Лабораторна робота №6

**Реалізація архітектури AlexNet CNN за допомогою TensorFlow і Keras**

**Мета:** Отримати навички реалізації архітектури AlexNet CNN з використанням бібліотек TensorFlow та Keras

**ТЕОРЕТИЧНІ ВІДОМОСТІ:**

**Що таке AlexNet?**

AlexNet - згорткова нейронна мережа, яка надала великий вплив на розвиток машинного навчання, особливо - на алгоритми комп'ютерного зору. Мережа, яку створили Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever, з великим відривом виграла конкурс LSVRC-2012 з розпізнавання зображень ImageNet у 2012 році (з кількістю помилок 15,3% проти 26,2% у другого місця).

Архітектура AlexNet схожа з створеної Yann LeCum мережею LeNet. Однак у AlexNet більше фільтрів на шарі і вкладених згортальних шарів. Мережа включає в себе згортки, максимальне об'єднання, дропаутов, аугментацію даних, функції активацій ReLU і стохастичний градієнтний спуск.

**Особливості AlexNet**

1. Як функція активації використовується Relu замість арктангенса для додавання в модель нелінійності. За рахунок цього при однаковій точності методу швидкість стає в 6 разів швидше.
2. Використання дропаутов замість регуляризації вирішує проблему перенавчання. Однак час навчання подвоюється з показником дропаутів 0,5.
3. Проводиться перекриття об'єднань для зменшення розміру мережі. За рахунок цього рівень помилок першого і п'ятого рівнів знижуються до 0,4% і 0,3%, відповідно.

**Датасета ImageNet**

ImageNet - набір з 15 мільйонів помічених зображень з високою роздільною здатністю, розділених на 22 000 категорій. Зображення зібрані в інтернеті та позначені вручну за допомогою краудсорсинга Amazon's Mechanical Turk. Починаючи з 2010 року проводиться щорічний конкурс ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), який є частиною Pascal Visual Object Challenge. У Челлендж використовується частина датасета ImageNet від 1000 зображень в кожній з 1000 категорій. Всього виходить 1,2 мільйона зображень для навчання, 50 000 зображень для перевірки і 150 000 - для тестування. ImageNet складається із зображень з різним розміром. Тому для конкурсу їх масштабують до фіксованого розміру 256 × 256. Якщо спочатку зображення було прямокутним, то його обрізають до квадрата в центрі зображення.

**Архітектура**

Мережа має біля 3 000000 параметрів, 650000 нейронів і її навчання в 2012 році займало від п'яти до семи днів.

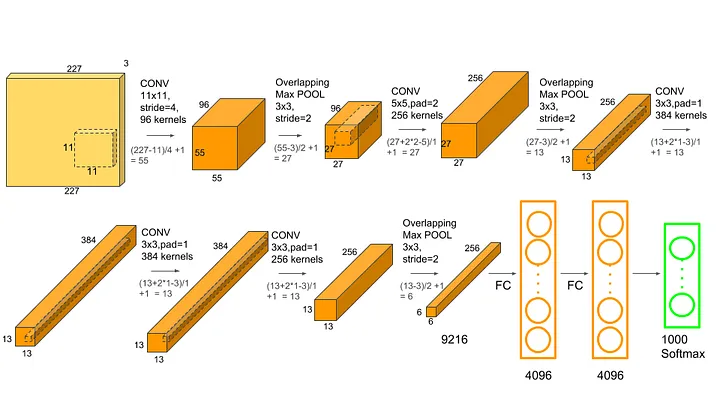
Архітектура мережі приведена на рис. 1.

Рисунок 1 — Архітектура AlexNet

Як видно, AlexNet складається із п’яти згорткових шарів і трьох повнозв' язних шарів:

1) Входом для AlexNet є RGB-зображення розміром 224×224×3, яке пропускається через перший згортковий шар з 96 картами ознак з фільтрами розмірами 11×11 і кроком 4. На виході отримується 55×55×96-тензор до якого застосовується агрегуючий оператор (max-pool) з ядром 3×3 і кроком 2. В результаті отримується 27×27×96-тензор який передається в другий шар.

2) Далі йде другий згортковий шар з 256 карт розміром 5×5 та кроком 1, за яким йде агрегуючий шар з 3×3-ядром та кроком 2. На виході отримується 13×13×256-тензор.

3, 4, 5) Третій, четвертий та пʼятий шари - це згорткові шари з розміром ядра 3×3 та кроком 1. Перші два шари мають 384 карти ознак, а третій використовував 256 карт ознак. За трьома згортковими шарами агрегований шар обʼєднання з розміром ядра 3×3, кроком 2 і який має 256 карт ознак

6) Згортковий шару вихід якого є повністю зʼєднаний тар з 9216 картами ознак розміром 1×1. Також застосовується процедура Dropout прорідження кількості нейронів в шарі для уникнення ефекту перенавчання.

7, 8) Далі знову два повноз’єднані шари з 4096 нейронами

9) Нарешті, вихідний шар з 1000 нейронів і функцією активації softmax.

Тобто **AlexNet** містить **вісім шарів** з ваговими коефіцієнтами. Перші п'ять з них згорткові, а решта три - повнозв'язні. Вихідні дані пропускаються через функцію втрат **softmax**, яка формує розподіл 1000 міток класів. Мережа максимізує багатолінійну логістичну регресію, що еквівалентно максимізації середнього по всім навчальним випадків логарифма ймовірності правильного маркування з розподілу очікування. Ядра другого, четвертого і п'ятого згортальних шарів пов'язані тільки з тими картами ядра в попередньому шарі, які знаходяться на одному і тому ж графічному процесорі. Ядра третього згорткового шару пов'язані з усіма картами ядер другого шару. Нейрони в повнозв'язних шарах пов'язані з усіма нейронами попереднього шару.

Таким чином, AlexNet містить 5 згорткових шарів і 3 повнозв'язних шарів. **Relu** застосовується після кожного згорткового і повнозв'язного шару. Дропаутов застосовується перед першим і другим повнозв'язну шарами. Мережа містить 62,3 мільйона параметрів і витрачає 1,1 мільярда обчислень при прямому проході. Згорткові шари, на які припадає 6% всіх параметрів, виробляють 95% обчислень.

Переваги AlexNet:

1) AlexNet вважається віхою CNN для класифікації зображень;

2) Багато методів, наприклад conv+pooling design, dropout, GPU, паралельні обчислення, ReLU, все ще є промисловим стандартом комп’ютерного зору;

3) Унікальною перевагою AlexNet є пряме введення зображення в модель класифікації;

4) Шари згортки можуть автоматично виділяти краї зображень, а повністю пов’язані шари вивчають ці функції;

5) Теоретично складність візуальних шаблонів можна ефективно витягти, додавши більше конвлейера.

Недоліки AlexNet:

1) AlexNet НЕ достатньо глибокий порівняно з пізнішими моделями, такими як VGGNet, GoogLENet і ResNet;

2) Використання нормального розподілу для ініціювання вагових коефіцієнтів у нейронних мережах не може ефективно вирішити проблему зникнення градієнта, пізніше замінене методом Ксав’є;

3) Продуктивність перевершують більш складні моделі, такі як GoogLENet (6,7%) і ResNet (3,6%).

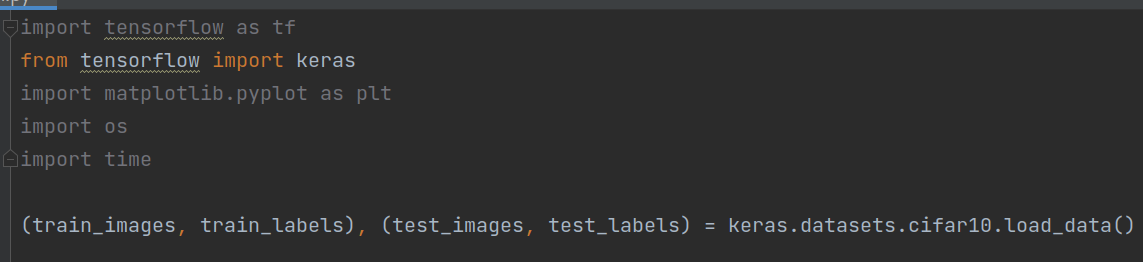
Налаштування середовища перед виконанням:

1) pip3 install keras==2.4.3 numpy==1.19.3 pillow==7.0.0 scipy==1.4.1 h5py==2.10.0 matplotlib==3.3.2 opencv-python keras-resnet==0.2.0

2) pip3 install tensorflow-cpu==2.4.0

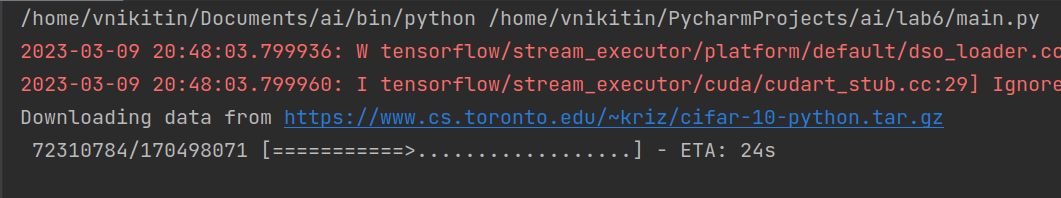
Процедура:

1) створимо новий модуль та пропишемо необхідні імпорти і завантаження датасету (рис. 1)

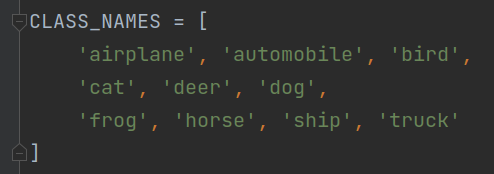
Рисунок 1 — Імпорти та завантаження датасету

Набір даних CIFAR-10 містить 60 000 кольорових зображень, кожне з розмірами 32x32px. Вміст зображень у наборі даних вибрано з 10 класів. Зображення CIFAR-10 були агреговані одними з творців мережі AlexNet Алексом Крижевським і Джеффрі Хінтоном.

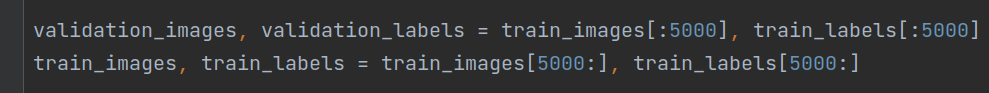
Якщо запустити скрипт на даному етапі, то буде схожий вивід, як на рис. 2.

Рисунок 2 — Завантаження датасету

2) Щоб посилатися на імена класів зображень на етапі візуалізації, список Python, що містить класи, ініціалізується ім’ям змінної CLASS\_NAMES (рис. 2).

Рисунок 3 — Ініціалізація змінної CLASS\_NAMES

3) За замовчуванням набір даних CIFAR розділений на 50 000 навчальних даних і 10 000 тестових даних. Останнім розділом набору даних, який нам потрібен, є дані перевірки. Дані перевірки отримані шляхом отримання останніх 5000 зображень у навчальних даних.

Рисунок 4 — Отримання перевірочних даних

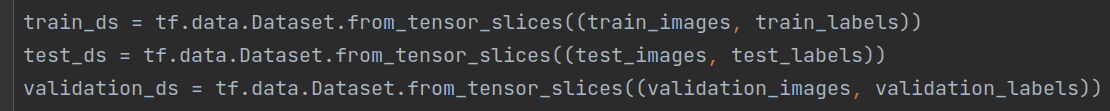
Навчальний набір даних: це група нашого набору даних, яка використовується для безпосереднього навчання нейронної мережі. Навчальні дані стосуються розділу набору даних, доступного для нейронної мережі під час навчання.

Набір даних перевірки: ця група набору даних використовується під час навчання для оцінки продуктивності мережі на різних ітераціях.

Тестовий набір даних: цей розділ набору даних оцінює продуктивність нашої мережі після завершення фази навчання.

4) TensorFlow надає набір функцій і операцій, які дозволяють легко маніпулювати даними та модифікувати їх через визначений конвеєр введення.

Щоб отримати доступ до цих методів і процедур, ми повинні перетворити наш набір даних на ефективне представлення даних, яке знайоме TensorFlow. Це досягається за допомогою tf.data.Dataset API (рис. 5).

Рисунок 5 — Перетворення набору даних на TensorFlow представлення

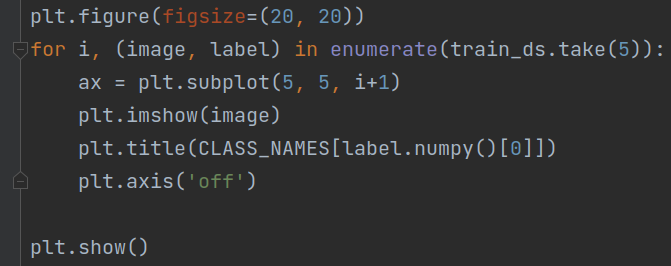
Зокрема, метод tf.data.Dataset.from\_tensor\_slices бере розділи набору даних навчання, тестування та перевірки та повертає відповідне представлення набору даних TensorFlow.

5) Попередня обробка в будь-якому машинному навчанні пов'язана з перетворенням даних з однієї форми в іншу.

Зазвичай попередня обробка проводиться, щоб переконатися, що дані, що використовуються, знаходяться у відповідному форматі.

Спочатку візуалізуємо зображення в наборі даних CIFAR-10.

Наведений нижче фрагмент коду використовує бібліотеку Matplotlib для представлення інформації про пікселі даних із п’яти навчальних зображень у фактичних зображеннях. Також є індикатор класу, до якого належить кожен зображений вміст на зображеннях.

Рисунок 6 — Представлення інформації про пікселі даних

**Якщо зображення не виводиться, то можно спробувати встановити наступний пакет**:

pip3 install pyqt5

Результат виклику plt.show() зображено на рис. 7.

Рисунок 7 — Вивід CIFAR-10 зображень

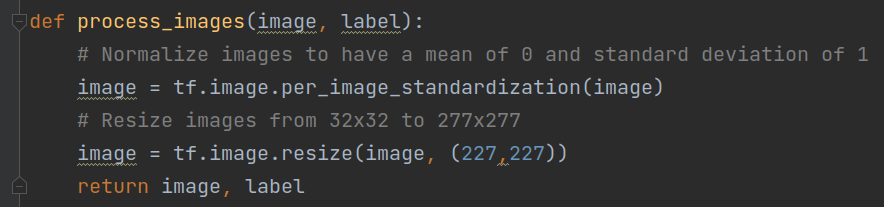
6) Основні перетворення попередньої обробки, які будуть накладені на дані, представлені в мережі:

- Нормалізація та стандартизація зображень;

- Зміна розміру зображень з 32x32 до 227x227. Мережевий вхід AlexNet очікує зображення 227x227.

Ми створимо функцію під назвою process\_images (рис. 8).

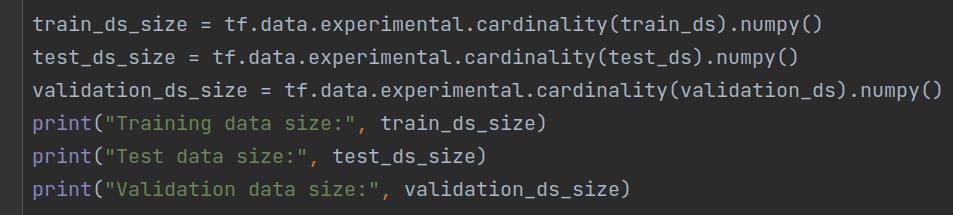
Ця функція виконуватиме всю попередню обробку даних, яку ми потребуємо. Ця функція викликається далі в процесі машинного навчання.

Рисунок 8 — Реалізація process\_image

7) створення пайплайну

Конвеєр введення/даних описується як серія функцій або методів, які викликаються послідовно одна за одною. Вхідні конвеєри – це ланцюжок функцій, які або діють на дані, або примусово виконують операцію з даними, що проходять через конвеєр.

Давайте отримаємо розмір кожного розділу набору даних, який ми створили; розміри розділів набору даних потрібні для того, щоб набір даних був ретельно перетасований перед передачею через мережу (рис. 9).

Рисунок 9 — Отримання розміру кожного розділу даних

8) Для нашого основного конвеєра вводу/даних ми виконаємо три основні операції (рис. 10):

- Попередня обробка даних датасету;

- Перемішування **датасету;**

- Пакет даних у датасеті.

Рисунок 10 — Виконання операцій

9) Реалізація архітектури AlexNet CNN з нуля

Завдяки використанню Keras Sequential API ми можемо реалізувати послідовні шари нейронної мережі в наших моделях, які накладаються один на одного.

Ось типи рівнів, з яких складається архітектура AlexNet CNN, а також короткий опис:

- Згортковий шар: згортка — це математичний термін, який описує множення скалярного добутку між двома наборами елементів. У рамках глибокого навчання операція згортки діє на фільтри/ядра та масив даних зображення в межах згорткового рівня. Тому згортковий шар — це просто шар, на якому виконується операція згортки, яка відбувається між фільтрами та зображеннями, що проходять через згорточну нейронну мережу.

- Шар пакетної нормалізації: пакетна нормалізація — це техніка, яка пом’якшує вплив нестабільних градієнтів у нейронній мережі за допомогою введення додаткового рівня, який виконує операції над вхідними даними попереднього рівня. Операції стандартизують і нормалізують вхідні значення, після чого вхідні значення перетворюються за допомогою операцій масштабування та зсуву.

- Шар MaxPooling: максимальне об’єднання — це варіант підвибірки, де максимальне піксельне значення пікселів, які потрапляють у сприйнятливе поле одиниці в межах рівня підвибірки, береться як вихід. Наведена нижче операція максимального об’єднання має вікно 2x2 і ковзає по вхідних даних, виводячи середнє значення пікселів у приймальному полі ядра.

- Зведення шару: приймає вхідну форму та зводить дані вхідного зображення в одновимірний масив.

- Щільний шар: щільний шар має вбудовану кількість довільних одиниць/нейронів. Кожен нейрон є перцептроном.

Варто згадати деякі інші операції та методи, які використовуються в AlexNet CNN:

- Функція активації: математична операція, яка перетворює результат або сигнали нейронів у нормалізований результат. Метою функції активації як компонента нейронної мережі є введення нелінійності в мережу. Включення функції активації дозволяє нейронній мережі мати більшу репрезентативну потужність і вирішувати складні функції.

- Функція активації Rectified Linear Unit (ReLU): тип функції активації, яка перетворює результати значення нейрона. Перетворення, накладене ReLU на значення з нейрона, представлено формулою y=max(0,x). Функція активації ReLU зменшує будь-які негативні значення нейрона до 0, а позитивні значення залишаються незмінними. Результат цього математичного перетворення використовується як вихідні дані поточного рівня та використовується як вхідні дані для наступного рівня в нейронній мережі.

- Функція активації Softmax: тип функції активації, яка використовується для отримання розподілу ймовірностей набору чисел у вхідному векторі. Результатом функції активації softmax є вектор, у якому його набір значень представляє ймовірність появи класу або події. Сума всіх значень у векторі дає 1.

- Вилучення: Техніка виключення працює шляхом випадкового зменшення кількості взаємопов’язаних нейронів у нейронній мережі. На кожному кроці навчання кожен нейрон має шанс залишитися поза увагою або, точніше, виключити зібрані внески від підключених нейронів.

Код на рис. 11 представляє реалізацію Keras архітектури AlexNet CNN.

Рисунок 11 — Архітектура **AlexNet CNN**

10) TensorBoard

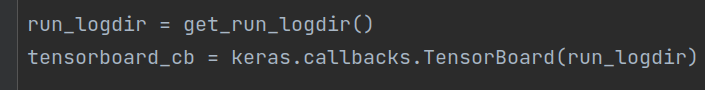
Перш ніж перейти до навчання, валідації та оцінки мережі з даними, ми спочатку повинні налаштувати деякі засоби моніторингу.

TensorBoard — це інструмент, який надає набір механізмів візуалізації та моніторингу. Для роботи в цій лабораторній ми будемо використовувати TensorBoard для моніторингу прогресу навчання мережі.

Зокрема, ми будемо контролювати такі показники: втрата навчання, точність навчання, втрата підтвердження, точність підтвердження.

У наведеному нижче фрагменті короткого коду ми створюємо посилання на каталог, у якому ми хочемо зберігати всі файли TensorBoard. Функція get\_run\_logdir повертає розташування точного каталогу, назва якого відповідає поточному часу початку фази навчання.

Щоб завершити цей поточний процес, ми передаємо каталог для зберігання пов’язаних файлів TensorBoard для конкретного навчального сеансу зворотному виклику TensorBoard.

Рисунок 12 — Моніторинг прогресу навчання

11) компіляція мережі

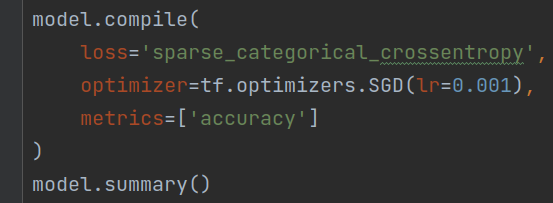
Процеси компіляції передбачають визначення наступного:

- Функція втрати: метод, який кількісно визначає, «наскільки добре» працює модель машинного навчання. Кількісна оцінка – це результат (вартість), заснований на наборі вхідних даних, які називаються значеннями параметрів. Значення параметрів використовуються для оцінки прогнозу, а «втрата» — це різниця між прогнозами та фактичними значеннями;

- Алгоритм оптимізації: оптимізатор у нейронній мережі — це алгоритмічна реалізація, яка полегшує процес градієнтного спуску в нейронній мережі шляхом мінімізації значень втрат, які надаються через функцію втрат. Для зменшення втрат надзвичайно важливо, щоб значення вагових коефіцієнтів у мережі були вибрані належним чином;

- Швидкість навчання: невід’ємний компонент реалізації нейронної мережі, оскільки це значення фактора, яке визначає рівень оновлень значень ваг мережі. Швидкість навчання є різновидом гіперпараметра.

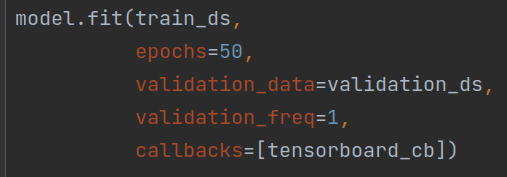
Ми також можемо надати короткий опис мережі, щоб мати більше уявлення про склад шару мережі, запустивши функцію model.summary() (рис. 13).

Рисунок 13 — Компіляція та опис мережі

12) навчання

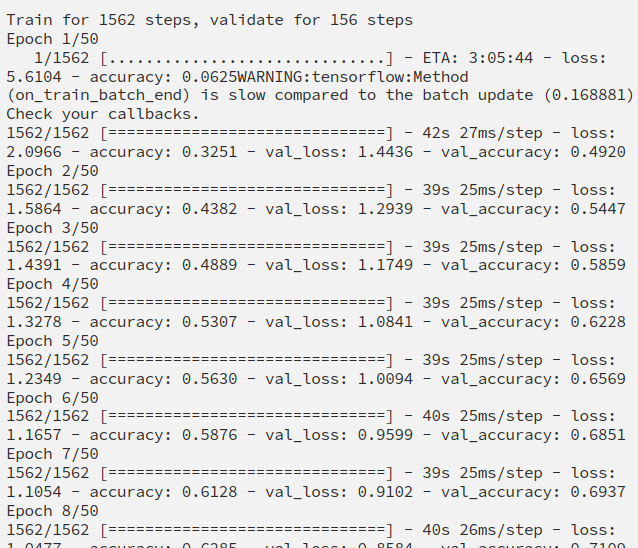
Навчання спеціальної мережі AlexNet дуже просте за допомогою модуля Keras, доданий через TensorFlow. Ми просто повинні викликати метод fit() і передати відповідні аргументи.

Епоха - це числове значення, яке вказує кількість разів, коли мережа була піддана всім точкам даних у навчальному наборі даних.

Рисунок 14 — Тренування мережі

Після виконання цієї клітинки коду в блокноті мережа почне навчатися та перевіряти надані дані. Ви почнете бачити журнали навчання та перевірки, як на рис. 15.

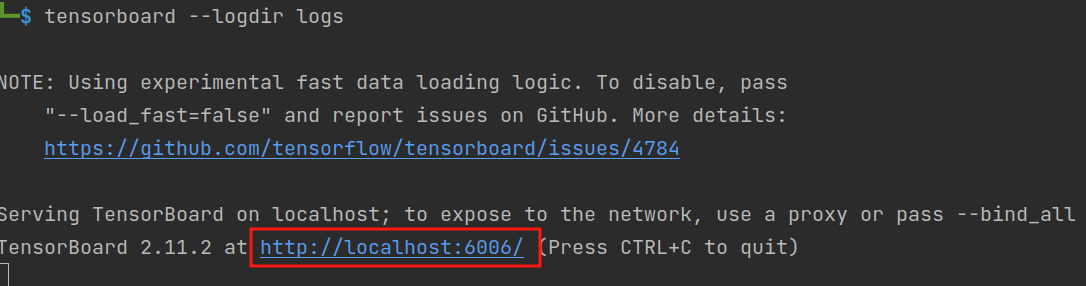
Після виконання цієї клітинки коду в блокноті мережа почне навчатися та перевіряти надані дані (рис. 15).

Рисунок 15 — Навчальні та валідаційні логи

Для кращої візуалізації та моніторингу ефективності навчання можно скористатись функціоналом TensorBoard.

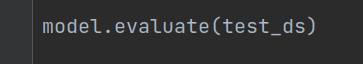
Потрібно ввести у терміналі наступне:

tensorboard --logdir logs

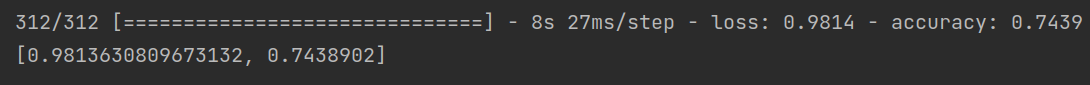
Рисунок 16 — Запуск сервера TensorBoard

13) оцінка

Етап оцінювання надасть оцінку продуктивності навченої моделі на основі тестових даних. На етапі оцінювання моделі ми будемо використовувати пакет тестових даних, створених на попередніх етапах (рис. 17).

Рисунок 17 — Оцінка продуктивності

Після виконання ми отримуємо оцінку, яка вказує на продуктивність моделі на тестових даних (рис. 18).

Рисунок 18 - Точність

**ПІДГОТОВКА ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №6**

Перед виконанням лабораторної роботи рекомендується ознайомитись з відповідним розділом лекційного матеріалу курсу та засвоїти теоретичний матеріал.

**ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ**

Звіт з лабораторної роботи обов’язково повинен містити наступну інформацію:

* назва комп’ютерного практикуму;
* мета роботи;
* відповіді на завдання у текстовому форматі та графічними зображеннями за необхідності.

**Завдання на лабораторну роботу:**

1. Реалізувати засобами TensorFlow та Keras AlexNet;
2. Отримати оцінку точності навченої мережі.

**Контрольні запитання:**

1. Що таке AlexNet?

2. Яка архітектура AlexNet?

3. Які переваги та недоліки даної мережі?

4. Навіщо потрібна компіляція мережі?

5. Як відбувається навчання?

**Навчальні матеріали та ресурси:**

1. https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98

2. https://gist.github.com/hsleonis/ad672eadbc1827095d4f1aefa49936b9