НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №4
з дисципліни «Системи глибинного навчання»
на тему
“Розпізнавання двовимірних кольорових об’єктів за допомогою згорткової нейронної мережі”

Виконав:
студент групи КМ-01
Шолоп Л. О.

Викладач:
Терейковський І. А.

Київ — 2023

Зміст

Теоретичні відомості

Згорткові нейронні мережі (Convolutional Neural Networks або CNN) є потужним інструментом у сфері обробки зображень та визначення образів. Вони виникли спеціально для роботи з великою кількістю даних, такими як зображення, і здатні ефективно впоратися з просторовою структурою таких даних. Основні концепції та елементи згорткових мереж включають:

1. ***Згорткові шари (Convolutional Layers):***
   - Основний будівельний блок CNN. Використовується для виявлення локальних шаблонів у вхідних даних.
   - Згортка - операція, яка застосовує фільтр (ядро) до малих фрагментів вхідного зображення для виділення особливостей.
2. ***Пулінгові шари (Pooling Layers):***
   - Використовуються для зменшення просторових розмірів представлення та зменшення обчислювальної складності.
   - Зазвичай використовуються максимальне або середнє пулінги.
3. ***Повні з'єднувальні шари (Fully Connected Layers):***
   - Один або кілька повністю з'єднаних шарів в кінці мережі.
   - Використовуються для зведення просторової інформації в одновимірний вектор перед подачею на вихід.
4. ***Функції активації:***
   - Розташовані після кожного згорткового та повністю з'єднувального шару.
   - Зазвичай використовують ReLU (Rectified Linear Unit) або інші нелинійні функції для введення нелинійності в модель.
5. ***Фільтри та ядра:***
   - Фільтри або ядра використовуються для виявлення різних властивостей в зображеннях, таких як границі, форми, текстури тощо.
6. ***Стеки згорткових шарів:***
   - Згорткові мережі зазвичай мають декілька шарів, які утворюють стек для більш складної і абстрактної репрезентації.
7. ***Занурення (Invariance):***
   - Згорткові шари дозволяють нейронній мережі виявляти ознаки, які є інваріантними відносно зсувів та зміни масштабу, що робить їх ефективними для визначення образів.
8. ***Регуляризація:***
   - Для запобігання перенавчанню використовують техніки регуляризації, такі як Dropout, які випадковим чином вимикають деякі нейрони під час тренування.

Основна частина

Запускаємо таймер, щоб знати скільки часу витратить програма на навчання моделі та її тестування.

Будемо імпортувати такі бібліотеки:

- Numpy – для математичних розрахунків
- Pandas – для табличного відображення виводу
- Matplotlib – для демонстрації картинок у датасеті
- TensorFlow – для створення мережі CNN
- Keras – для створення шарів моделі
- Sklearn – для оцінки моделі

Завантажуємо датасет. Нормалізуємо дані в ньому шляхом поділу на 255. Перекодовуємо мітки в категорійні змінні. Розбиваємо тестову вибірку на 2 рівні частини. Одна частина лишиться тестовим набором (тест моделі в самому кінці), інша частина валідаційним набором (тест моделі після кожної епохи). Ініціалізуємо розмір навчальної вибірки аналогічно до прикладу в умові. Створюємо список із назвами класів зображень

Відображаємо картинки, для розуміння з чим будемо працювати

Ініціалізуємо структуру мережі аналогічно до прикладу в умові. Додаємо шари Dropout для запобіганню перенавчання моделі. В якості оптимізатора будемо використовувати SGD (градієнтний спуск) із функцією втрат – категорійною кроссентропією, в якості метрики на кожній епосі будемо відображати точність.

Якщо модель навчена, то імпортуємо її з файлу, якщо ні, то навчаємо її протягом 25 епох та зберігаємо готову модель у файл.

Тепер коли модель завершила навчання знайдемо метрики для розуміння наскільки добре модель навчилась:

```
Метрики моделі на тестових даних:
  Метрика  Значення
 Accuracy  0.691400
Precision  0.700315
   Recall  0.691400
 F1-score  0.686122
```

Отже, точність моделі приблизно рівна 68-70 %

Для реального використання це замала точність. Покращити результати можна шляхом збільшення епох навчання, підбору кращого оптимізатора, фільтрації зображень, які є важко розпізнаваними.

У результаті програма навчила та протестувала модель за 24 хв, 55 сек

Додаток А – Код програми

```
import time
start_timer = time.time()

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score
from keras import layers, models
from keras.optimizers import SGD
from keras.datasets import cifar10
from keras.utils import to_categorical
```

```
# Завантажимо дані
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Нормалізуємо дані
X_train = X_train / 255.0
X_test  = X_test / 255.0

# Створимо мітки класів
y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test,  10)

# Створимо валідаційну вибірку для тестування моделі на кожній епосі
X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.5,
random_state=10)

# Ініціалізуємо розмір навчальної вибірки на кожній епосі
batch_size = 32

# Ініціалізуємо назви класів датасету cifar 10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',
'horse', 'ship', 'truck']
```

```
print(f'Записів у тренувальній вибірці: {len(X_train)}')
print(f'Записів у тестовій вибірці: {len(X_test)}')
print(f'Записів у валідаційній вибірці: {len(X_val)}')
```

```
selected_images = []
selected_labels = []
```

```
for class_index in range(10):
    index = np.where(np.argmax(y_train, axis=1) == class_index)[0][42]
    selected_images.append(X_train[index])
    selected_labels.append(y_train[index])

plt.figure(figsize=(15, 5))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(selected_images[i])
    plt.title(class_names[np.argmax(selected_labels[i])])
    plt.axis('off')

plt.show()
```

```
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# Додавання повнозв'язних шарів
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))  # 10 класів виводу

# Компіляція моделі
sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
try:
    model = models.load_model('my_model.keras')
except:
    history = model.fit(X_train, y_train,
                        epochs=25,
                        batch_size=batch_size,
                        validation_data=(X_val, y_val))
    model.save('my_model.keras') # Зберігаємо модель
```

```
test_results = model.evaluate(X_test, y_test, verbose=0)

test_accuracy = test_results[1]
predictions = np.argmax(model.predict(X_test, verbose=0), axis=1)
test_precision = precision_score(np.argmax(y_test, axis=1), predictions,
average='weighted')
test_recall = recall_score(np.argmax(y_test, axis=1), predictions,
average='weighted')
test_f1_score = f1_score(np.argmax(y_test, axis=1), predictions,
average='weighted')

metrics = pd.DataFrame({
    'Метрика': ['Accuracy', 'Precision', 'Recall', 'F1-score'],
    'Значення': [test_accuracy, test_precision, test_recall, test_f1_score]
})

print("Метрики моделі на тестових даних:")
print(metrics.to_string(index=False))
```

```
elapsed_time = time.time() - start_timer
minutes, seconds = divmod(elapsed_time, 60)

if minutes > 0:
    print(f'Час роботи: {int(minutes)} хв, {int(seconds)} сек')
else:
    print(f'Час роботи: {int(seconds)} сек')
```