**Звіт до комп’ютерного практикуму №1.**

**Моделі машинного навчання**

**ПІБ: Головня Олександр Ростиславович**

**Група: ІП-11**

**Мета роботи:** ознайомитись з принципами функціонування, створення, навчання та використання моделей машинного навчання.

***З*авдання:** для обраної задачі класифікації (або регресії) на основі типового датасету створити модель машинного навчання, навчити її на датасеті, перевірити результат на тестовій вибірці.

**Номер варіанту: 15**

**Завдання для варіанту: CIFAR10 Двошаровий персептрон**

**Засоби виконання практикуму:** середовище, мова, фреймворк, якщо використовувався

**Набір даних (датасет) CIFAR10:** Містить 60000 зображень розміром 32х32 пікселя, розділених на 10 класів, по 6000. Кожне зображення належить до одного з наступних класів: літак, автомобіль, птах, кіт, олень, собака, жаба, конь, корабель, вантажівка. Набір даних розділений на дві частини: 50 000 зображень для навчання та 10 000 для тестування.

**Задача:** класифікація кольорових зображень

**Попередня обробка даних:** Масштабування значень пікселів(Значення пікселів зображень мають діапазон від 0 до 255, де 0 відповідає чорному, а 255 - білому. Масштабування їх до діапазону від 0 до 1 полегшує процес навчання моделі)

1. **Модель машинного навчання:** коротка характеристика, структура, вхідні вихідні параметри, програмний код створення.  
   **Вхідний шар**: Вхідні дані представляються у вигляді зображень розміром 32x32 пікселі з трьома каналами (RGB). Це означає, що вхідний шар має форму **(32, 32, 3)**.
2. **Прихований шар**: Це шар **Dense** з 128 нейронами та функцією активації ReLU.
3. **Вихідний шар**: Це останній шар **Dense** з 10 нейронами (відповідає кількості класів у CIFAR-10) без функції активації.

model = tf.keras.Sequential([

tf.keras.layers.Flatten(input\_shape=(32, 32, 3)),

tf.keras.layers.Dense(128, activation='relu'),

tf.keras.layers.Dense(10)

])

**Навчання моделі:** обґрунтування вибору алгоритму навчання, функції втрат, метрик тощо.

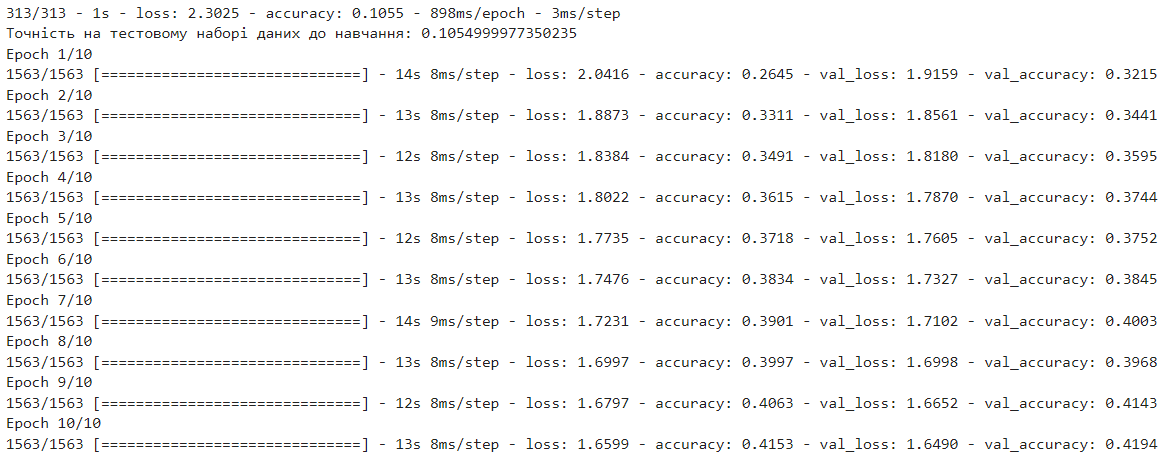
Алгоритм оптимізації: Для цієї моделі використовується алгоритм оптимізації Adam. Adam є популярним алгоритмом оптимізації для нейронних мереж, який комбінує в собі ідеї адаптивного коефіцієнта навчання та експоненціально знижуючогося середнього квадрату градієнту.

Функція втрат: Для функції втрат використовується Sparse Categorical Crossentropy. Ця функція використовується для задач класифікації, де класи не перетинаються (одне зображення належить тільки до одного класу). Функція втрат обчислює різницю між прогнозованими й фактичними мітками класів.

Метрика: У цьому випадку використовується метрика Accuracy (точність). Ця метрика вимірює відсоток правильно класифікованих зображень з усіх зображень.

Вибір цих параметрів базується на тому, що вони показали гарні результати для подібних задач класифікації зображень. Алгоритм Adam зазвичай добре справляється з навчанням нейронних мереж, функція втрат Sparse Categorical Crossentropy є стандартною для багатьох задач класифікації, а метрика Accuracy є простою у використанні та інтерпретації для оцінки результатів.

**Результати навчання:** числові результати та динаміка навчання, бажано наочно представити результати.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

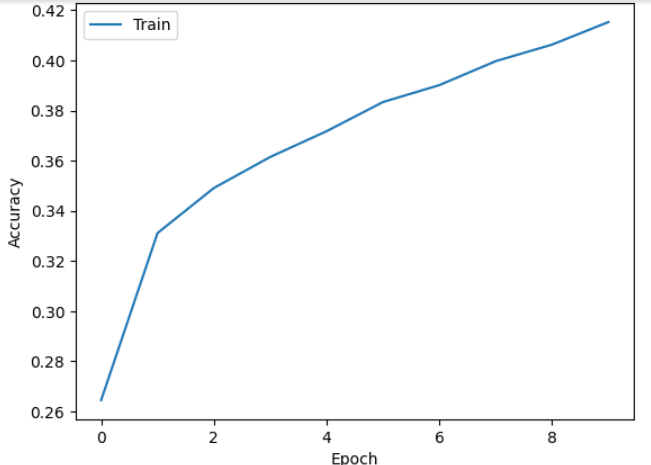


Рис. 1 – Результати навчання для 10 епох  


На графіку можна побачити зміну точності моделі під час навчання на кожній епохі.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Результати отримані при тестуванні кращого навчання моделі  


**Оцінка результатів навчання:**

* чи вдалось навчити модель? Чому так вважаєте?

Модель навчилася, тому що точність підвищувалась, а втрати зменшувалися. Також можна оцінити результати моделі на тестовому наборі, щоб побачити, як вона працює на нових даних.



* чи є перенавчання? Чому так вважаєте?

Виходить, що точність на тестовому наборі даних менше, ніж точність на навчальному наборі даних. Цей розрив між точністю навчання і точністю тесту і є перенавчання. Модель машинного навчання працює гірше з новими, раніше небаченими вхідними даними, ніж з навчальними даними.

(Звісно, якщо зменшити к-ксть епох, то перенавчання буде менше)

* чи добре підходить модель для задачі/датасету?

Модель показала покращення у точності на тренувальному наборі, що свідчить про певний успіх у навчанні. Однак, для набору даних CIFAR-10 такі значення точності є досить низькими.

Отже, можна сказати, що модель ще має простір для покращення.

* як можна покращити результати у вашому випадку?

Можливі шляхи покращення включають в себе використання більш складних моделей, налаштування гіперпараметрів, збільшення обсягу тренувальних даних або використання передньо навчених моделей для передобучення.

**Лістинг коду та результати виконання:**

import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

print(tf.\_\_version\_\_)

cifar10 = tf.keras.datasets.cifar10

(train\_images, train\_labels), (test\_images, test\_labels) = cifar10.load\_data()

class\_names = ['Airplane', 'Automobile', 'Bird', 'Cat', 'Deer',

               'Dog', 'Frog', 'Horse', 'Ship', 'Truck']

print(train\_images.shape)

print(train\_labels)

print(test\_images.shape)

plt.figure()

plt.imshow(train\_images[0])

plt.colorbar()

plt.grid(False)

plt.show()

train\_images = train\_images / 255.0

test\_images = test\_images / 255.0

plt.figure(figsize=(10,10))

for i in range(25):

    plt.subplot(5,5,i+1)

    plt.xticks([])

    plt.yticks([])

    plt.grid(False)

    plt.imshow(train\_images[i], cmap=plt.cm.binary)

    plt.xlabel(class\_names[train\_labels[i][0]])

plt.show()

model = tf.keras.Sequential([

    tf.keras.layers.Flatten(input\_shape=(32, 32, 3)),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dense(10)

])

model.compile(optimizer='adam',

              loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),

              metrics=['accuracy'])

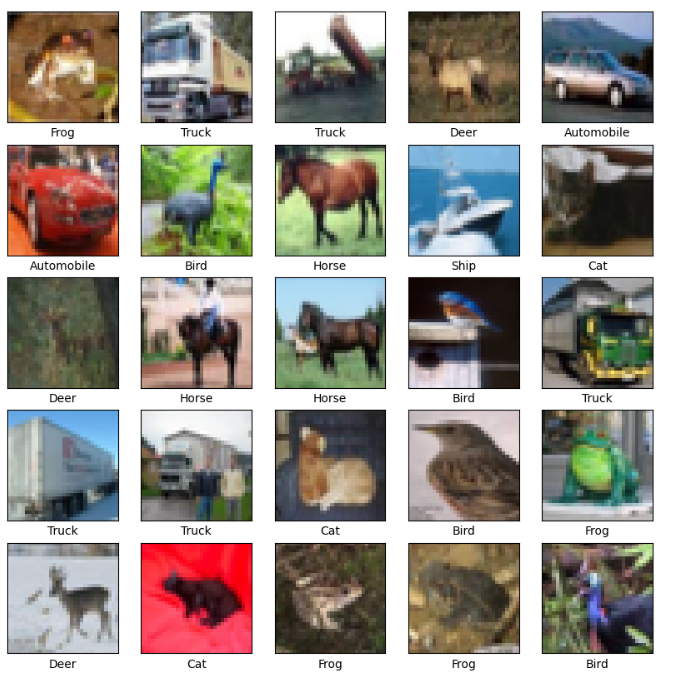
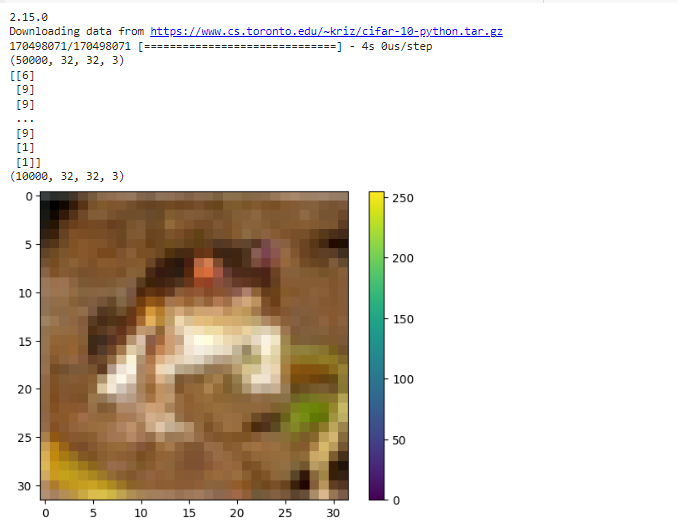
train\_images\_resized = train\_images.reshape(train\_images.shape[0], 32, 32, 3)

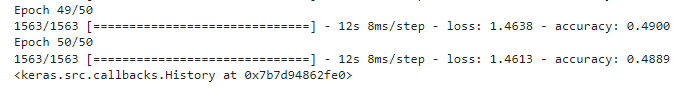
test\_images\_resized = test\_images.reshape(test\_images.shape[0], 32, 32, 3)

train\_images\_rescaled = train\_images\_resized / 255.0

test\_images\_rescaled = test\_images\_resized / 255.0

model.fit(train\_images\_rescaled, train\_labels, epochs=50)





test\_loss, test\_accuracy = model.evaluate(test\_images\_rescaled, test\_labels, verbose=2)

print(f"Точність на тестовому наборі даних: {test\_accuracy}")

probability\_model = tf.keras.Sequential([model, tf.keras.layers.Softmax()])

predictions = probability\_model.predict(test\_images\_rescaled)

print(predictions[0])

print(np.argmax(predictions[0]))

num\_images = 15

for i in range(num\_images):

    img = test\_images[i]

    actual\_label = class\_names[test\_labels[i][0]]

    predictions\_single = probability\_model.predict(img.reshape(1, 32, 32, 3))

    predicted\_class = np.argmax(predictions\_single)

    predicted\_label = class\_names[predicted\_class]

    plt.figure(figsize=(3, 3))

    plt.imshow(img)

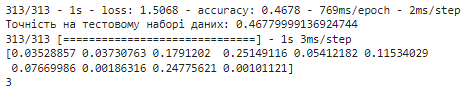
    plt.colorbar()

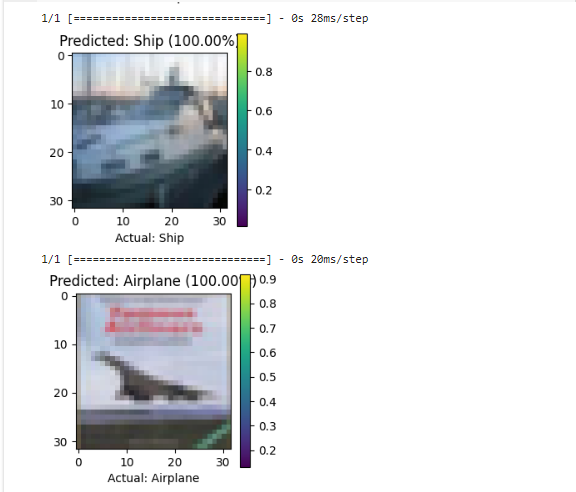
    plt.grid(False)

    plt.title(f"Predicted: {predicted\_label} ({np.max(predictions\_single)\*100:.2f}%)")

    plt.xlabel("Actual: " + actual\_label)

    plt.show()

****

****