Практичне заняття 7

Тема: Розпізнавання рукописних зображень за методами глибокого навчання Deep Learning.

Мета. Придбати практичні навички по розпізнаванню чорно-білих зображень рукописного тексту з використанням бібліотеки **Keras**; побудові нейронних мереж по розпізнавання образів за методами глибокого навчання **Deep Learning**.

Завдання 1

1. Програмними засобами побудувати модель нейронної мережі для розпізнавання зображень рукописних цифр, які влаштовані в дата-сеті бібліотеки **Keras**.
2. Побудувати архітектуру нейронної мережі для розпізнавання чорно-білих рукописних цифр та вивести:
   - з бази даних зразків рукописних цифр **MNIST** (скорочення від "***Modified National Institute of Standards and Technology***") 25 зображень;
   - сформувати архітектуру нейронної мережі по розпізнаванню рукописних зразків з **трьома прошарками**: вхідним, прихованим та вихідним;
   - вхідний прошарок **Flatten** для зразків зображень 28x28 перетворити на вектор із 784 елементів;
   - для прихованого прошарку **Dense** задати 128 нейронів, для вихідного – 10 нейронів (рис. 1), враховуючи функцію активації **bias**. Вивести структуру побудованої нейронної мережі на консоль.

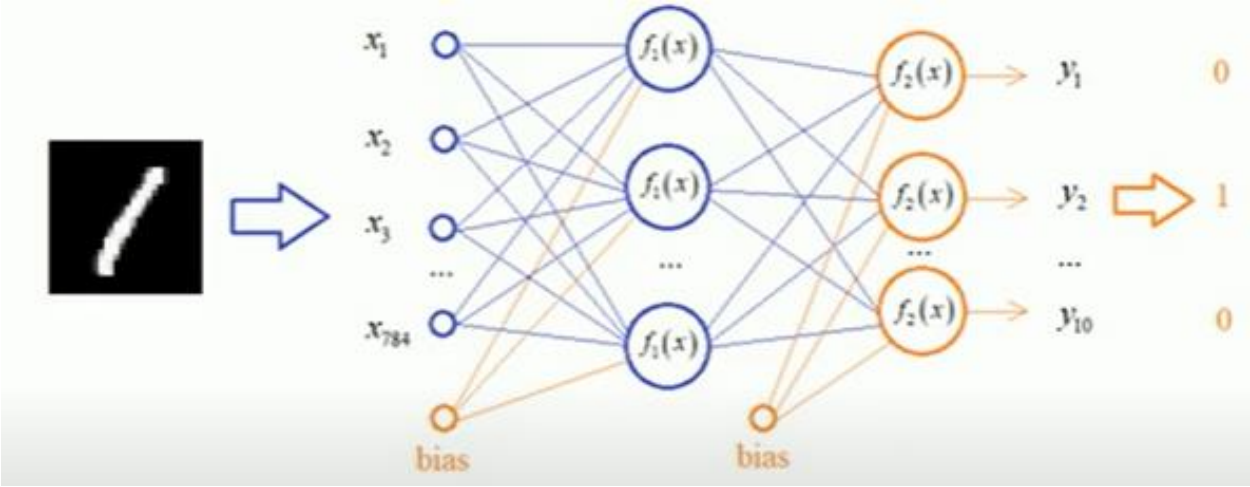

Рисунок 1. – Архітектура нейронної мережі по розпізнавання рукописних зразків.

   - провести стандартизацію (нормалізацію) вхідних даних у бінарному форматі за **10**-ю класами;
   - провести навчання нейронної мережі з функцією втрат (**loss function**) та способом оптимізації градієнтного алгоритму по класифікації за

категоріальною крос-ентропією (**categorical_crossentropy**) та активаційною функцією вихідних нейронів **softmax**, вказавши метод оптимізації **Adam** з врахуванням відповідної метрики;

- реалізувати процес навчання побудованої нейронної мережі за відповідними параметрами:
  - розмір батча (**batch_size**) 32 зразки;
  - розподіл навчальної та тестової вибірки (**validation_split**) забезпечити у співвідношенні 20% валідації.
- записати результати точності розпізнавання зразків відповідних епох в процесі навчання;
- виконати розпізнавання одного тестового зображення заданого дата-сету та вивести його на консоль;
- визначити кількість не достовірно розпізнаних зображень заданого дата-сету та вивести 5 таких зразків на консоль.

3. Побудувати архітектуру нейронної мережі для розпізнавання рукописних зразків відповідно до варіанту (табл. 1). Вивести на екран відповідні зразки. Записати результати точності навчання НМ.

Таблиця 1. Варіанти завдань для побудови архітектури нейронних мереж глибокого навчання

Параметри	**№ варіанта**								
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Вибір кількості зразків	30	35	40	45	55	60	65	70	75
Кількість нейронів прихованого прошарку	256	64	32	512	300	150	400	100	450
Кількість прихованих прошарків	2	1	2	1	2	1	2	1	2
Кількість епох	3	5	7	9	12	14	6	8	2
Розмір батча	30	25	42	15	20	5	35	28	12
Розподіл валідації	10%	12%	15%	17%	18%	22%	25%	27%	30%
Тестова перевірка цифр	5 25 7	8 12 35	14 20 39	30 20 40	25 17 50	22 11 59	33 44 55	11 22 44	70 50 45
Кількість помилково розпізнаних зразків	10	12	15	20	22	3	7	9	6

4. Зберегти файл з результатами розпізнавання зразків заданого дата-сету за алгоритмом глибокого навчання.

Завдання 2

1. Програмними засобами побудувати модель нейронної мережі для розпізнавання чорно-білих зразків дата-сету **Fashion MNIST**, влаштованого в бібліотеці ***Keras***.
2. Побудувати архітектуру нейронної мережі для розпізнавання рукописних зразків відповідно до варіанту (табл. 1). Вивести на екран відповідні зразки. Записати відповідні результати точності навчання НМ.
3. Зберегти файл з результатами розпізнавання зразків заданого дата-сету за алгоритмом глибокого навчання.

Зміст звіту

1. Титульна сторінка, тема та мета практичного заняття.
2. Опис виконання завдань по пунктам з наданням рисунків, скріншотів.
3. Тексти виконаних програм
4. Висновки.

Теоретичні відомості

Keras: бібліотека глибокого навчання на Python

Keras - бібліотека глибокого навчання, що представляє собою високорівневий ***API***, написаний на мові ***Python*** і здатний працювати з ***TensorFlow***, ***Theano***, ***CNTK***. Бібліотека ***Keras*** розроблена для реалізації швидкого навчання за методами ***Deep Learning***, яка:

- дозволяє легко та швидко створювати прототипи (завдяки зручності, модульності та масштабуванню);
- підтримує згорткові, рекурентні мережі, так і їх комбінації;
- працює як на процесорі (CPU), так і на графічному процесорі (GPU).

Keras: розпізнавання рукописних цифр

Перед нами поставлена задача – побудувати нейронну мережу, яка буде розпізнавати рукописні цифри. Скористаємось стандартним дата-сетом, який міститься БД зображень ***MNIST.***

MNIST - (***Modified National Institute of Standards and Technology***) – стандартна база даних зразків рукописного набору цифр. Бібліотека зразків ***MNIST*** постачається разом із ***Keras.*** Для доступу до дата-сету ***MNIST*** потрібно виконати наступний імпорт в ***Python***:

```
import tensorflow as tf
mnist=tf.keras.datasets.mnist
```

Після підключення бібліотек, завантажуємо стандартний дата-сет

MNIST:

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

В даному дата-сеті міститься 60 000 зображень у навчальній вибірці та 10 000 – у тестовій. В даному випадку будемо використовувати визначення:

- ***x_train*** – зображення цифр навчальної вибірки;
- ***y_train*** – вектор відповідних цифр (наприклад, якщо на і-му зображенні зображено 5, то y_train[i] = 5);
- ***x_test*** – зображення цифр тестової вибірки;
- ***y_test*** – вектор відповідних цифр для тестової вибірки.

Кожне зображення має розмір 28x28 пікселів. Для обробки зображень необхідно задати їх градацію у відтінках сірого, тобто кожен піксель має значення від 0 до 255, де 0 – чорний колір, 255 – білий (рис. 2):

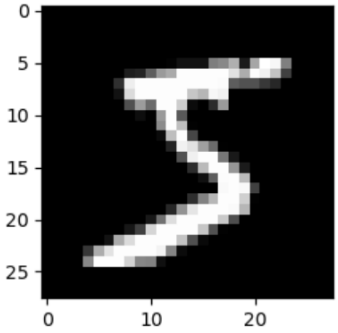

Рисунок 2. – Зразок зображення дата-сету ***MNIST***

Виведемо із дата-сету ***MNIST*** перші 25 зображень, використовуючи наступний програмний код:

```
# Відображення перших 25 зображень з навчальної вибірки MNIST
plt.figure
plt.figure(figsize=(10,5))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i], cmap=plt.cm.binary)

plt.show()
```

Як бачимо різні зображення мають довільне написанням цифр. Перед нами постає задача - навчити нейронну мережу правильно розпізнавати задані рукописні цифри.

Реалізуємо структуру нейронної мережі. Чіткої методології по моделюванню нейронної мережі немає, тоді як деяка структура НМ вибирається розробником виходячи з його уявлень по реалізації поставленого завдання. В даному випадку скористаємось архітектурою звичайної повнозв'язаної нейронної мережі, яка містить (рис. 3):

- ***28 x 28 = 784*** входів;
- ***128*** нейронів прихованого прошарку;
- ***10*** нейронів вихідного прошарку.

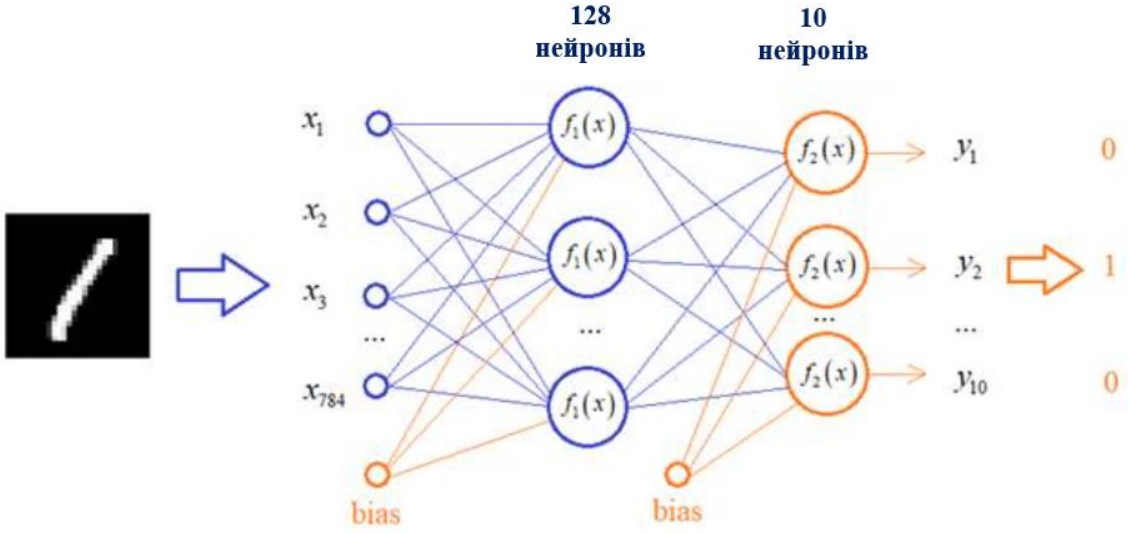

Рисунок 3. – Архітектура нейронної мережі по розпізнавання рукописних зразків.

Для прихованого шару нейронної мережі виберемо функції активації ***ReLu***, для вихідних нейронів - функції активації ***softmax***, що дозволить інтерпретувати вихідні значення в належності до відповідного класу десяткових цифр.

Реалізуємо структуру даної нейронної мережі. Перший (***вхідний***) прошарок повинен перетворювати зображення 28x28 пікселів у вектор із 784 елементів. Для такої операції в ***Keras*** можна створити прошарок спеціального виду – ***Flatten*** (рис. 4).

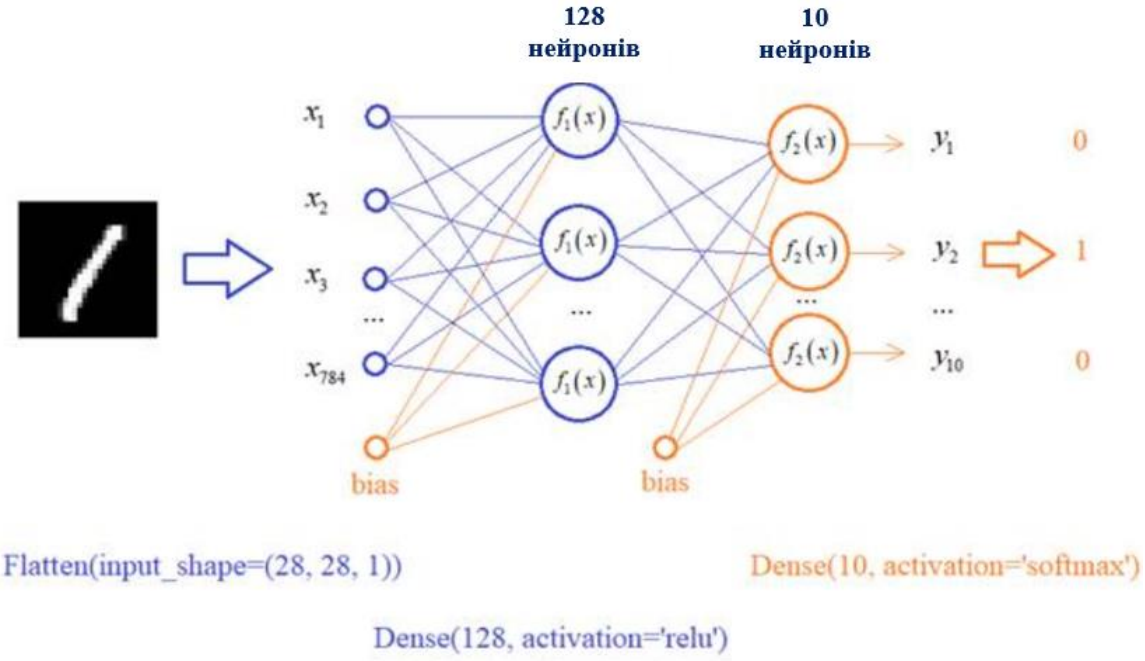

Рисунок 4. – Формування прошарків нейронної мережі

Наступний ***прихований*** прошарок реалізуємо за допомогою класу Dense, який повноцінно пов’яже 784 входи із 128 нейронами даного прошарку. Вихідний прошарок містить 10 нейронів, які пов'язані із 128 нейронами прихованого прошарку. Загальну модель нейронної мережі Keras можна записати як:

```
models=tf.keras.models
layers=tf.keras.layers
Flatten=tf.keras.layers.Flatten
Dense=tf.keras.layers.Dense

model = keras.Sequential([
    Flatten(input_shape=(28, 28, 1)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
# виведення структури Нейронної мережі на екран
print(model.summary())
```

На наступному етапі бажано вхідні значення вектора ***x*** стандартизувати таким чином, щоб всі значення знаходились у діапазоні від 0 до 1. Виконаємо дане перетворення наступним програмним кодом:

```
x_train = x_train / 255
x_test = x_test / 255
```

Після виконання, кожне значення тензору ***x_train*** та ***x_test*** буде розділене на максимальне значення 255, яке може прийматись. На виході отримаємо величини в діапазоні від 0 до 1.

Надалі потрібно підготувати відповідний формат вихідних значень. Для кожного зображення рукописної цифри вектор ***y_train*** буде відображати клас відповідного числа (рис. 5).

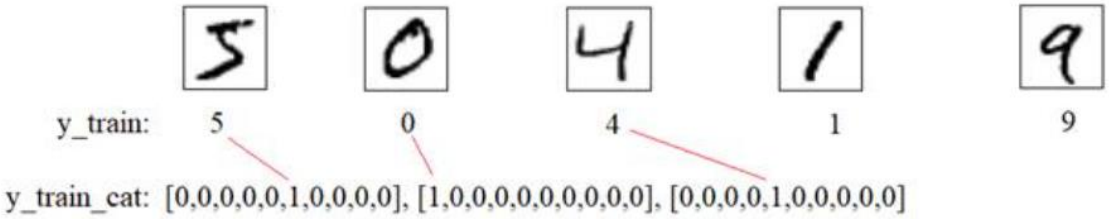

Рисунок 5. – Нормалізація вихідних значень в нейронній мережі

На вихідному прошарку потрібно побудувати вектор з 10-ма виходами, в якому кожен нейрон буде відповідати певній рукописній цифрі від 0 до 9. Для такого перетворення в Keras існує функція, яку застосуємо для навчальної та тестової вибірки:

```
y_train_cat = keras.utils.to_categorical(y_train, 10)
```

```
y_test_cat = keras.utils.to_categorical(y_test, 10)
```

Для отримання наборів векторів ***y_train_cat*** та ***y_test_cat*** за заданим форматом, необхідно вказати другий параметр функції 10, що свідчить про розмірність кожного вектору.

Навчання нейронної мережі по розпізнаванню рукописних цифр

Виберемо функцію втрат (***loss function***) та спосіб оптимізації алгоритм ***градієнтного зсуву***. Важливо для задач класифікації, при моделюванні, першочергово застосовується категоріальна функція крос-ентропії ***categorical_crossentropy*** та активаційна функція вихідних нейронів ***softmax***.

Функцію активації можна записати, з врахуванням критерію якості, наступним чином:

```
model.compile(optimizer='adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
```

В даному програмному коді вказана функція оптимізації ***Adam*** із врахуванням метрики. Метрика, при вирішенні задач класифікації, свідчить про відсоток правильно розпізнаних цифр у нейронній мережі. При побудові алгоритму навчання, необхідно врахувати ***metrics*** мінімізацію похибки у відсотках, що забезпечить зменшення втрат - категоріальної функції крос-ентропії.

На даному етапі нейронна мережа підготовлена до процесу навчання, достатньо запустити на виконання наступний програмний код:

```
model.fit(x_train, y_train_cat, batch_size=32,
epochs=10, validation_split=0.2)
```

де, вказані наступні параметри:

- ***batch_size = 32*** –розмір батча (32 картинки), після яких буде виконуватися коригування вагових коефіцієнтів;
- ***validation_split = 0,2*** – розбиття навчальної вибірки на навчальну та тестову. Значення 0,2 визначає, що на кожній епосі 20% випадкових зразків навчальної вибірки поміщаються у валідаційну вибірку (допустиме значення тестової вибірки від 10% до 30%).

Перевіримо роботу нейронної мережі на тестовій вибірці наступним програмним кодом:

```
#перевірка критерію якості навчання мережі на тестовій вибірці
model.evaluate(x_test, y_test_cat)
```

Метод ***evaluate*** надає можливість визначити тестову вибірку за один прямий прохід та обчислити величину критерію якості та відповідної метрики.

Перевірка розпізнаних рукописних цифр

Виконаємо розпізнавання будь-якого тестового зразка нейронною мережею, виконавши наступний програмний код:

```
n = 1
x = np.expand_dims(x_test[n], axis=0)
res = model.predict(x)
print(res)
```

Спочатку виділяємо із тензора ***n-й*** зразок і пропускаємо його по мережі, використовуючи метод ***predict***. На виході отримаємо 10 значень, за якими визначаємо клас вказаної рукописної цифри. В результаті отримаємо вектор вихідних значень:

[[0 0 1 0 0 0 0 0 0]]

Максимальне значення відповідає потрібному класу. В даному випадку – це число ***1***, яке є третім виходом, що свідчить про розпізнану цифру ***2***. Виведемо номер максимального числа із даного вектору, скориставшись влаштованою функцією ***argmax*** модуля ***numpy***:

```
print(np.argmax(res))
```

В результаті отримаємо індекс під номером ***2*** та відобразимо на екрані визначений тестовий зразок:

```
plt.imshow(x_test[n], cmap=plt.cm.binary)
plt.show()
```

Отримаємо зображення рукописної цифри ***2***, що, в результаті, свідчить про правильність функціонування побудованої нейронної мережі.

В будь-якому випадку нейронна мережа не може 100% розпізнати всі зразки, які містяться в дата сеті. Визначимо не вірно розпізнані зразки нейронною мережею. Пропустимо через НМ всю тестову вибірку та вектори вихідних значень перетворимо в числа від 0 до 9 наступним кодом:

```
pred = model.predict(x_test)
pred = np.argmax(pred, axis=1)

print(pred.shape)

print(pred[:20])
print(y_test[:20])
```

Сформуємо маску, яка буде містити ***True*** для вірних варіантів розпізнавання і ***False*** – для невірних. За допомогою даної маски виділимо з тестової вибірки всі неправильні результати:

```
mask = pred == y_test
```

```
print(mask[:10])
x_false = x_test[~mask]
y_false = x_test[~mask]

print(x_false.shape)
```

Виведемо перші 5 неправильно розпізнаних нейронною мережею зображень з них на екран:

```
for i in range( 5):
  print("Значеннння мережі: "+str(y_test[i]))
  plt.imshow(x_false[i], cmap=plt.cm.binary)
  plt.show()
```

В даній моделі нейронної мережі можна проекспериментувати з різною кількістю нейронів прихованого шару та проаналізувати результати класифікації. Структуру мережі можна змінювати – задавати в прихованому прошарку не 128 нейронів, як вказано у прикладі, а взяти 100 нейронів або 50; побудувати мережу з двома прихованими шарами і т. д. Важливо, при цьому, спостерігати за величиною функції втрат – чим менше її значення, тим точніше буде нейронна мережа розпізнавати рукописні цифри.

Контрольні питання

1. Модель штучного нейрона.
2. Основні типи активаційних функцій.
3. Спосіб обчислення коефіцієнтів w_{ij} в алгоритмах навчання.
4. Постановка задачі навчання ШНМ.
5. Параметри алгоритму навчання.
6. Підготовка даних для навчання ШНМ.
7. Визначення навчальної та тестової вибірки в НМ
8. Основні типів функцій втрат.