Міністерство освіти і науки

України Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

ЗВІТ

лабораторної роботи №4

з курсу «Програмні засоби проєктування і реалізації неромережевих систем»

Тема: «Згорткові нейронні мережі»

Перевірив:

Шимкович В. М.

Виконав:

Студент Гр. ІП-01 Шпилька В.С.

Київ 2023

Завдання: Написати програму що реалізує згорткову нейронну мережу AlexNet для розпізнавання об’єктів з датасету ImageNet.

Модель:

```python
import tensorflow as tf

def AlexNet():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4), activation='relu'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        tf.keras.layers.Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        tf.keras.layers.Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1), activation='relu', padding='same'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1), activation='relu', padding='same'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(4096, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

    return model
```

```
    model = tf.keras.models.load_model("Artifacts/Models/v1/Model/tf")
    model.summary()
[14] ✓ 1.2s

... Output exceeds the size limit. Open the full output data in a text editor
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 55, 55, 96)        34944

 batch_normalization (BatchN  (None, 55, 55, 96)       384
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 27, 27, 96)       0
 )

 conv2d_1 (Conv2D)           (None, 27, 27, 256)       614656

 batch_normalization_1 (Batc  (None, 27, 27, 256)      1024
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 13, 13, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 13, 13, 384)       885120

 batch_normalization_2 (Batc  (None, 13, 13, 384)      1536
 hNormalization)

...
Total params: 59,900,682
Trainable params: 59,897,930
Non-trainable params: 2,752
_________________________________________________________________
```

Попередня обробка та побудова пайплайну для даних:

Preprocessing:

1. Стандартизація картинки
2. Зміна розмірності до 227, 227, 3

Data pipeline:

1. Створення датасету з картинок
2. Випадкове перемішування датасету(після кожної ітерації)
3. Використання функції preprocessing
4. Розподілення на батчі

Функція тренування:

```
def train(version,
          validation_num = settings.VALIDATION_NUM,
          save_folder = settings.SAVE_FOLDER,
          epochs = settings.EPOCHS,
          batch_size = settings.BATCH_SIZE,
          lr = settings.DEFAULT_LR):

    #load data
    (train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.cifar10.load_data()

    validation_images, validation_labels = train_images[:validation_num], train_labels[:validation_num]
    train_images, train_labels = train_images[validation_num:], train_labels[validation_num:]

    #creating tf.data
    train_ds = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
    validation_ds = tf.data.Dataset.from_tensor_slices((validation_images, validation_labels))


    #creating data pipelines
    train_ds = create_data_pipeline(train_ds, batch_size)
    validation_ds = create_data_pipeline(validation_ds, batch_size)

    model = AlexNet()

    #values for schedules
    initial_learning_rate = 10**(-2)
    final_learning_rate = 10**(-4)
    learning_rate_decay_factor = (final_learning_rate / initial_learning_rate)**(1/epochs)
    steps_per_epoch = len(train_ds)

    learning_rate = lr
    if(lr == -1):
        learning_rate = tf.keras.optimizers.schedules.ExponentialDecay(
                        initial_learning_rate=initial_learning_rate,
                        decay_steps=steps_per_epoch,
                        decay_rate=learning_rate_decay_factor
                    )

    model.compile(loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'],
                  optimizer=tf.keras.optimizers.SGD(learning_rate=learning_rate))
```

```
path_to_save = save_folder + '/' + version + '/'

checkpoint_dir = path_to_save + "Checkpoints/"
checkpoint_path = checkpoint_dir + "cp-{epoch:04d}.ckpt"
checkpoint = ModelCheckpoint(filepath=checkpoint_path,
                             monitor='val_loss',
                             verbose=1,
                             save_weights_only = True,
                             mode='auto')

tf_path = path_to_save + "Model/tf"
fullModelSave = ModelCheckpoint(filepath=tf_path,
                             monitor='val_loss',
                             verbose=1,
                             save_best_only=True,
                             mode='auto')


log_dir = path_to_save + "Logs/"
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir)

callbacks_list = [checkpoint, tensorboard_callback, fullModelSave]

model.fit(
    train_ds,
    epochs = epochs,
    shuffle=False,
    validation_data = validation_ds,
    callbacks = callbacks_list,
    verbose = 1)
```

1. Скачування і підготовка датасету та дата пайплайну
2. Створення моделі
3. Компіляція моделі
4. Створення колбеків для збереження логів, чекпоінтів та моделі
5. Тренування моделі

Гіперпараметри:

```
RANDOM_SEED = 42
BATCH_SIZE = 32
SAVE_FOLDER = './Artifacts/Models'
EPOCHS = 50
DEFAULT_LR = -1
VALIDATION_NUM = 5000
LABELS = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
```

(learning rate = -1 означає використання schedules.ExponentialDecay)

В результаті тренування нейроної мережі було отримано точність у 82,26 відсотків:

```
sys.path.append(os.getcwd() + '/Preprocessing')
from AlexNetPreprocessor import create_test_data_pipeline

(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.cifar10.load_data()
test_ds = tf.data.Dataset.from_tensor_slices((test_images, test_labels))
test_ds = create_test_data_pipeline(test_ds, 32)
```

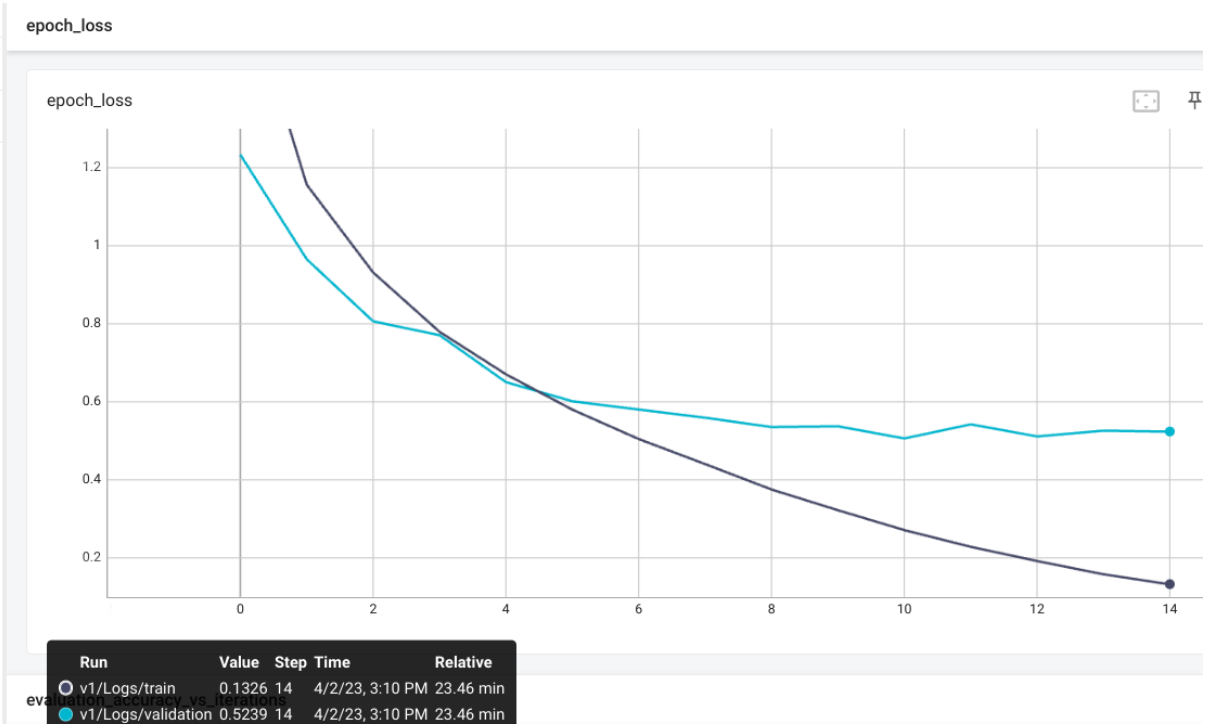
[3] ✓ 0.7s

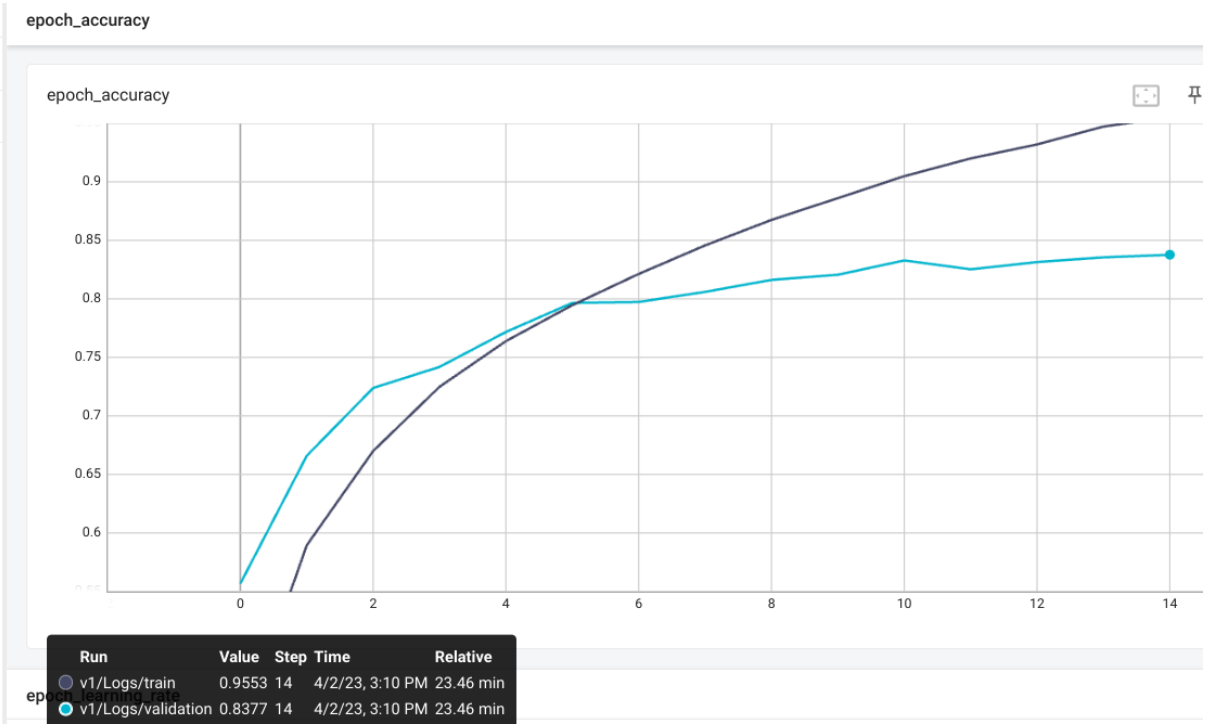
```
model.evaluate(test_ds)
```

[4] ✓ 2m 26.6s

```
313/313 [==============================] - 147s 468ms/step - loss: 0.5456 - accuracy: 0.8226

[0.5456198453903198, 0.8226000070571899]
```

Зміна loss і точності від епохи:



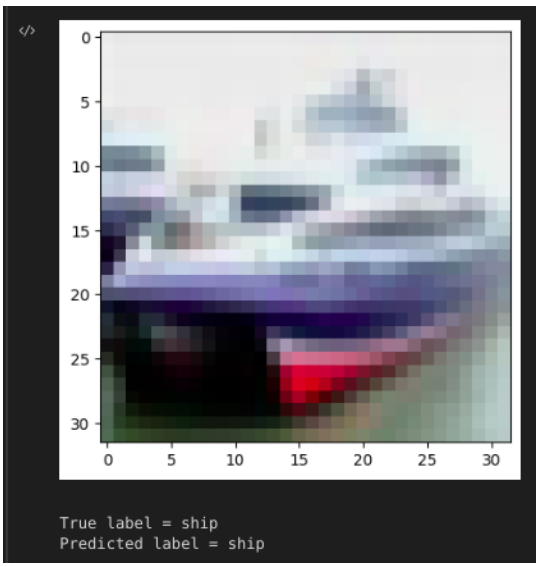


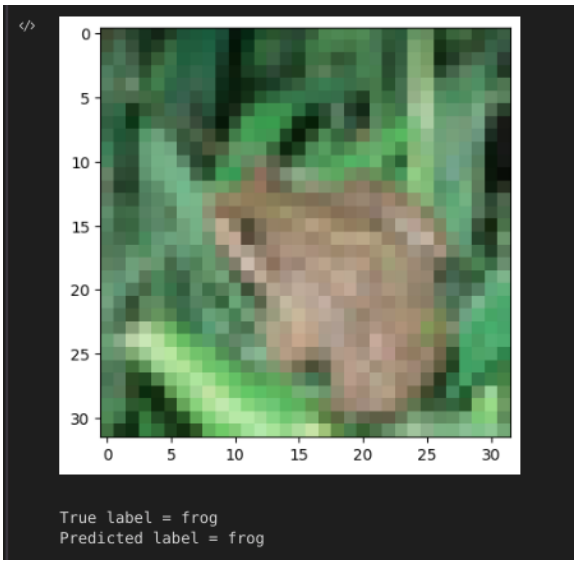
Приклад роботи:

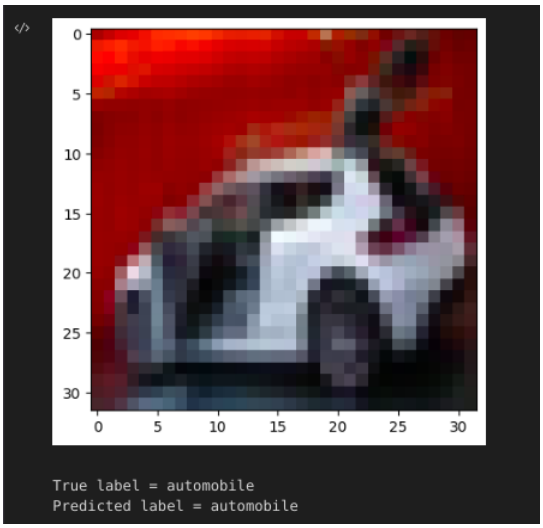
```
sys.path.append(os.getcwd() + '/config')
from settings import LABELS

plt.figure(figsize=(3, 3))
for images, true_classes in test_ds.take(1):
    predictions = model.predict(images)
    true_classes = true_classes.numpy().squeeze(axis=1)
    classes = np.argmax(predictions, axis=1)
    for index, img in enumerate(images):
        img = img.numpy()
        imgplot = plt.imshow(test_images[index])
        plt.show()
        print('True label =', LABELS[true_classes[index]])
        print('Predicted label =', LABELS[classes[index]])
        if(index > 10):
            break
```

[6] ✓ 2.6s

```
1/1 [==============================] - 0s 470ms/step
```







Висновок:

В результаті виконання лабораторної роботи було побудовано згорткову нейрону мережу AlexNet для класифікації об’єктів з CIFAR10 датасету. Всього нейрона мережа має 59млн параметрів. Для даного датасету дана нейрона мережа показала гарні результати, а саме точність в 82,26 відсотків.