

Звіт до комп'ютерного практикуму №1.

Нейронні мережі прямого поширення

ПІБ: Грисюк Михайло Олександрович

Група: ІК-21мп

Мета роботи: ознайомитись з принципами побудови, навчання та використання мереж прямого поширення, дослідити вплив параметрів моделі, алгоритму навчання та даних на результати роботи.

Завдання: Для задачі класифікації (регресії) на основі типового датасету створити нейронну мережу прямого поширення (багатошаровий персептрон). Навчити її, перевірити результат на тестовій вибірці, оцінити результати, наявність недонавчання або перенавчання. Провести дослідження впливу параметрів, відповідно варіанту, на результати роботи мережі.

Номер варіанту: 5

Завдання для варіанту: Для задачі класифікації на основі датасету **Iris** створити нейронну мережу прямого поширення. Навчити її, перевірити результат на тестовій вибірці, оцінити результати, наявність недонавчання або перенавчання. Провести дослідження **впливу кількості шарів та кількості нейронів у шарах** на результати роботи мережі.

Засоби виконання практикуму: Дану комп'ютерну практику було виконано в середовищі VSCode зі спеціально встановленими розширеннями Jupiter та іншими. Використовувалась мова програмування Python та фреймворком TensorFlow, це є самі популярні інструменти для створення та навчання нейронних мереж.

Набір даних (датасет): Іриси Фішера складаються з даних про 150 екземплярів ірис, по 50 екземплярів із трьох видів:

- Ірис щетинистий (Iris-setosa).
- Ірис віргінський (Iris-virginica).
- Ірис різнокольниковий (Iris-versicolor).

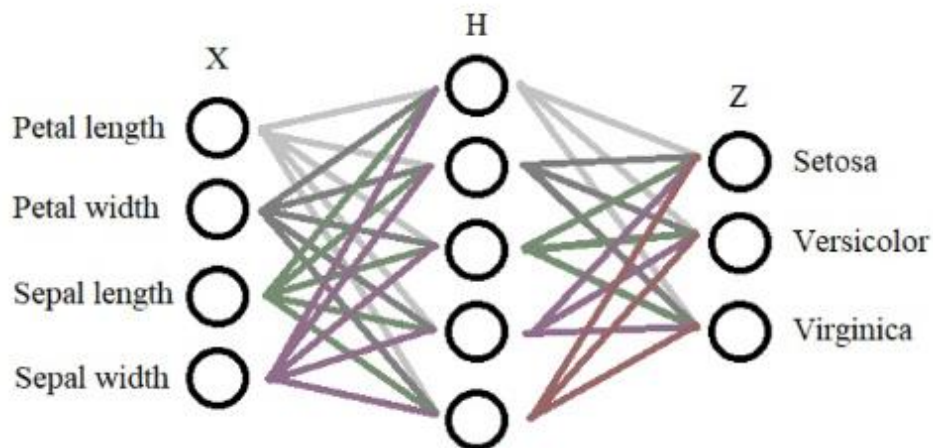
Для кожного екземпляра виміряно чотири характеристики (в сантиметрах):

- Довжина чашолистика (sepal_length).
- Ширина чашолистика (sepal_width).
- Довжина пелюстки (petal_length).
- Ширина пелюстки (petal_width).

Попередня обробка даних: попередньої обробки даних немає.

Модель машинного навчання:

На вході у нас є 4 класи (характеристики) - X, також нам знадобиться внутрішній шар - H, в ньому буде 10 нейронів, далі на виході ми маємо 3 класи, які залежать від властивості кольорів - Z. Виходить така конструкція мережі:



Код моделі навчання:

```
def simple_mlp_model(num_classes):  
    input_ = tf.keras.layers.Input(shape=(4,))  
    x = tf.keras.layers.Dense(10, activation='relu')(input_)  
    output_ = tf.keras.layers.Dense(num_classes, activation='softmax')(x)  
    return tf.keras.models.Model(input_, output_, name='Classifier')
```

```
num_classes = metadata.features['label'].num_classes  
model = simple_mlp_model(num_classes)  
model.summary()
```

✓ 0.3s

Model: "Classifier"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 4)]	0
dense (Dense)	(None, 10)	50
dense_1 (Dense)	(None, 3)	33

Total params: 83

Trainable params: 83

Non-trainable params: 0

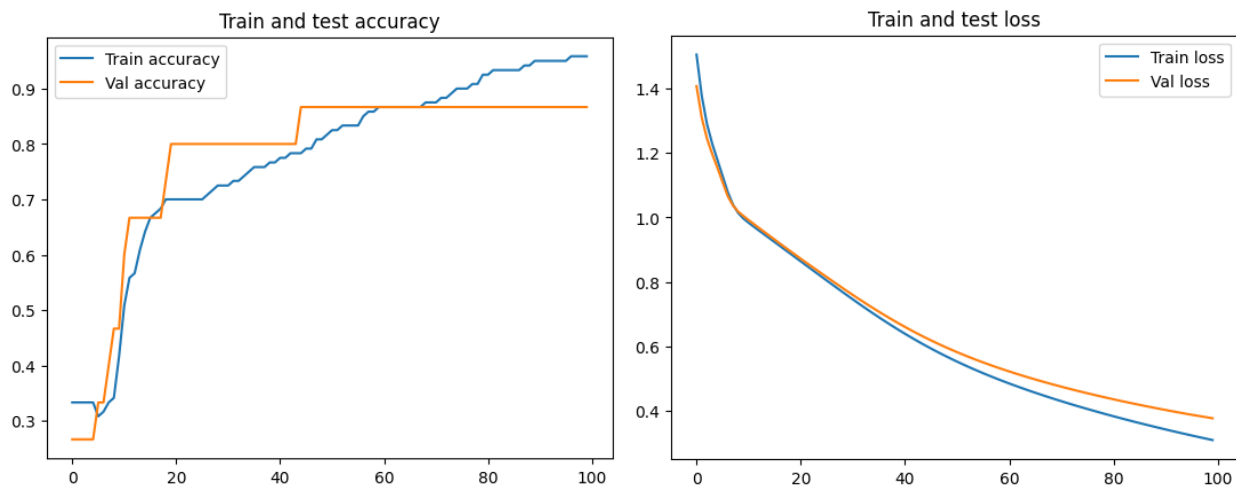
Константи:

```
BATCH_SIZE = 10  
EPOCHS = 100  
LEARNING_RATE = 0.001
```

✓ 0.8s

Навчання моделі: Ми використовували алгоритм навчання мультикласової класифікації, адже нам потрібно знайти до якого класу відноситься даний набір значень, а для функції втрат тут доречно використати **Categorical Crossentropy** - типовий вибір і він вимагає softmax ФА, що підходить для нас. Softmax ми використовували для отримання ймовірностей які вказують до якого класу найбільш відноситься даний набір значень.

Результати навчання:



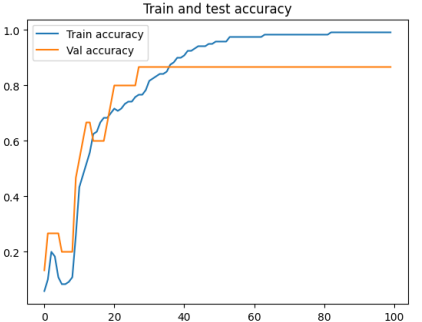
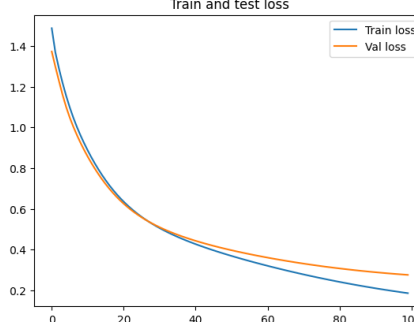
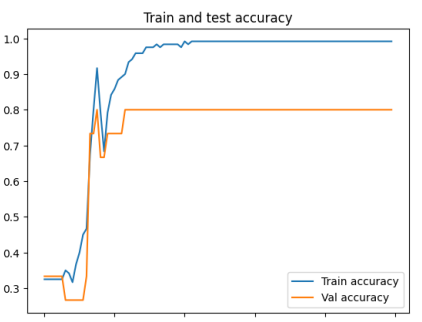
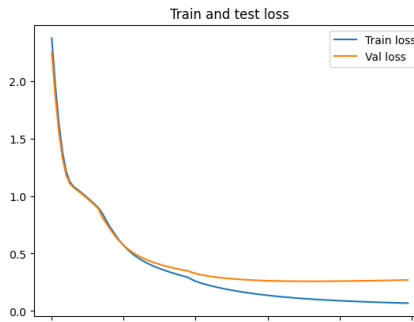
```
2/2 [=====] - 0s 6ms/step - loss: 0.3360 - accuracy: 0.9333
Test loss : 0.3360339403152466
Test Accuracy : 0.9333333373069763
```

Оцінка результатів навчання: наша модель навчається, але у нас недонавчання, адже при тестуванні у нас високий результат функції низька точність тестування. Щоб покращити результат нам можна збільшити кількість шарів або збільшити кількість шарів, або те і те.

Задача дослідження: Вплив кількості шарів та кількості нейронів у шарах

Результати експериментального дослідження:

Варіант покращення результату	Точність	Втрати	Оцінка результату
Додавання шару з 10 нейронами	<p>The graph shows Train accuracy (blue) reaching ~0.98 and Val accuracy (orange) reaching ~0.87 by epoch 100.</p>	<p>The graph shows Train loss (blue) and Val loss (orange) both decreasing to ~0.07 by epoch 100.</p>	<pre>train_loss: 0.0761 train_accuracy: 0.983 val_loss: 0.2740 val_accuracy: 0.8667 test_loss: 0.074 test_accuracy : 1.0</pre> <p>На останніх епохах (після 80) відбувається перенавчання. Покращення результату.</p>

Збільшення нейронів до 20	 	<p> train_loss: 0.1858 train_accuracy: 0.9917 val_loss: 0.2756 val_accuracy: 0.8667 test_loss : 0.097 test_accuracy : 1.0 </p> <p>Велике покращення результату</p>
Збільшення нейронів до 20 та додавання шару з 10 нейронами	 	<p> train_loss: 0.0684 train_accuracy: 0.9917 val_loss: 0.2700 val_accuracy: 0.800 test_loss : 0.061 test_accuracy : 1.0 </p> <p>Перенавчання після 50 епохи. Покращення результату.</p>

Висновки за результатами дослідження: Згідно нашими результатами можна зробити висновок, що при збільшенні шарів та нейронів у шарі результати точності та втрат покращуються але може відбутися перенавчання, що може бути проблемою для розпізнавання нових даних. Краще за все буде використання одного шару 20 нейронами – воно показало найкращий результат.