Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем та управління

Лабораторна робота №1
З дисципліни: «Технології глибокого навчання»

Студента групи КНШІ-41
Крушельницький С. М.

м. Тернопіль – 2025

Тема: Використання згорткових нейронних мереж для класифікації зображень

Мета: Навчитись розробляти архітектури згорткових нейронних мереж для класифікації зображень

Хід роботи:

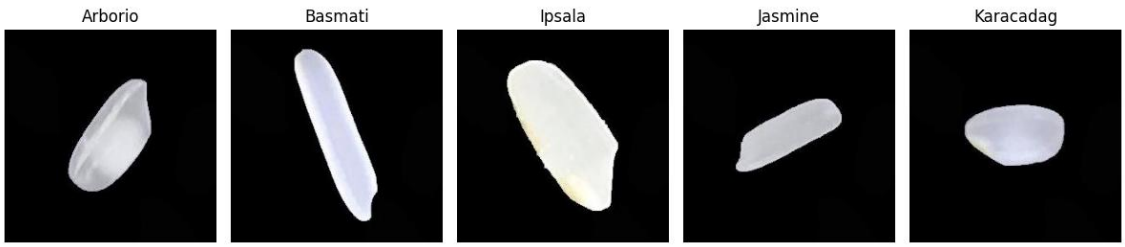
1. Код програми:

```python
import os
import random
import numpy as np
import tensorflow as tf
from tensorflow.keras import Input, Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Dropout, Dense, Flatte
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import plotly.express as px
import splitfolders
import cv2
import seaborn as sns


# Перевіряємо доступні GPU
gpus = tf.config.list_physical_devices('GPU')
print("Available GPUs:", gpus)

# Вибираємо потрібний GPU і дозволяємо динамічне виділення пам'яті
tf.config.set_visible_devices(gpus[1], 'GPU')  # або 0
tf.config.experimental.set_memory_growth(gpus[1], True)

print("Using GPU:", gpus[1])
```

```python
SEED = 123
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

print("TensorFlow:", tf.__version__)
print("GPU available:", tf.config.list_physical_devices('GP
U'))
rootPath = '../data/Rice_Image_Dataset'
out_dir = "../data/rice_imgs"
```

```python
class_names = sorted([d for d in os.listdir(rootPath) if os.path.isdir(os.path.join(rootPath,
sizes = [len(os.listdir(os.path.join(rootPath, name))) for name in class_names]
print("Класи:", class_names)
print("К-сть зображень по класах:", sizes)

fig = px.pie(
    names=class_names,
    values=sizes,
    width=500,
    title='Розподіл класів (Rice Image Dataset)',
    hole=0.4
)
fig.update_layout({'title': {'x': 0.5}})
fig.show()
```

```python
def load_random_img(dir_path, labels, n_cols=5):
    plt.figure(figsize=(12, 6))
    for i, label in enumerate(labels[:n_cols], 1):
        file = random.choice(os.listdir(os.path.join(dir_path, label)))
        image_path = os.path.join(dir_path, label, file)
        img = cv2.imread(image_path)[:, :, ::-1]  # BGR->RGB
        plt.subplot(1, n_cols, i)
        plt.title(label)
        plt.imshow(img)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

load_random_img(rootPath, class_names)
```




```python
if not os.path.exists(out_dir) or not os.path.isdir(os.path.join(out_dir, "train")):
    splitfolders.ratio(rootPath, output=out_dir, seed=SEED, ratio=(.7, .15, .15))

batch_size = 128
img_height, img_width = 256, 256
input_shape = (img_height, img_width, 3)
AUTOTUNE = tf.data.AUTOTUNE


from tensorflow.keras import backend as K
K.set_image_data_format('channels_last')
print("Image data format:", K.image_data_format())  # має бути 'channels_last'
```

```python
def prepare_dataset(subset):
    ds = tf.keras.utils.image_dataset_from_directory(
        os.path.join(out_dir, subset),
        labels='inferred',
        label_mode='categorical',
        image_size=(img_height, img_width),
        batch_size=batch_size,
        shuffle=(subset=='train'),
        seed=SEED
    )
    normalization_layer = tf.keras.layers.Rescaling(1./255)
    ds = ds.map(lambda x, y: (normalization_layer(x), y), num_parallel_calls=AUTOTUNE)
    ds = ds.prefetch(AUTOTUNE)
    return ds

train_ds = prepare_dataset('train')
val_ds = prepare_dataset('val')
test_ds = prepare_dataset('test')
```

```python
def build_cnn(input_shape=(256, 256, 3), n_classes=5):
    model = Sequential([
         # очікує зображення 256x256x3
        Input(shape=input_shape),

        # --- Перший блок ---
        Conv2D(32, 3, activation='relu', padding='same'),  # Згортка 3x3, 32 фільтри
        MaxPooling2D(2),  # Зменшення розмірності вдвічі
        BatchNormalization(),  # Нормалізація для стабілізації навчання

        # --- Другий блок ---
        Conv2D(64, 3, activation='relu', padding='same'), # Згортка 3x3, 64 фільтри
        MaxPooling2D(2),
        BatchNormalization(),

        # --- Третій блок ---
        Conv2D(128, 3, activation='relu', padding='same'), # Згортка 3x3, 128 фільтри
        MaxPooling2D(2),
        Dropout(0.3),  # dropout щоб не мати overfitting

        # --- Перетворення в вектор ---
        Flatten(),  # Згортковий вихід перетворюється у одномірний вектор

        # --- Fully connected layers ---
        Dense(256, activation='relu'),
        Dropout(0.4),  # хелпує уникнути overfitting на Dense шарах
        Dense(n_classes, activation='softmax')  # Вихідний шар для класифікації n_classes
    ])
    return model

n_classes = train_ds.element_spec[1].shape[-1]
CNN = build_cnn(input_shape=(img_height, img_width, 3), n_classes=n_classes)
CNN.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
CNN.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
batch_normalization (BatchNormalization)	(None, 128, 128, 32)	128
conv2d_1 (Conv2D)	(None, 128, 128, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 64, 64, 64)	256
conv2d_2 (Conv2D)	(None, 64, 64, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 128)	0
dropout (Dropout)	(None, 32, 32, 128)	0
flatten (Flatten)	(None, 131072)	0
dense (Dense)	(None, 256)	33,554,688
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 5)	1,285

Total params: 33,649,605 (128.36 MB)

Trainable params: 33,649,413 (128.36 MB)

Non-trainable params: 192 (768.00 B)

```python
from tensorflow.keras.utils import plot_model

plot_model(
    CNN,
    to_file='cnn_model.png',
    show_shapes=True,         # показує розміри тензорів після кожного шару
    show_layer_names=True,    # показує назви шарів
    rankdir='TB',             # напрямок Top-Bottom (можна 'LR' для Left-Right)
    expand_nested=False
)
```

```python
# Callbacks
ckpt_path = "best_cnn.keras"
callbacks = [
    EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True),
    ModelCheckpoint(ckpt_path, monitor='val_accuracy', save_best_only=True)
]

# Training
EPOCHS = 11
history = CNN.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    callbacks=callbacks,
    verbose=1
)
```

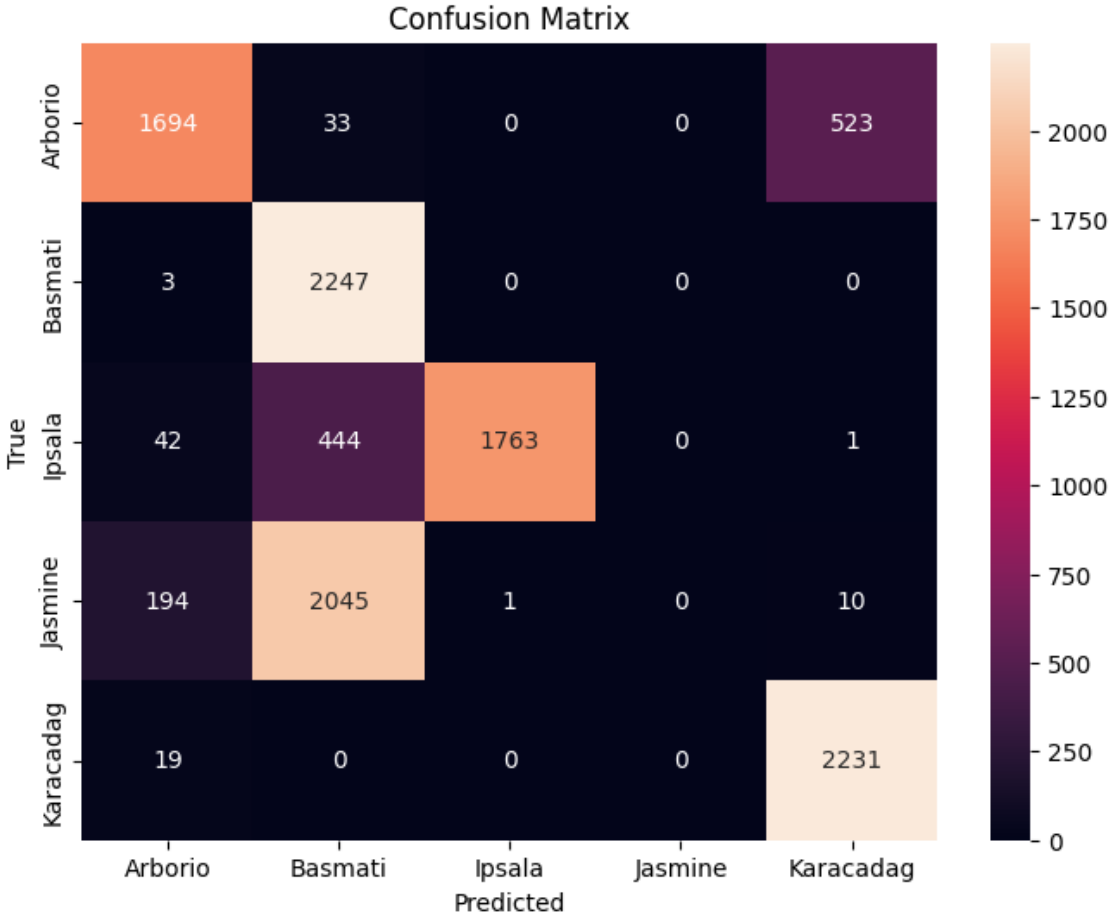
```
Epoch 1/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 119s 290ms/step - accuracy: 0.7126 - loss: 4.5743 - val_accuracy: 0.5135 - val_loss: 7.7805
Epoch 2/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 116s 283ms/step - accuracy: 0.7640 - loss: 3.7928 - val_accuracy: 0.4459 - val_loss: 8.9224
Epoch 3/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 120s 291ms/step - accuracy: 0.7200 - loss: 4.5063 - val_accuracy: 0.7121 - val_loss: 4.6336
Epoch 4/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 116s 282ms/step - accuracy: 0.7003 - loss: 4.8247 - val_accuracy: 0.4812 - val_loss: 8.3585
Epoch 5/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 116s 282ms/step - accuracy: 0.6438 - loss: 5.7373 - val_accuracy: 0.5679 - val_loss: 6.9641
Epoch 6/11
411/411 ━━━━━━━━━━━━━━━━━━━━ 116s 283ms/step - accuracy: 0.6859 - loss: 5.0594 - val_accuracy: 0.6876 - val_loss: 5.0322
```

```python
loss, acc = CNN.evaluate(test_ds, verbose=1)
print(f"Test loss: {loss:.4f}, Test accuracy: {acc*100:.2f}%")
```

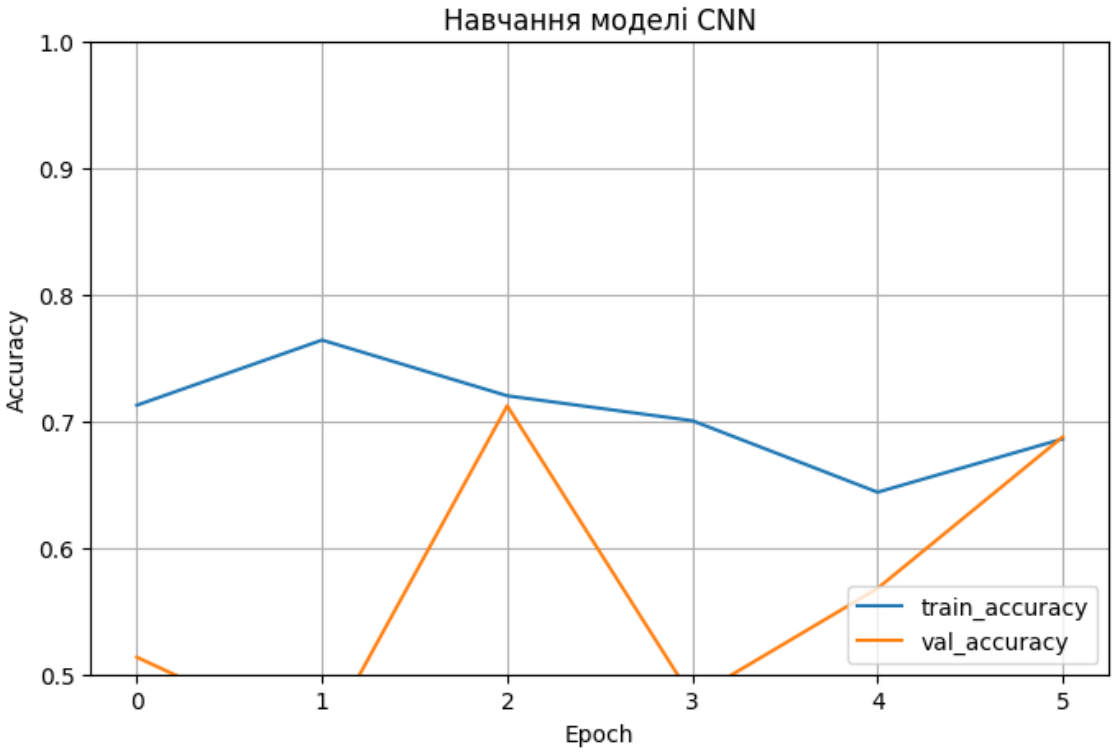
```
88/88 ━━━━━━━━━━━━━━━━━━━━ 10s 109ms/step - accuracy: 0.7053 - loss: 4.7436
Test loss: 4.7436, Test accuracy: 70.53%
```

```python
y_pred_proba = CNN.predict(test_ds, verbose=1)
y_pred = np.argmax(y_pred_proba, axis=1)

y_true = np.concatenate([y for x, y in test_ds], axis=0)
y_true = np.argmax(y_true, axis=1)

print("Classification Report:\n", classification_report(y_true, y_pred, target_names=class_names))
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", xticklabels=class_names, yticklabels=class_names)
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")
plt.show()
```

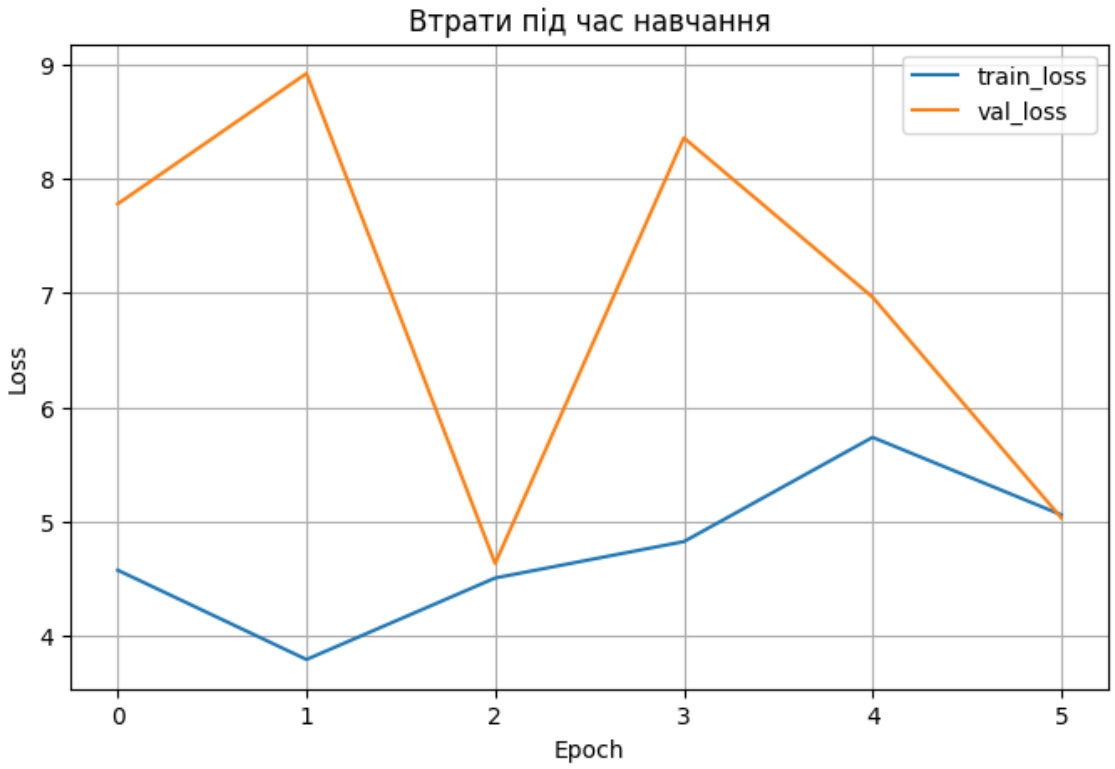
```
88/88 ━━━━━━━━━━━━━━━━━━━━ 7s 75ms/step
Classification Report:
              precision    recall  f1-score   support

     Arborio       0.87      0.75      0.81      2250
     Basmati       0.47      1.00      0.64      2250
      Ipsala       1.00      0.78      0.88      2250
     Jasmine       0.00      0.00      0.00      2250
   Karacadag       0.81      0.99      0.89      2250

    accuracy                           0.71     11250
   macro avg       0.63      0.71      0.64     11250
weighted avg       0.63      0.71      0.64     11250
```



```python
plt.figure(figsize=(8,5))
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.title("Навчання моделі CNN")
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



```python
plt.figure(figsize=(8,5))
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_los
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title("Втрати під час навчання")
plt.legend(loc='upper right')
plt.grid(True)
plt.show()
```



2. Зручніше код можна переглянути тут: https://github.com/highbrow-228/deep_learning_technologies/blob/main/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%20%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%961/lab_1.ipynb
3. Висновки:
    a. Модель з 1 епохи отримала точність 0.71, але після декількох епох точність падає, що може свідчити про **нестабільне навчання** або занадто великий learning rate.

b. Валідаційна точність коливається сильно (0.44 - 0.71 - 0.48 - 0.56 - 0.68), це **overfitting/underfitting** по черзі.
c. Test accuracy приблизно **70%**, що для 5 класів із балансованим набором даних – помірний результат.
d. Модель добре класифікує **Basmati та Karacadag**, але майже повністю **не вгадує Jasmine** (добре видно з confusion matrix)**.**
e. Ідеї для покращення:
    i. Додати шари або фільтри в мережу
    ii. Регулювання learning rate + регуляризація

Висновок: на даній лабораторній роботі я повторив як працює CNN. Досяг 70% точності на наборі даних, але валідаційна точність коливалась, що свідчить про нестабільне навчання. Модель добре класифікує Basmati та Karacadag, але майже не розуміє і не розпізнає Jasmine. Коливання точності зумовлені overfitting та високим learning rate. Покращення можливе через додавання шарів, регулювання learning rate та регуляризацію.