МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №2
з дисципліни “ Технології машинного навчання”
на тему
“ Класифікація зображень засобами TensorFlow ”

Виконав:
студент
групи КН-419
Адаменко Д.С.

Прийняла:
асистентка
кафедри ІСМ
Захарія Л.М.

Львів-2019

Мета роботи: розробити скрипт моделі нейронної мережі для класифікації зображень архітектурного оздоблення вікна будинку історичної забудови міста Львова.

Рис. 1. Тип вікна «centre bor»

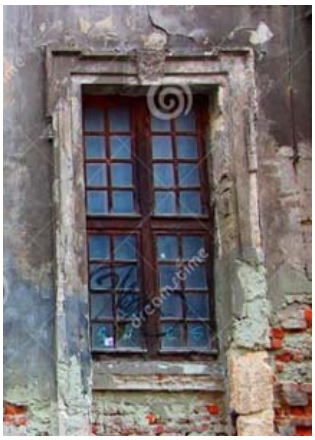

Рис. 2. Тип вікна «georgion»

Рис. 3. Тип вікна «window gril»

Рис. 4. Тип вікна «arched»

Рис. 5. Тип вікна «egress window»

Код програми:

```
from __future__ import absolute_import, division, print_function, unicode_li
terals
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import time
import glob
import cv2
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

def plot_image(i, predictions_array, true_label, img):
  predictions_array, true_label, img = predictions_array, true_label[i], img
[i]
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  plt.imshow(img, cmap=plt.cm.binary)

  predicted_label = np.argmax(predictions_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(predictions_array),
                                class_names[true_label]),
                                color=color)

def plot_value_array(i, predictions_array, true_label):
  predictions_array, true_label = predictions_array, true_label[i]
  plt.grid(False)
  plt.xticks(range(10))
  plt.yticks([])
  thisplot = plt.bar(range(10), predictions_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(predictions_array)

  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')

img_height = 300;
img_width = 300;
class_names = ['centre_bor', 'window_gril', 'georgion', 'arched', 'egress_wi
ndow']
train_images = []
```

```
train_labels = []
for i in range(5):
  for filename in sorted(glob.glob(f'/content/drive/My Drive/Denys/
window{i+1}/*.jpg')):
      img = Image.open(filename)
      train_images.append(np.asarray(img))
      train_labels.append(i)

plt.figure(figsize=(8, 40))
for i in range(100):
    plt.subplot(20,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
plt.show()

train_images = np.array(train_images)[:, :, :, 0]
train_images = train_images / 255.0

plt.figure()
plt.imshow(train_images[0])
plt.colorbar()
plt.grid(False)
plt.show()

# налаштування моделі
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(img_height, img_width)),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(10, activation='softmax')
])

# компілювання
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# тренування
start_time = time.time()
model.fit(np.array(train_images), np.array(train_labels), epochs=10)
print("Training execution: %s s" % (time.time() - start_time))
# оцінювання точності
test_loss, test_acc = model.evaluate(np.array(train_images),  np.array(train
_labels), verbose=2)
print('\nTest accuracy:', test_acc)
test_images=[]
true_labels = [4, 3, 1, 0, 1, 3, 4, 2, 0, 1]
# дані для прогнозу
for filename in sorted(glob.glob(f'/content/drive/My Drive/Denys/
for_testing/*.jpg')):
    img = Image.open(filename)
    test_images.append(np.asarray(img))
```

```
test_images = np.array(test_images)[:, :, :, 0]
test_images = test_images / 255.0
print(test_images.shape)

# передбачення
start_time = time.time()
predictions = model.predict(test_images)
print("Prediction execution: %s s" % (time.time() - start_time))

num_rows = 5
num_cols = 2
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i in range(num_images):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(i, predictions[i], true_labels, test_images)
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(i, predictions[i], true_labels)
plt.tight_layout()
plt.show()
```

Результату виконання:

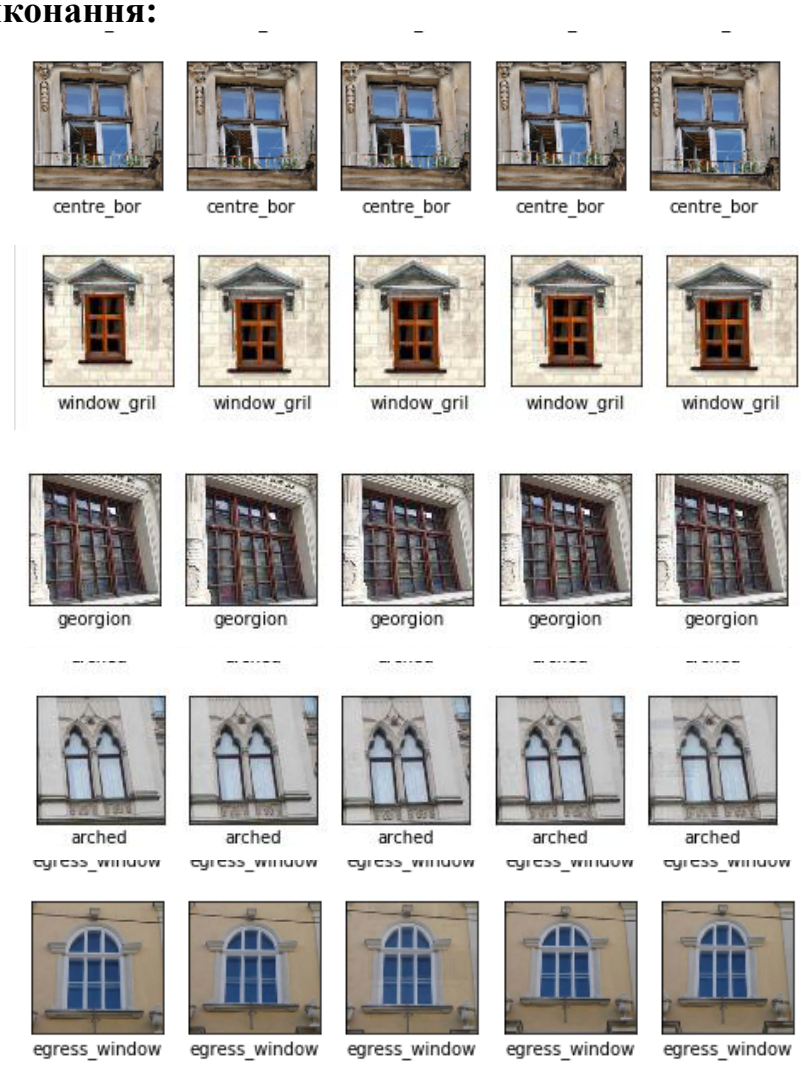

Рис. 1. Види вікон для навчання

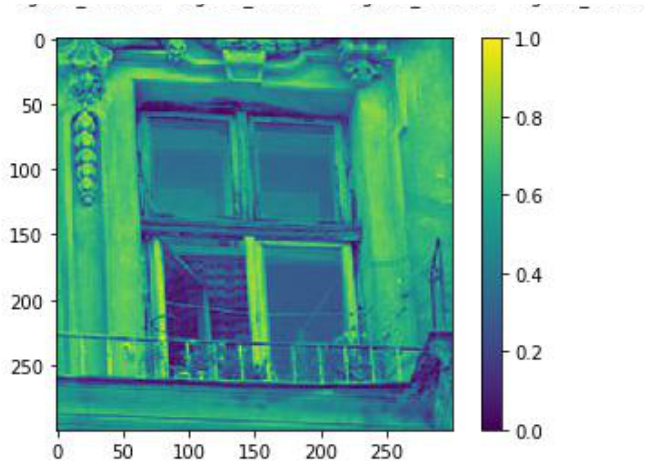

Рис. 2. Результат попереднього опрацювання даних

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 298, 298, 300)     8400
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 149, 149, 300)     0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 147, 147, 600)     1620600
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 73, 73, 600)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 71, 71, 600)       3240600
_________________________________________________________________
flatten_2 (Flatten)          (None, 3024600)           0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               387148928
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 392,019,818
Trainable params: 392,019,818
Non-trainable params: 0
_________________________________________________________________
(10, 300, 300)
Train on 100 samples, validate on 10 samples
Epoch 1/100
100/100 [==============================] - 1s 14ms/sample - loss: 30.9752 - accuracy: 0.2700 - val_loss: 75.7752 - val_accuracy: 0.1000
Epoch 2/100
100/100 [==============================] - 0s 4ms/sample - loss: 67.3374 - accuracy: 0.2500 - val_loss: 40.9053 - val_accuracy: 0.6000
```

...

```
Epoch 45/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.7010e-06 - accuracy: 1.0000 - val_loss: 8.0494 - val_accuracy: 0.8000
Epoch 46/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.6259e-06 - accuracy: 1.0000 - val_loss: 8.0437 - val_accuracy: 0.8000
Epoch 47/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.5115e-06 - accuracy: 1.0000 - val_loss: 8.0386 - val_accuracy: 0.8000
Epoch 48/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.4447e-06 - accuracy: 1.0000 - val_loss: 8.0331 - val_accuracy: 0.8000
Epoch 49/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.3446e-06 - accuracy: 1.0000 - val_loss: 8.0283 - val_accuracy: 0.8000
Epoch 50/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.2826e-06 - accuracy: 1.0000 - val_loss: 8.0231 - val_accuracy: 0.8000
Epoch 51/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.1944e-06 - accuracy: 1.0000 - val_loss: 8.0185 - val_accuracy: 0.8000
Epoch 52/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.1575e-06 - accuracy: 1.0000 - val_loss: 8.0132 - val_accuracy: 0.8000
Epoch 53/100
100/100 [==============================] - 0s 4ms/sample - loss: 2.0705e-06 - accuracy: 1.0000 - val_loss: 8.0086 - val_accuracy: 0.8000
Epoch 54/100
100/100 [==============================] - 0s 5ms/sample - loss: 2.0204e-06 - accuracy: 1.0000 - val_loss: 8.0037 - val_accuracy: 0.8000
Epoch 55/100
100/100 [==============================] - 0s 5ms/sample - loss: 1.9334e-06 - accuracy: 1.0000 - val_loss: 7.9997 - val_accuracy: 0.8000
Epoch 56/100
100/100 [==============================] - 0s 5ms/sample - loss: 1.8845e-06 - accuracy: 1.0000 - val_loss: 7.9956 - val_accuracy: 0.8000
Epoch 57/100
100/100 [==============================] - 0s 5ms/sample - loss: 1.8559e-06 - accuracy: 1.0000 - val_loss: 7.9910 - val_accuracy: 0.8000
Epoch 58/100
100/100 [==============================] - 0s 5ms/sample - loss: 1.7880e-06 - accuracy: 1.0000 - val_loss: 7.9868 - val_accuracy: 0.8000
```

Рис. 4. Результат точності та втрат

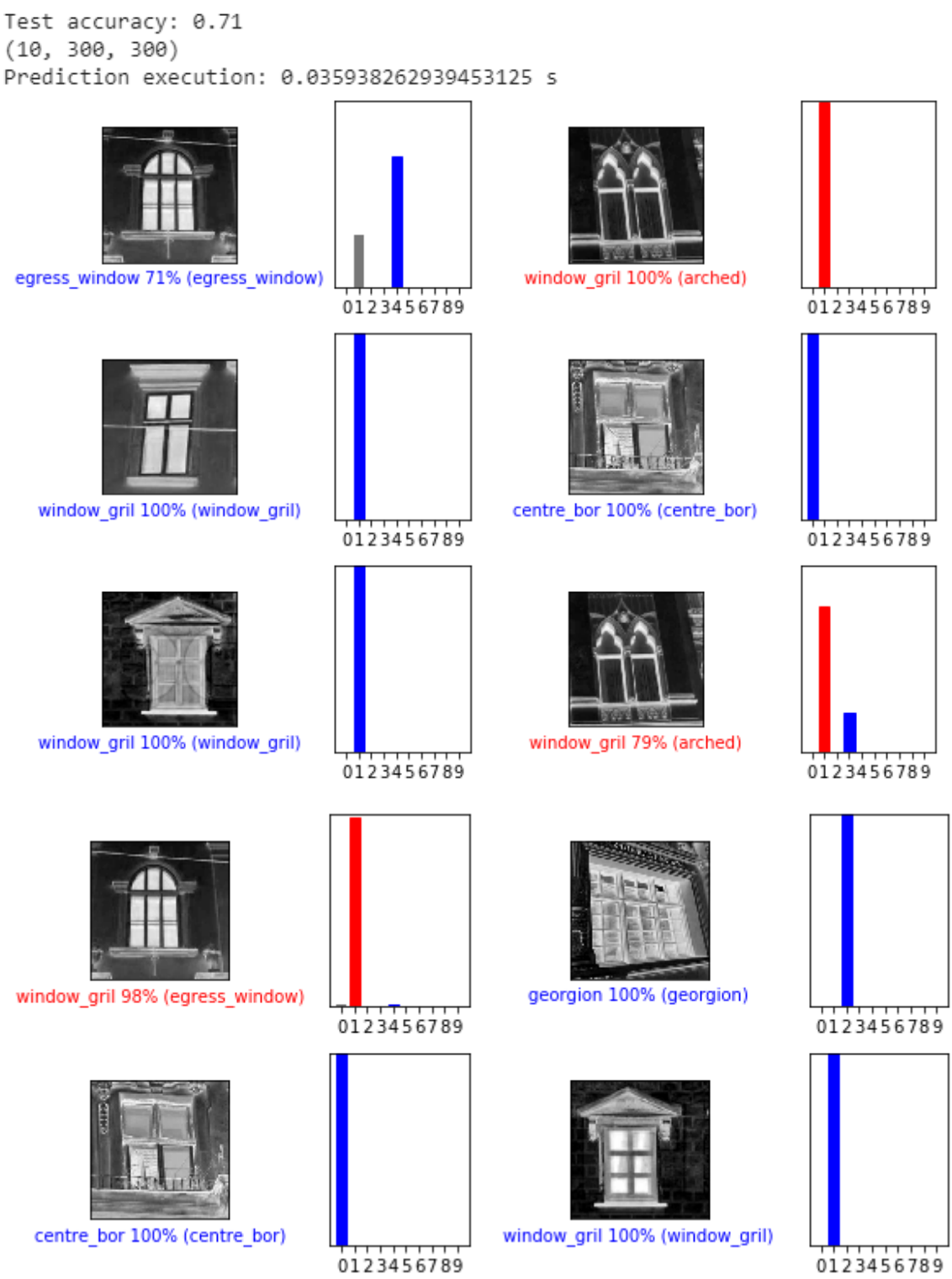

Рис. 6. Результат передбачування

Висновок: під час виконання лабораторної роботи було розроблено скрипт моделі нейронної мережі для класифікації зображень архітектурного оздоблення вікна будинку історичної забудови міста Львова.