1 ПРАКТИЧНА РЕАЛІЗАЦІЯ

Даний розділ присвячується аналізу часового ряду тестувальної системи DOTS за допомогою мови програмування Python.

Python — це легка в освоєнні, потужна мова програмування. Він має ефективні структури даних високого рівня і простий, але ефективний підхід до об'єктно-орієнтованого програмування [7].

Python став загальноприйнятою мовою програмування для багатьох сфер застосування науки про дані. Дана мова програмування поєднує в собі міць мов програмування з простотою використання предметно орієнтованих скриптових мов типу MATLAB або R.

В Python є бібліотеки для завантаження даних, візуалізації, статистичних обчислень, обробки природної мови, обробки зображень і багато чого іншого [8].

Також використовувався фреймфорк Anaconda. Це безкоштовний, включаючи комерційне використання, і готовий до використання в середовищі підприємства дистрибутив Python, який об'єднує всі ключові бібліотеки Python, необхідні для роботи в області науки про дані, математики та розробки, в одному зручному для користувача крос-платформенном дистрибутиві [9].

Для тренування нейронних мереж було обрано бібліотеку PyTorch. PyTorch - це бібліотека для програм Python, яка сприяє побудові проектів глибокого навчання [10].

Також використовувалась бібліотеки NumPy та matplotlib. NumPy є основним пакетом для наукових обчислень з Python [11]. Matplotlib — це всебічна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій у Python [12].

1.1 Дані для аналізу

Для демонстрації та аналізу роботи алгоритмів були використані дані з датасету CIFAR-10.

Набір даних CIFAR-10 складається з 60000 кольорових зображень

розміром 32x32 у 10 класах, по 6000 зображень на клас. Існує 50000 навчальних зображень та 10000 тестових зображень [13].

Набір даних розділений на п'ять навчальних партій та одну тестову партію, кожна з 10000 зображень. Тестова партія містить рівно 1000 довільно обраних зображень з кожного класу. Навчальні партії містять решту зображень у довільному порядку, але деякі навчальні партії можуть містити більше зображень одного класу, ніж інші [13]. Навчальні партії містять рівно

жного класу.

ень з датасету наведено у рис. **??**.

Рисунок 1.1 – Приклад зображень з датасету CIFAR-10

1.2 Аналіз роботи алгоритму Deep Infomax

Робота методу Deep InfoMax буде розглядатись в залежності від значеть гіперпараметрів α, β таγ, а також коефіцієнту швидкості навчання. В кожному експерименті кількість епох навчання нейронної мережі складатиме 300.

Якщо взяти $\alpha = 0,1\,\beta = 0,1$та$\gamma = 0,1$, то помилка на тренувальних даних буде 12,73 % (див. рис. **??**) та модель тренувалась приблизно 14 годин.

На рис. **??** можна побачити, що модель доволі погано узагальнює результати.

Якщо взяти $\alpha = 0,5\,\beta = 0,5$та$\gamma = 0,5$, то помилка на тренувальних даних буде 8,05 % (див. рис. **??**) та модель тренувалась приблизно 11 годин.

Рисунок 1.2 – Залежність помилки від епохи навчання при $\alpha = 0,1;\ \beta = 0,1;\ \gamma = 0,1$

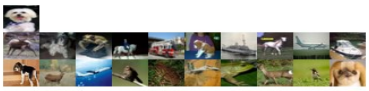

Рисунок 1.3 – Результати тестування при $\alpha = 0,1;\ \beta = 0,1;\ \gamma = 0,1$

При $\alpha = 0,5\,\beta = 0,9$та$\gamma = 0,1$ помилка на тренувальних даних буде 1,28 % (див. рис. **??**) та модель тренувалась приблизно 6 годин.

Якщо взяти $\alpha = 0,5\,\beta = 1$та$\gamma = 0,1$, то помилка на тренувальних даних буде 1,43 % (див. рис. **??**) та модель тренувалась приблизно 6 годин.

Отже, найкращі результати модель отримує при $\alpha = 0,5\,\beta = 0,9$та$\gamma = 0,1$. Результати тестування з даними гіперпараметрами наведено на рис. **??**.

Усі вищенаведені результати були отримані з урахуванням коефіцієнту швидкості навчання 0,001. Розглянемо результати роботи алгоритму з

Рисунок 1.4 – Залежність помилки від епохи навчання при $\alpha = 0,5$; $\beta = 0,5$; $\gamma = 0,5$

оптимальними гіперпараметрами з урахуванням інших можливих значень цього коефіцієнта.

Якщо взяти коефіцієнт швидості навчання 0,003 то помилка на тренувальних даних буде 1,35 % (див. рис. **??**) та модель тренувалась приблизно 6 годин.

1.3 Аналіз роботи алгоритму Momentum Contrast

1.4 Висновки за розділом

В даному розділі була продемонстрована робота алгоритмів Deep InfoMax та Momentum Contrast. БУли реалізовані програми мовою програмування Python за допомогою спеціальних бібліотек для аналізу даних та програмного фреймворку Anaconda.

Як видно з результатів, модель Deep InfoMax дає якісніший прогноз, в той час як алгоритм Momentum Contrast потребує менше часових та обчислювальних ресурсів.

Рисунок 1.5 – Залежність помилки від епохи навчання при $\alpha = 0,5;\ \beta = 0,9;\ \gamma = 0,1$

Рисунок 1.6 – Залежність помилки від епохи навчання при $\alpha = 0,5;\ \beta = 1;\ \gamma = 0,1$

Рисунок 1.7 – Результати тестування при $\alpha = 0,5$; $\beta = 1$; $\gamma = 0,1$

Рисунок 1.8 – Залежність помилки від епохи навчання при коефіцієнті швидкості навчання 0,003