МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

з лабораторної роботи №4

з навчальної дисципліни «Проектування та реалізація програмних систем з нейронними мережами»

Виконав:
студент групи ІП-15
Мєшков Андрій Ігорович

Перевірив:
Шимкович В.М.

Київ 2024

ЛАБОРАТОРНА РОБОТА №4

Тема: Згорткові нейронні мережі.

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

Хід роботи

- Завантажуємо датасет

```
(train_ds, test_ds), ds_info = tfds.load('imagenette/160px-v2', split=['train',
'validation'], as_supervised=True, with_info=True)
```

- Передобробка даних

```
train_ds = train_ds.map(lambda img, label: (tf.image.resize(img, (128, 128)), label))
test_ds = test_ds.map(lambda img, label: (tf.image.resize(img, (128, 128)), label))
```

- Завантаження класів

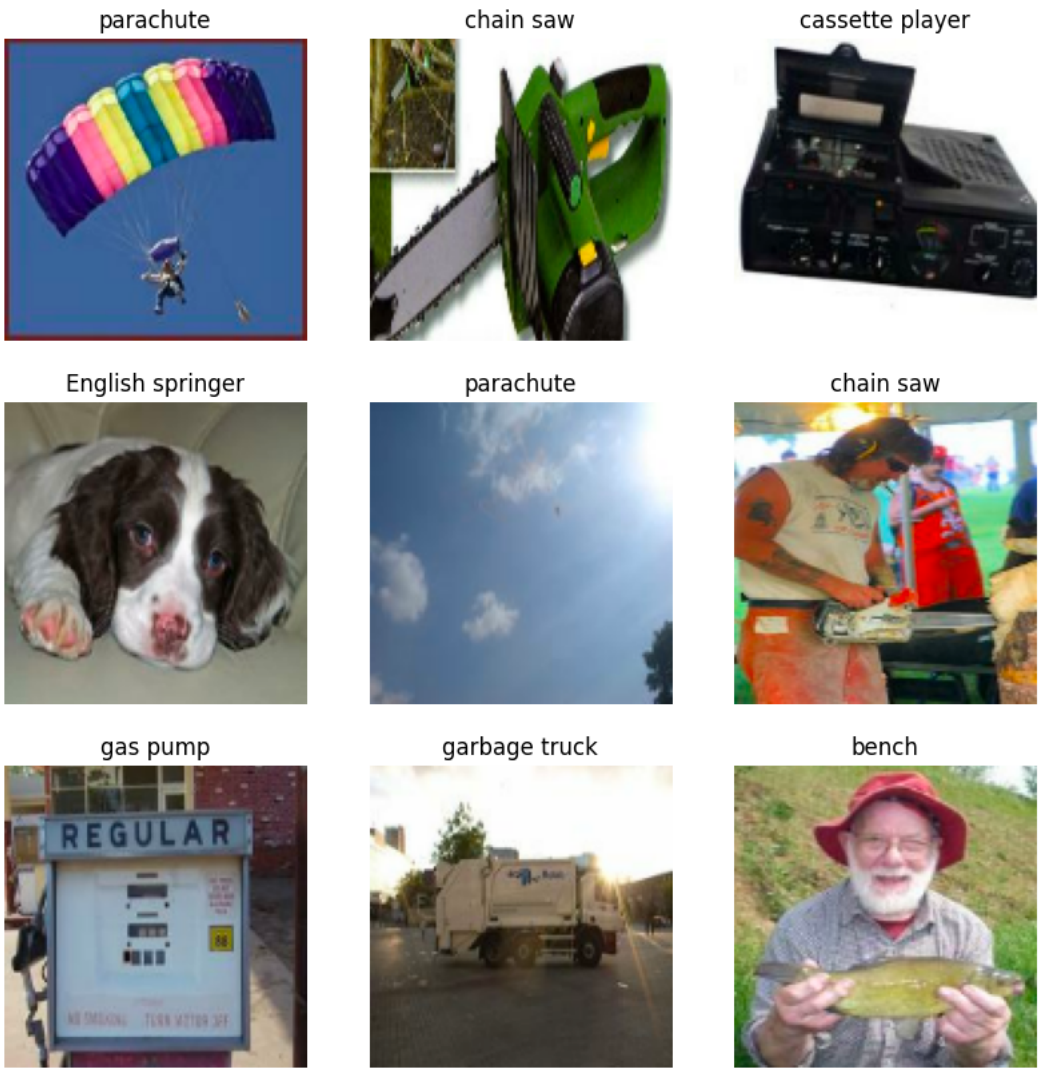
```
img_size, img_labels = ds_info.features['label'].num_classes,
ds_info.features['label'].int2str
label_names = [img_labels(id) for id in range(img_size)]

print(f'Усього {img_size} класів')
for i, label in enumerate(label_names):
    print(i, '-', label)
```

```
Усього 10 класів
0 - n01440764
1 - n02102040
2 - n02979186
3 - n03000684
4 - n03028079
5 - n03394916
6 - n03417042
7 - n03425413
8 - n03445777
9 - n03888257
```

```
class_names = ['bench', 'English springer', 'cassette player', 'chain saw', 'church',
'French horn', 'garbage truck', 'gas pump', 'golf ball', 'parachute']


plt.figure(figsize=(10, 10))
for i, example in enumerate (test_ds.take(9)):
    image, label = example
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(image. numpy().astype("uint8"))
    plt.title(class_names [label.numpy()])
    plt.axis ("off")
plt.show()
```



- Підготовка даних

```
X_train, y_train = list(map(lambda img: img[0], train_ds)), list(map(lambda img: img[1],
train_ds))
X_test, y_test = list(map(lambda img: img[0], test_ds)), list (map (lambda img: img[1],
test_ds))
y_train, y_test = to_categorical(y_train, img_size), to_categorical(y_test, img_size)


train_size, test_size = ds_info.splits['train'].num_examples,
ds_info.splits['validation'].num_examples
print(f'Навчальна вибірка: {train_size}, Тестувальна вибірка {test_size}')
```

```
Навчальна вибірка: 9469, Тестувальна вибірка 3925
```

```
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=40,
height_shift_range=0.2)
test_datagen = ImageDataGenerator(rescale=1./255)
train_ds = NumpyArrayIterator(x=np.array(X_train), y=np.array(y_train),
image_data_generator=train_datagen, batch_size=16)
```

```
test_ds = NumpyArrayIterator(x=np.array(X_test), y=np.array(y_test),
image_data_generator=test_datagen, batch_size=32)
```

- Определення моделі AlexNet

```
model = Sequential([
    Conv2D(96, kernel_size=(11,11), strides=4, activation='relu',
input_shape=(128,128,3), padding='valid'),
    BatchNormalization(),
    MaxPool2D(pool_size=(3,3), strides=(2,2)),

    Conv2D(256, kernel_size=(5,5), padding='same', activation='relu', strides=(1,1)),
    BatchNormalization(),
    MaxPool2D(pool_size=(3,3), strides=(2,2)),

    Conv2D(384, kernel_size=(3,3), padding='same', activation='relu', strides=(1,1)),
    BatchNormalization(),

    Conv2D(192, kernel_size=(3,3), padding='same', activation='relu', strides=(1,1)),
    BatchNormalization(),

    Conv2D(256, kernel_size=(3,3), padding='same', activation='relu', strides=(1,1)),
    BatchNormalization(),
    MaxPool2D(pool_size=(3,3), strides=(2,2)),
    Flatten(),

    Dense(2048, activation='relu'),
    Dropout(0.5),

    Dense(2048, activation='relu'),
    Dropout(0.5),

    Dense(10, activation='softmax')
])
```

- Компіляція моделі

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

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 30, 30, 96)	34,944
batch_normalization_10 (BatchNormalization)	(None, 30, 30, 96)	384
max_pooling2d_6 (MaxPooling2D)	(None, 14, 14, 96)	0
conv2d_11 (Conv2D)	(None, 14, 14, 256)	614,656
batch_normalization_11 (BatchNormalization)	(None, 14, 14, 256)	1,024
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_12 (Conv2D)	(None, 6, 6, 384)	885,120
batch_normalization_12 (BatchNormalization)	(None, 6, 6, 384)	1,536
conv2d_13 (Conv2D)	(None, 6, 6, 192)	663,744
batch_normalization_13 (BatchNormalization)	(None, 6, 6, 192)	768
conv2d_14 (Conv2D)	(None, 6, 6, 256)	442,624
batch_normalization_14 (BatchNormalization)	(None, 6, 6, 256)	1,024
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten_2 (Flatten)	(None, 1024)	0
dense_6 (Dense)	(None, 2048)	2,099,200
dropout_4 (Dropout)	(None, 2048)	0
dense_7 (Dense)	(None, 2048)	4,196,352
dropout_5 (Dropout)	(None, 2048)	0
dense_8 (Dense)	(None, 10)	20,490

Total params: 8,961,866 (34.19 MB)

Trainable params: 8,959,498 (34.18 MB)

Non-trainable params: 2,368 (9.25 KB)

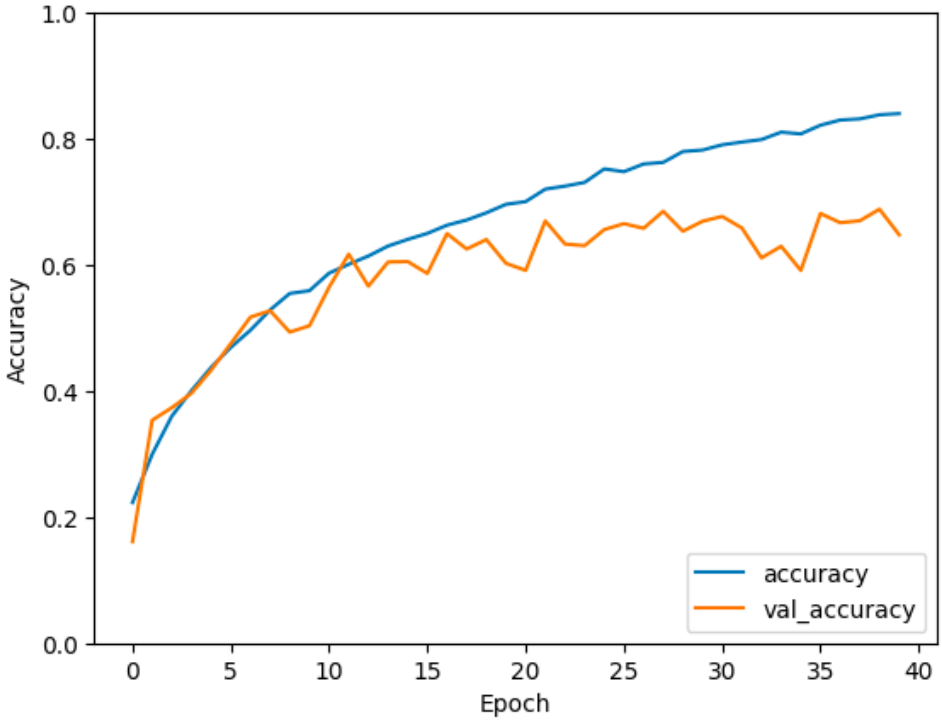
- Навчання моделі

```
history = model.fit(train_ds, validation_data=test_ds, epochs=40)
```

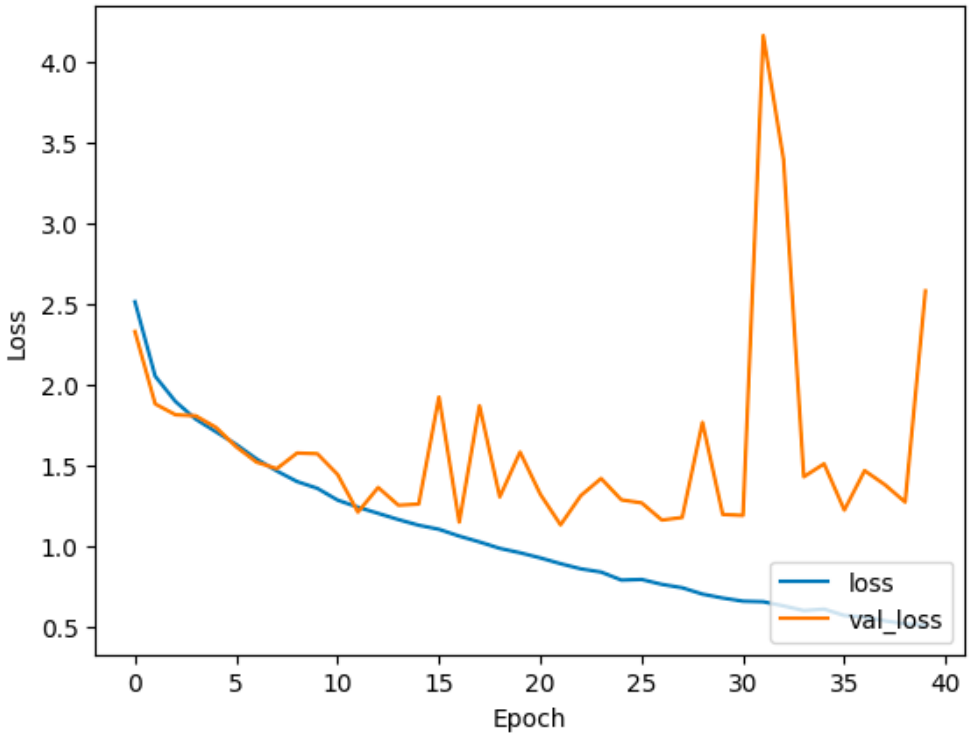
```
Epoch 1/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 188s 306ms/step - accuracy: 0.1881 - loss: 3.2831 - val_accuracy: 0.1610 - val_loss: 2.3261
Epoch 2/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 164s 275ms/step - accuracy: 0.2783 - loss: 2.0996 - val_accuracy: 0.3536 - val_loss: 1.8773
Epoch 3/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 156s 263ms/step - accuracy: 0.3469 - loss: 1.9253 - val_accuracy: 0.3735 - val_loss: 1.8101
Epoch 4/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 156s 262ms/step - accuracy: 0.3896 - loss: 1.7946 - val_accuracy: 0.3964 - val_loss: 1.8035
Epoch 5/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 152s 256ms/step - accuracy: 0.4311 - loss: 1.7306 - val_accuracy: 0.4331 - val_loss: 1.7316
Epoch 6/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 151s 255ms/step - accuracy: 0.4619 - loss: 1.6304 - val_accuracy: 0.4752 - val_loss: 1.6110
Epoch 7/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 151s 255ms/step - accuracy: 0.4834 - loss: 1.5543 - val_accuracy: 0.5172 - val_loss: 1.5164
Epoch 8/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 151s 254ms/step - accuracy: 0.5272 - loss: 1.4566 - val_accuracy: 0.5274 - val_loss: 1.4759
Epoch 9/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 156s 262ms/step - accuracy: 0.5601 - loss: 1.3826 - val_accuracy: 0.4935 - val_loss: 1.5729
Epoch 10/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 152s 256ms/step - accuracy: 0.5593 - loss: 1.3533 - val_accuracy: 0.5034 - val_loss: 1.5684
Epoch 11/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 153s 258ms/step - accuracy: 0.5898 - loss: 1.2758 - val_accuracy: 0.5646 - val_loss: 1.4397
Epoch 12/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 154s 259ms/step - accuracy: 0.6048 - loss: 1.2217 - val_accuracy: 0.6171 - val_loss: 1.2061
Epoch 13/40
...
Epoch 39/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 160s 270ms/step - accuracy: 0.8340 - loss: 0.5104 - val_accuracy: 0.6884 - val_loss: 1.2687
Epoch 40/40
592/592 ━━━━━━━━━━━━━━━━━━━━ 161s 271ms/step - accuracy: 0.8399 - loss: 0.5124 - val_accuracy: 0.6479 - val_loss: 2.5782
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

- Вивчення результатів

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```




```
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='lower right')
plt.show()
```



- Приклад

```
def plot_predictions(img_id, predictions, ds):
    img, label = ds[img_id]
    img_pred = predictions[img_id]
    plt.figure(figsize=(6, 3))
    plot_pred_img(img_pred, img, label)


def plot_pred_img(img_pred, img, img_label):
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(img[0], cmap=plt.cm.binary)
    pred_label = np.argmax(img_pred)
    real_label = np.argmax(img_label)
    color = 'green' if pred_label == real_label else 'red'

    plt.xlabel(f"{class_names[real_label]} ({class_names[pred_label]})", color=color)


predictions = model.predict(test_ds)
plot_predictions(149, predictions, test_ds)
```

Висновок:

Під час виконання даної лабораторної роботи було досліджено структуру та принцип роботи згорткових нейронних мереж, а саме було написано програму яка реалізує нейронну мережу AlexNet для розпізнавання зображень з датасету ImageNet. Мережу було успішно навчено розпізнавати зображеня також її було протестовано та перевірено на тестових даних.