конфігурація неймережі	**Гіперпараметри**	**Точність**	**Коментарі**
FC(10), FC(3)	lr = 0.003, batch_size = 128, epochs = 100	test = 70%, train = 98%	Базовий варіант
FC(10), FC(3)	lr = 0.001, batch_size = 128, epochs = 300	test = 72%, train = 99%	Зменшили швидкість навчання в 3 рази, для компенсації збільшили кількість епох у стільки ж разів

3. РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПОВНОЗВ'ЯЗНОЇ НЕЙРОННОЇ МЕРЕЖУ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ ІЗ НАБОРУ ДАНИХ CIFAR100

4. ВІДПОВІДЬ НА КОНТРОЛЬНІ ПИТАННЯ

ВИСНОВКИ

У ході цієї індивідуальної роботи було проведено глибоке дослідження гіперпараметрів нейронних мереж та їх впливу на ефективність розв'язання задач штучного інтелекту. Експериментальний аналіз підкреслив значущість правильного вибору та налаштування ключових гіперпараметрів, таких як розмір пакету, швидкість навчання, архітектура мережі, функції активації та алгоритми оптимізації, для досягнення оптимальної продуктивності нейронних мереж у різних застосуваннях.

Одним із ключових висновків роботи є те, що не існує універсального набору гіперпараметрів, який би ідеально підходив для всіх типів задач та датасетів. Тому процес налаштування гіперпараметрів вимагає систематичного підходу, включаючи методи автоматизованого пошуку, такі як сітковий пошук, випадковий пошук та байєсівська оптимізація, для виявлення найбільш ефективних конфігурацій.

Крім того, результати дослідження підкреслюють важливість розуміння теоретичних основ нейронних мереж та принципів машинного навчання, що дозволяє більш критично підходити до процесу налаштування моделей та адаптації їх до конкретних задач.

Враховуючи широкий спектр можливостей застосування нейронних мереж у різних областях, від комп'ютерного зору до обробки природної мови, ця робота не тільки сприяє глибшому розумінню студентами основ штучного інтелекту, але й надає практичні навички та знання, необхідні для ефективного застосування цих технологій у майбутньому.

Таким чином, ця індивідуальна робота демонструє, що успіх у застосуванні нейронних мереж значною мірою залежить від глибокого розуміння та вміння налаштувати їх гіперпараметри відповідно до

специфіки задачі та доступних даних, підкреслюючи важливість поєднання теоретичних знань і практичних навичок у галузі штучного інтелекту..

ПЕРЕЛІК ПОСИЛАНЬ

Базові

1 Методи та системи штучного інтелекту: навч. посіб. / укл. Д.В. Лубко, С.В. Шаров. – Мелітополь: ФОП Однорог Т.В., 2019.– 264 с.

2 Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2020.

3 Глибовець А.М., Глибовець М.М., Поляков М.В. Інтелектуальні мережі // НаУКМА, Дніпропетровськ, 2014.

4 Прикладні системи штучного інтелекту / Бібліотека підручників та статтей Posibniki (2022). URL: https://posibniki.com.ua/catalog-prikladni-sistemi-shtuchnogo-intelektu

5 Зайченко Ю.П. Основи проектування інтелектуальних систем. Навч. посібник. – Київ: Видавничий дім «Слово», 2004. – 352с.

6 Руденко О. Г., Бодянський Є. В. Штучні нейронні мережі: Навчальний посібник. — Харків: ТОВ "Компанія СМІТ", 2006. — 404 с.

7 Іванченко Г. Ф. Прикладні системи штучного інтелекту. Навч.посібник. - К.:КНЕУ, 2014.-630 с.

8 Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей: монографія / С. О. Субботін, А. О. Олійник, О. О. Олійник; за заг. ред. С. О. Субботіна. – Запоріжжя: ЗНТУ, 2009. – 375 с.

Додаткові

9 Бондаренко М. Ф. Мозгоподобные структуры : справочное пособие. / М. Ф. Бондаренко, Ю. П. Шабанов-Кушнаренко. – Т. 1. – Київ : Наукова думка, 2011. – 460 с.

10 Alex Smola and S.V.N. Vishwanathan. Introduction to Machine Learning. Cambridge University Press 2008. https://alex.smola.org/drafts/thebook.pdf

11 Shai Shalev-Shwartz, Shai Ben-David. UNDERSTANDING MACHINE LEARNING. From Theory to Algorithms. Cambridge University Press 2014. https://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf

12 Методичні вказівки для самостійної роботи з курсу «Теорія інтелекту» : для студ. спец. 121 «Інженерія програмного забезпечення»

/ О. Ю. Чередніченко, Д. Л. Орловський, А. М. Копп ; Харківський політехнічний ін-т, нац. техн. ун-т. – Харків : НТУ «ХПІ», 2017. – 76 с.
13 Козуля Т.В. Формування знання-орієнтованого інформаційного забезпечення досліджень складних систем" / Т.В. Козуля, Н.В. Шаронова , М.М. Козуля // Системні дослідження та інформаційні технології – 2017, №3 – С.37-46.
14 Дорошенко А.Ю. Розробка програмних компонентів інформаційної системи екстракції фактографічних даних з веб-ресурсів / А.Ю.Дорошенко, Н.В. Шаронова, Б.О. Єна, О.В. Янголенко // Проблеми інформаційних технологій. – 2018. – №1 (023). – С.27-35.
Web-ресурси
15 Guide to the SWEBOK // https://www.computer.org/education/bodies-of-knowledge/software-engineering
16 Information FrameWork model overview – IBM // https://www.ibm.com/analytics/industry-models
17 https://www.aaai.org/home.html
18 http://www.lelandl.standford.edu/group/STS/global.htm l
19 http://www.lambent.com/Systems/sysbib.htm
20 http://pespmcl.vub.ac.be/EVOCOPUB.html
21 http://pespmcl.vub.ac.be/SYSTHEOR.html
22 http://www.aries.eu.int
23 http://www.trader.recl.com