Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Звіт

Лабораторна робота № 6 з дисципліни

«Штучний інтелект в задачах обробки зображень»

«Реалізація архітектури AlexNet CNN за допомогою TensorFlow і Keras»

Виконав: *ІП-01 Черпак А. В.* _____

(шифр, прізвище, ім'я, по батькові)

Перевірив: *Нікітін В. А.* _____

(прізвище, ім'я, по батькові)

Київ 2022

Мета:

Отримати навички реалізації архітектури AlexNet CNN з використанням бібліотек TensorFlow та Keras.

Завдання

1. Реалізувати засобами TensorFlow та Keras AlexNet;
2. Отримати оцінку точності навченої мережі.

Хід роботи

1. Методи для завантаження та обробка даних:

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
from tensorflow.python.data.ops.dataset_ops import DatasetV1
from numpy import ndarray


def get data() -> tuple[DatasetV1, DatasetV1, DatasetV1]:
    (train_images, train_labels), (test_images, test_labels) =
keras.datasets.cifar10.load_data()

    validation_images, validation_labels = train_images[:5000],
train_labels[:5000]
    train_images, train_labels = train_images[5000:], train_labels[5000:]

    train_ds = tf.data.Dataset.from_tensor_slices((train_images,
train_labels))
    test_ds = tf.data.Dataset.from_tensor_slices((test_images, test_labels))
    validation_ds = tf.data.Dataset.from_tensor_slices((validation_images,
validation_labels))
    return train_ds, test_ds, validation_ds


def visualize_data(train_ds: DatasetV1, classes: list[str]) -> None:
    plt.figure(figsize=(20, 20))
    for i, (image, label) in enumerate(train_ds.take(10)):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(image)
        plt.title(classes[label.numpy()[0]])
        plt.axis('off')

    plt.show()


def process_images(image: ndarray, label: str) -> tuple[ndarray, str]:
    # Normalize images to have a mean of 0 and standard deviation of 1
    image = tf.image.per_image_standardization(image)
    # Resize images from 32x32 to 277x277
    image = tf.image.resize(image, (227, 227))
    return image, label


def get_ds_size(ds: DatasetV1) -> int:
    return tf.data.experimental.cardinality(ds).numpy()


def process_ds(ds: DatasetV1) -> DatasetV1:
    return
ds.map(process_images).shuffle(buffer_size=get_ds_size(ds)).batch(batch_size=3
2, drop_remainder=True)
```

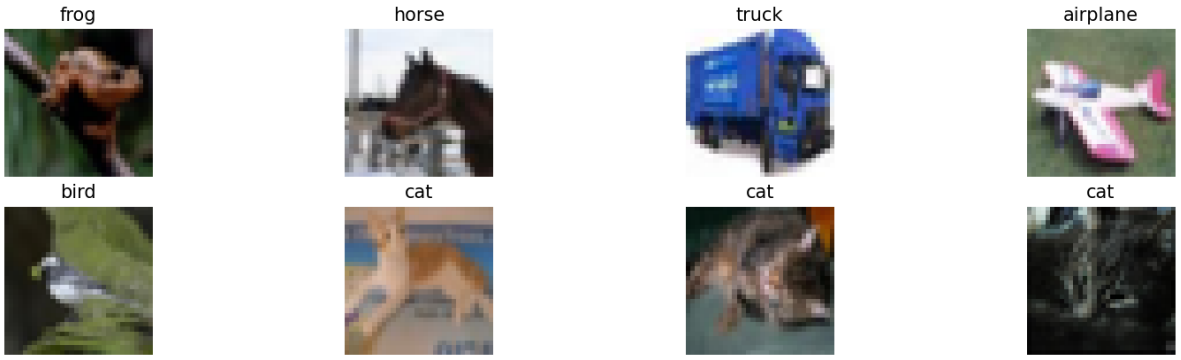
2. Створення та компіляція моделі:

```
import tensorflow as tf
from keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten,
Dense, Dropout
from keras.models import Sequential


def AlexNet(class_number) -> Sequential:
    model = Sequential([
        Conv2D(filters=96, kernel_size=(11, 11), strides=(4, 4),
               activation='relu', input_shape=(227, 227, 3)),
        BatchNormalization(),
        MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        Conv2D(filters=256, kernel_size=(5, 5), strides=(1, 1),
               activation='relu', padding='same'),
        BatchNormalization(),
        MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1),
               activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(filters=384, kernel_size=(3, 3), strides=(1, 1),
               activation='relu', padding='same'),
        BatchNormalization(),
        Conv2D(filters=256, kernel_size=(3, 3), strides=(1, 1),
               activation='relu', padding='same'),
        BatchNormalization(),
        MaxPool2D(pool_size=(3, 3), strides=(2, 2)),
        Flatten(),
        Dense(4096, activation='relu'),
        Dropout(0.5),
        Dense(4096, activation='relu'),
        Dropout(0.5),
        Dense(class_number, activation='softmax')
    ])

    return model


def compile_model(model: Sequential):
    model.compile(
        loss='sparse_categorical_crossentropy',
        optimizer=tf.optimizers.SGD(lr=0.001),
        metrics=['accuracy']
    )
    model.summary()
    return model
```

3. Тренування та оцінювання моделі:

```
from get_data import get_data, process_ds, get_ds_size, visualize_data
from nn_model import AlexNet, compile_model

CLASS_NAMES = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',
'horse', 'ship', 'truck']

if __name__ == '__main__':
    train_ds, test_ds, validation_ds = get_data()
    visualize_data(train_ds, CLASS_NAMES)

    train_ds_size = get_ds_size(train_ds)
    test_ds_size = get_ds_size(test_ds)
    validation_ds_size = get_ds_size(validation_ds)

    print("Training data size:", train_ds_size)
    print("Test data size:", test_ds_size)
    print("Validation data size:", validation_ds_size)

    train_ds = process_ds(train_ds)
    test_ds = process_ds(test_ds)
```

```
    validation_ds = process_ds(validation_ds)

    mdl = AlexNet(len(CLASS_NAMES))
    model = compile_model(mdl)

    model.fit(train_ds, epochs=10, validation_data=validation_ds,
validation_freq=1)
    model.evaluate(test_ds)
    model.summary()

    model.save('alexNet.h5')
```




```
Training data size: 45000
Test data size: 10000
Validation data size: 5000
```

```
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)

 conv2d_3 (Conv2D)           (None, 13, 13, 384)       1327488

 batch_normalization_3 (Batc  (None, 13, 13, 384)      1536
 hNormalization)

 conv2d_4 (Conv2D)           (None, 13, 13, 256)       884992

 batch_normalization_4 (Batc  (None, 13, 13, 256)      1024

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 256)        0
 2D)

 flatten (Flatten)           (None, 9216)              0

 dense (Dense)               (None, 4096)              37752832

 dropout (Dropout)           (None, 4096)              0

 dense_1 (Dense)             (None, 4096)              16781312

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 10)                40970

=================================================================
Total params: 58,327,818
Trainable params: 58,325,066
Non-trainable params: 2,752
```

```
Epoch 1/10
1406/1406 [==============================] - 1582s 1s/step - loss: 2.0493 - accuracy: 0.3516 - val_loss: 1.3424 - val_accuracy: 0.5274
Epoch 2/10
1406/1406 [==============================] - 1533s 1s/step - loss: 1.4775 - accuracy: 0.4755 - val_loss: 1.2074 - val_accuracy: 0.5749
Epoch 3/10
1406/1406 [==============================] - 1541s 1s/step - loss: 1.2941 - accuracy: 0.5396 - val_loss: 1.0796 - val_accuracy: 0.6262
Epoch 4/10
1406/1406 [==============================] - 1583s 1s/step - loss: 1.1652 - accuracy: 0.5880 - val_loss: 0.9930 - val_accuracy: 0.6536
Epoch 5/10
1406/1406 [==============================] - 1882s 1s/step - loss: 1.0627 - accuracy: 0.6232 - val_loss: 0.9402 - val_accuracy: 0.6715
Epoch 6/10
1406/1406 [==============================] - 2340s 2s/step - loss: 0.9765 - accuracy: 0.6569 - val_loss: 0.8566 - val_accuracy: 0.7063
Epoch 7/10
1406/1406 [==============================] - 2486s 2s/step - loss: 0.8991 - accuracy: 0.6815 - val_loss: 0.8074 - val_accuracy: 0.7204
Epoch 8/10
1406/1406 [==============================] - 2866s 2s/step - loss: 0.8430 - accuracy: 0.7031 - val_loss: 0.7865 - val_accuracy: 0.7274
Epoch 9/10
1406/1406 [==============================] - 3149s 2s/step - loss: 0.7866 - accuracy: 0.7224 - val_loss: 0.7525 - val_accuracy: 0.7420
Epoch 10/10
1406/1406 [==============================] - 3426s 2s/step - loss: 0.7358 - accuracy: 0.7410 - val_loss: 0.7402 - val_accuracy: 0.7442
```

Результати оцінювання:

```
312/312 [==============================] - 234s 745ms/step - loss: 0.7652 - accuracy: 0.7333
```

Контрольні запитання

1. AlexNet – це згорткова нейронна мережа для розпізнавання зображень, що була розроблена ще у 2012 році, але й досі задає стандарти у технологіях комп'ютерного зору.
2. Дана мережа складається з 5 згорткових і 3 повнозв'язних шарів. Між повнозв'язними шарами також розташовані Dropout, що дозволяють боротися з проблемою перенавчання, хоч і збільшують час навчання моделі. Мережа містить 58,327,818 параметрів (у моїй реалізації, в оригінальній, як зазначається, було 62,3м), і застосовує більше мільярда обчислень при прямому проході.
3. AlexNet вважається віхою загорткових нейронних мереж. Технології, що використовуються в ньому, досі задають стандарт для подібних мереж. Також дана модель дозволяє пряме введення зображення, а шари згортки можуть виділити краї зображень. Шляхом додавання більшої кількості загорткових шарів, можна досягти розпізнавання візуальних шаблонів більшої складності. Втім, дана модель менш глибока та недостатньо продуктивна порівняно з пізнішими моделями. Крім того, для ініціювання вагових коефіцієнтів використовується нормальний розподіл, що не дозволяє ефективно вирішувати проблему зникнення градієнта.
4. Під час компіляції мережі відбувається налаштування таких характеристик, як алгоритм оптимізації, функція втрат та швидкість навчання.
5. Під час навчання нейронної мережі ми передаємо у неї певні вхідні дані та очікуваний результат. Нейромережа спершу намагається передбачити результат для введених даних, а потім порівнює його з очікуваним результатом і коригує свої вагові коефіцієнти за допомогою методу зворотнього поширення помилки. Звісно, чим більшу кількість унікальних зображень було розглянуто мережею у процесі навчання, і чим більшою була кількість епох навчання – тим більш точною виявиться отримана мережа. Втім, варто також контролювати точність мережі на тестових даних, які не були безпосередньо розглянуті мережею у процесі навчання. Якщо точність з часом починає падати – ми, ймовірно, зіткнулися з таким явищем, як перенавчання, коли нейромережа «звикла» до таких вхідних даних і інші вже не сприймає. В такому випадку варто припинити навчання на цьому датасеті і спробувати потренувати мережу на нових даних.