Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Звіт до лабораторної роботи №3
«Нейронної мережі прямого розповсюдження для розпізнавання зображення»

з дисципліни

«Програмні засоби проектування та реалізації нейромережевих систем»

Виконала:
студентка групи ІМ-13
Мартинюк Марія Павлівна
номер у списку: 63

Перевірив:
Шимкович В. М.

Київ 2024

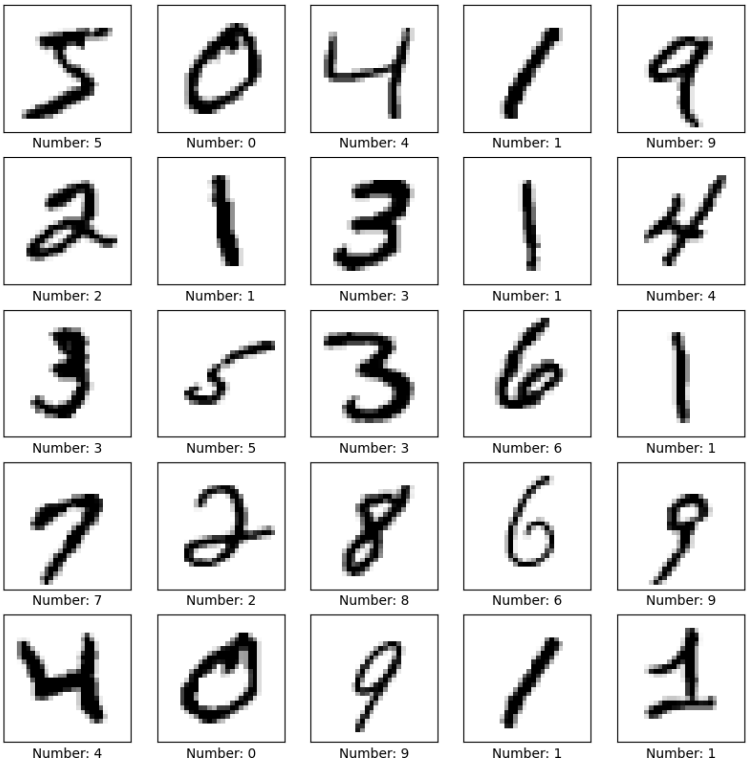
Завдання: Написати програму що реалізує нейронну мережу прямого розповсюдження для розпізнавання рукописних цифр.

Хід роботи:

Спершу для наочності результату тренування моделі було реалізовано додаткову функцію для виводу та збереження у окремий файл першочергових даних MNIST.

```
def show_numbers(samples_num, x_train, y_train):
    plt.figure(figsize=(12, 12))
    for i in range(samples_num):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(x_train[i], cmap=plt.cm.binary)
        plt.xlabel(f'Number: {y_train[i]}')
    plt.savefig('initial_data.png')
```

Результат її виконання продемонстрований нижче:



Далі було також реалізовано декілька функції відображення графіків для порівняння результатів зміни втрат та точності відносно епох навчання.

```
# function for graphs display
def loss_accuracy_graph_display(loss, val_loss, accuracy,
val_accuracy):
    plt.figure(figsize=(8, 8))
    plt.subplot(1, 2, 1)
    plt.plot(loss, label='Train Loss', color='green')
    plt.plot(val_loss, label='Validation Loss', color='red')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Model Loss')
    plt.legend()
    plt.grid(True)

    plt.subplot(1, 2, 2)
    plt.plot(accuracy, label='Accuracy', color='pink')
    plt.plot(val_accuracy, label='Validation Accuracy', color='black')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.title('Model Accuracy')
    plt.legend()
    plt.grid(True)

    plt.savefig('loss_accuracy.png')
```

Окрім цього, для відображення результатів роботи мережі на більш конкретних прикладах було реалізовано наступну функцію:

```
# function for predicted data display
def prediction_display(samples_num, x_test, y_test, prediction):
    plt.figure(figsize=(10, 10))
    for i in range(samples_num):
        plt.subplot(5, 5, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.grid(False)
        plt.imshow(x_test[i], cmap=plt.cm.binary)
```

```
        plt.title(f'\nNumber: {y_test[i]}. \nPrediction:
{np.argmax(prediction[i])} ')
    plt.savefig('numbersPredicted.png')
```

Для реалізації заданої нейронної мережі був створений відповідний клас, де відбувалось завантаження тренувальних та тестувальних даних, компіляція моделі та її навчання за допомогою fit(). Додатково було реалізовано метод evaluate_model, завдяки якому пізніше отримуються передбачення на певному наборі даних.

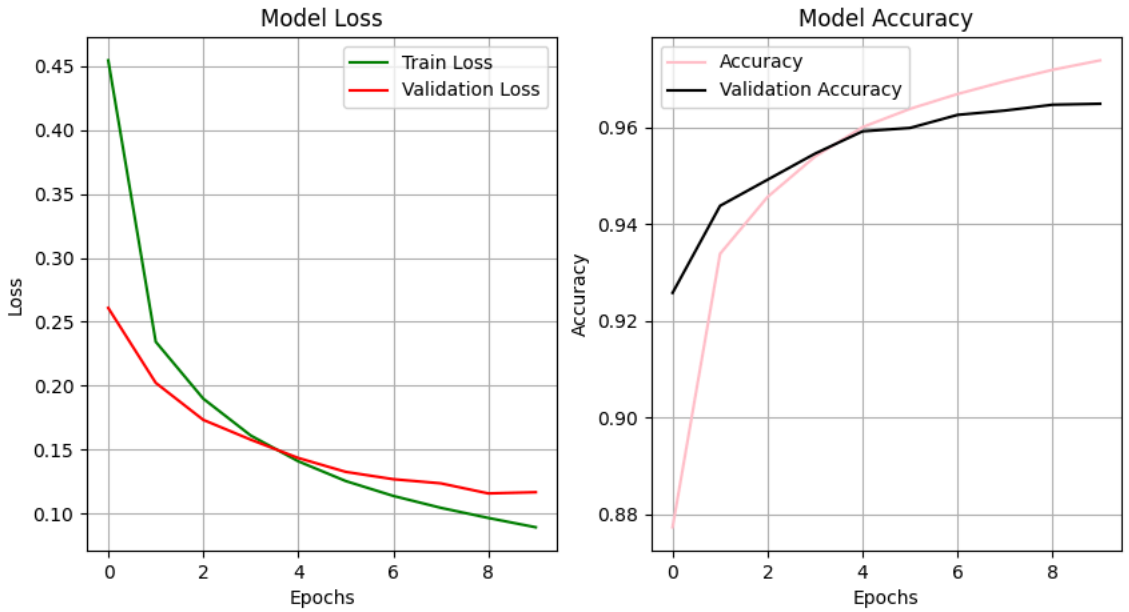
```
class ImageRecognitionNN:
    def __init__(self, batch_size, epochs, samples_num):
        self.batch_size = batch_size
        self.epochs = epochs
        self.samples_num = samples_num

        # MNIST training and testing dataset load
        (self.x_train, self.y_train), (self.x_test, self.y_test) =
mnist.load_data()

        # data normalization
        self.x_train = self.x_train / 255.0
        self.x_test = self.x_test / 255.0

        self.model = self.compile_model()

    # method for compilation
    def compile_model(self):
        model = Sequential([ Flatten(input_shape=(28, 28)), Dense(32,
activation='relu'), Dense(10, activation='softmax') ])
        model.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
        return model

    #method for training
    def train(self):
        loss = []
        accuracy = []
        val_loss = []
        val_accuracy = []
```

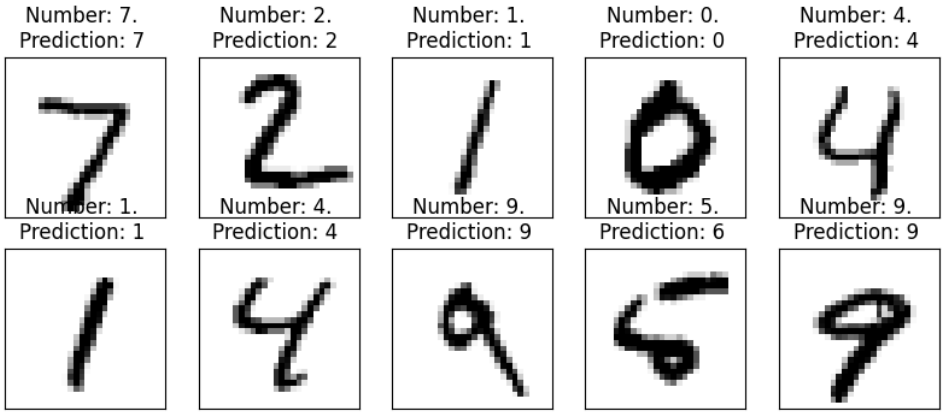
```
        for epoch in range(self.epochs):
            print(f"Epoch {epoch+1}/{self.epochs}")
            epoch_history = self.model.fit(self.x_train, self.y_train,
epochs=1, batch_size=batch_size, validation_data=(self.x_test,
self.y_test))

            loss.append(epoch_history.history['loss'][0])
            accuracy.append(epoch_history.history['accuracy'][0])
            val_loss.append(epoch_history.history['val_loss'][0])

val_accuracy.append(epoch_history.history['val_accuracy'][0])

        return loss, accuracy, val_loss, val_accuracy

    def evaluate_model(self):
        prediction = self.model.predict(self.x_test)
        return prediction
```

Наприкінці було оголошено розмір тренувального пакету, кількість епох навчання та кількість прикладів цифр для виведення. Також був створений та навчений екземпляр нейронної мережі та викликані відповідні функції для побудови графіків та відображення наборів цифр.

```
batch_size = 100
epochs = 10
samples_num = 25

neural_network = ImageRecognitionNN(batch_size, epochs, samples_num)

show_numbers(samples_num, x_train=neural_network.x_train,
y_train=neural_network.y_train)

loss, accuracy, val_loss, val_accuracy = neural_network.train()
loss_accuracy_graph_display(loss, val_loss, accuracy, val_accuracy)

prediction = neural_network.evaluate _model()
prediction_display(samples_num=10, x_test=neural_network.x_test,
y_test=neural_network.y_test, prediction=prediction)
```

Результати навчання:





Висновок:

Як результат виконання лабораторної роботи було створено нейронну мережу для розпізнавання зображень та навчено її на основі даних MNIST. Для побудови та навчання моделі використовувалися бібліотеки Keras і TensorFlow. Окрім цього для наочності було виведено результати навчання у вигляді графіків з відображення зміни втрат та точності відносно епох навчання. Додатково модель було протестовано на іншому наборі даних для перевірки якості роботи моделі в реальних умовах. За отриманими результатами можна зробити висновок, що мережа навчилась коректно класифікувати зображення.