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 5000 зображень від кожного класу.

Приклад зображень з датасету наведено у рис. 1.1.

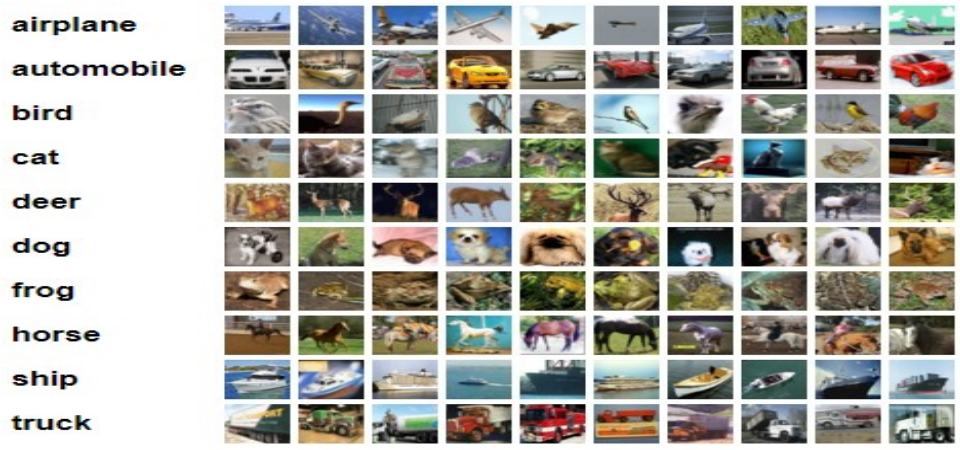

Рисунок 1.1 – Приклад зображень з датасету CIFAR-10

1.2 Аналіз роботи алгоритму Deep Infomax

Робота методу Deep InfoMax буде розглядатись в залежності від значень гіперпараметрів α, β та γ, 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 % та модель тренувалась приблизно 14 годин. На рис. 1.6 можна побачити, що модель доволі погано узагальнює результати.

Рисунок 1.2 – Результати тестування при $\alpha = 0,1$; $\beta = 0,1$; $\gamma = 0,1$

У таблиці 1.1 представлено результати тестування при різних гіперпараметрах:

Таблиця 1.1 – Довірні інтервали параметрів моделі

α	β	γ	Помилка %	Час год.
0,1	0,1	0,1	12.73	14
0,3	0,3	0,3	14,88	12
0,5	0,5	0,5	7,56	11
0,5	0,1	0,3	8,34	12
0,6	0,6	0,6	4,43	9
0,6	0,1	0,3	6,74	9
0,7	0,7	0,7	3,98	8
0,7	1	0,7	5,22	9
0,7	1	0,3	6,72	8
0,7	1	0,1	3,16	7
0,5	0,7	0,1	2,57	6
0,5	0,9	0,1	1,28	6
0,5	1	0,1	1,45	6

Як можна спостерігати, найменша помилка отримується при $\alpha = 0,5$, $\beta = 0,9$ та $\gamma = 0,1$. Результати тестування можна побачити на рис. 1.7. Помилка на тестувальних даних складала 36,96 %.

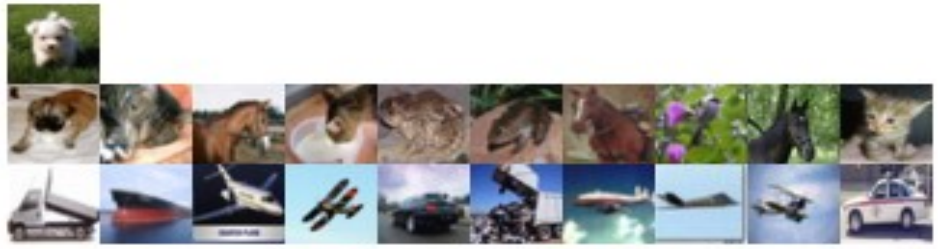

Рисунок 1.3 – Результати тестування при $\alpha = 0,1$; $\beta = 0,1$; $\gamma = 0,1$

Дані результати були отримані за умови, що коефіцінет швидкості навчання дорівнював 0,001. На рис. 1.4 представлено залежність якості тренування від значень даного параметру.

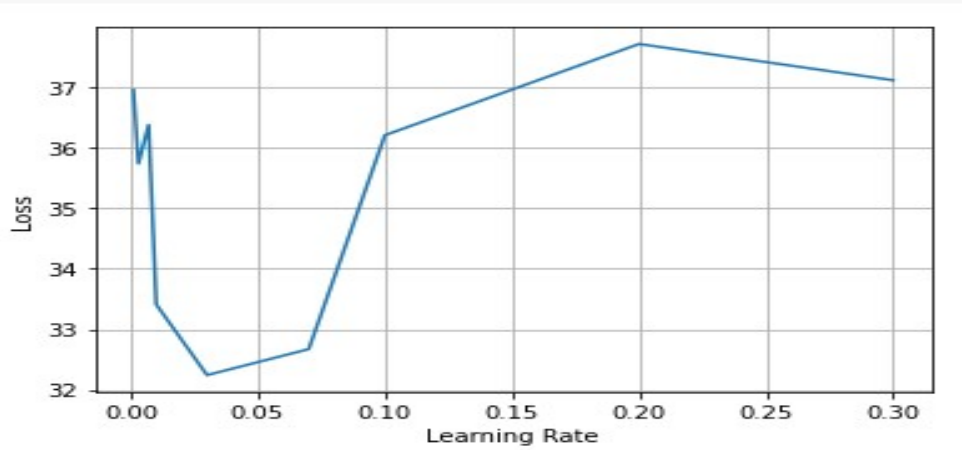

Рисунок 1.4 – Залежність якості тренування від коефіцієнту швидкості навчання

З рис. 1.4 видно, що найкращі результати отримуються при коефіцієнті 0,03. На рис. 1.9 можна бачити результати тестування. Помилка в такому випадку складає 32,24 %та час тренування — приблизно 5 годин.

Рисунок 1.5 – Результати тестування при оптимальних параметрах

1.3 Аналіз роботи алгоритму Momentum Contrast

Робота методу Momentum Contrast буде розглядатись в залежності від значень гіперпараметру τ, та коефіцієнту швидкості навчання. В кожному експерименті кількість епох навчання нейронної мережі складатиме 300.

Якщо взяти $\tau = 0,01$, то помилка на тренувальних даних буде 17,64 % та модель тренувалась приблизно дев'ять годин. На рис. 1.6 можна побачити, що модель доволі погано узагальнює результати.

Рисунок 1.6 – Результати тестування при $\tau = 0,01$

У таблиці 1.2 представлено результати тестування при різних гіперпараметрах:

Таблиця 1.2 – Довірні інтервали параметрів моделі

τ	Помилка %	Час год.
0,01	17.64	9
0,03	15,69	8
0,05	14,97	8
0,07	12,66	7
0,1	8,73	6
0,3	7,43	5
0,7	5,71	4
0,9	6,38	4

Як можна спостерігати, найменша помилка отримується при $\tau = 0,7$. Результати тестування можна побачити на рис. 1.7. Помилка на тестувальних даних складала 40,34 %.

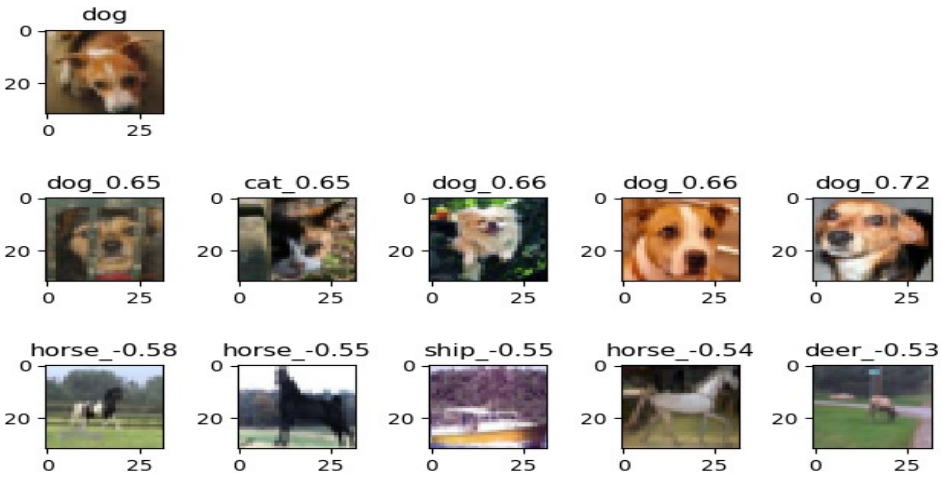

Рисунок 1.7 – Результати тестування при $\tau = 0,7$

Дані результати були отримані за умови, що коефіцінет швидкості навчання дорівнював 0,001. На рис. 1.8 представлено залежність якості тренування від значень даного параметру.

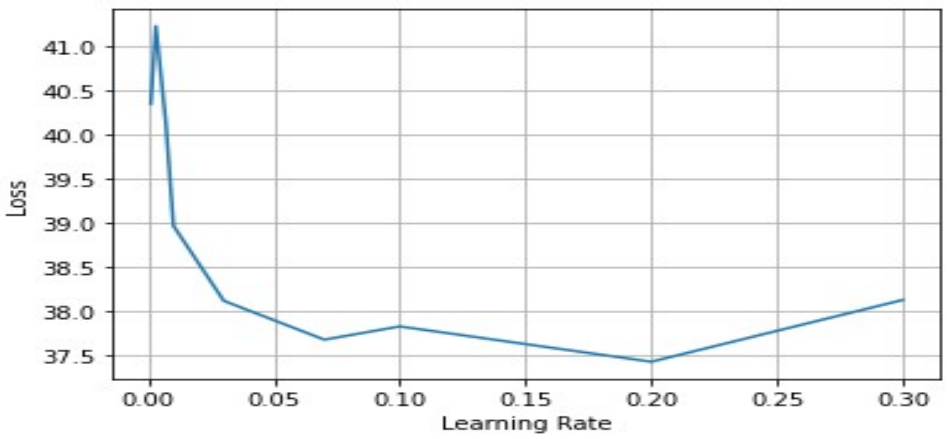

Рисунок 1.8 – Залежність якості тренування від коефіцієнту швидкості навчання

З рис. 1.8 видно, що найкращі результати отримуються при коефіцієнті 0,2. На рис. **??** можна бачити результати тестування. Помилка в такому випадку складає 37,42 %та час тренування — приблизно чотири години.

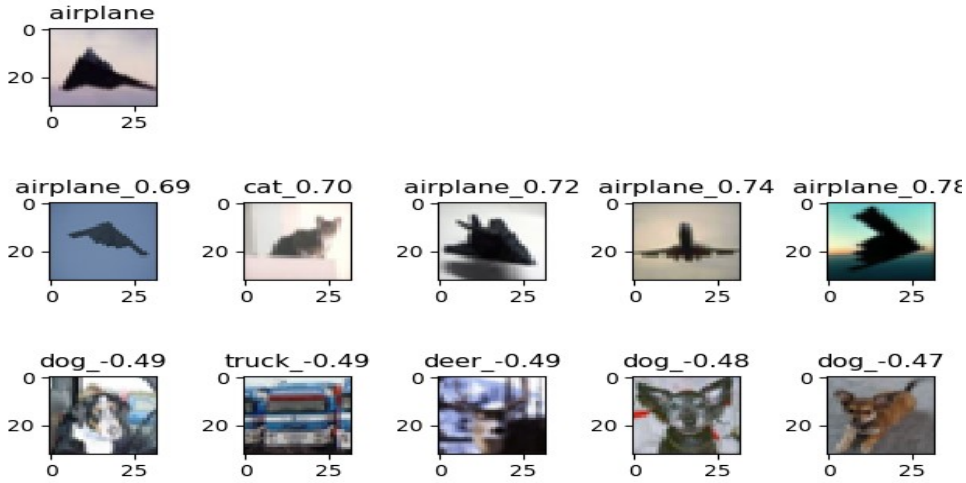

Рисунок 1.9 – Результати тестування при оптимальних параметрах

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.