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
Романецький М.С.

Викладач:
Професор кафедри ПМА
Терейковський І. А.

Київ — 2023

# Зміст

# Теоретичні відомості

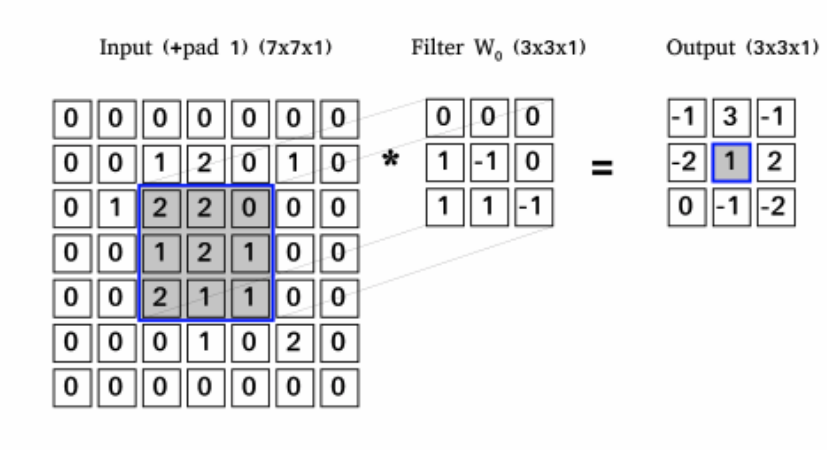
Згорткові нейронні мережі (ЗНМ) є одним з ключових типів нейромереж, розроблених для обробки великих обсягів даних, таких як зображення та відео. Основним елементом ЗНМ є згорткові шари, які відповідають за ефективне виявлення локальних особливостей у вхідних даних.

Головна ідея згорткових шарів полягає в тому, щоб використовувати фільтри або ядра для локального виявлення певних характеристик, таких як краї, форми або текстури, у різних частинах вхідного зображення. Ці фільтри застосовуються повздовж ширини та висоти вхідних даних, видаючи карту ознак (feature map), яка підкреслює наявність різних паттернів у вхідних даних.

Для подальшої обробки та аналізу отриманих карт ознак використовуються пулінгові шари, які зменшують просторовий обсяг даних, зберігаючи при цьому важливу інформацію. Цей процес дозволяє забезпечити інваріантність до масштабу та позиції виявлених паттернів.

Важливою характеристикою ЗНМ є можливість використання багатоканальних карт ознак, що дозволяє моделі адаптуватися до різних аспектів вхідних даних. Також, згорткові нейронні мережі часто включають повнозв’язні шари та шари активації для додаткового виявлення глобальних паттернів та узагальнення інформації.

ЗНМ успішно використовуються у завданнях комп'ютерного зору, розпізнавання образів та обробки природної мови, забезпечуючи високу ефективність та точність у вирішенні складних завдань з аналізу зображень та даних. Приклад згортки:




https://www.cosmos.esa.int/documents/5081622/0/Example_of_convolution.png/ae84d62e-d045-3b96-fbcd-9322279a5d16?t=1608221198626

# Основна частина

## Імпорт бібліотек

Запускаємо таймер, щоб в кінці вивести повний час роботи програми. Імпортуємо бібліотеки та виводимо версію TensorFlow на екран.

```
TensorFlow version: 2.14.0
```

Програма буде навчати та тестувати НМ на процесорі, оскільки на моєму пристрої TensorFlow не може розпізнати жодного GPU

```
[]
Порожній список означає, що TensorFlow не може знайти жодного доступного GPU
```

## Завантаження даних та їх нормалізація

Визначаємо розмір Batch`у рівним 32

Завантажуємо навчальні та тестові дані датасету cifar10

Ділимо значення на 255, тим самим нормалізуємо їх від 0 до 1

Перекодовуємо категорійні мітки

Оскільки зараз є лише навчальна та тестова вибірки. Поділимо тестову вибірку на 2 рівні частини. Одна частина буде валідаційною вибіркою, тобто такою, на якій НМ буде тестуватись після кожної епохи. Друга частина буде тестовою, тобто на ній буде тестуватись уже навчена модель.

## Розміри навчального, тестового та валідаційного наборів

```
Кількість записів у 'X_train': 50000
Кількість записів у 'X_test':  5000
Кількість записів у 'X_val':   5000
```

## Зображення кожного класу в датасеті cifar10


## Структура моделі CNN

### Функція втрат (loss function)

Використання `categorical_crossentropy` як функції втрат для моделі з 10 класами має наступні *переваги* та *недоліки*:

*Переваги*:

- `categorical_crossentropy` є стандартною функцією втрат для багатокласової класифікації. Вона вимірює різницю між двома ймовірнісними розподілами: істинними мітками та передбаченнями моделі.
- Вона надає велику помилку, коли модель передбачає неправильний клас з високою впевненістю, що допомагає моделі швидше навчитися.

*Недоліки*:

- Якщо деякі класи є незбалансованими, `categorical_crossentropy` може призвести до того, що модель буде занадто зосереджена на більш поширених класах і ігноруватиме рідкісні класи.
- Вона може бути чутливою до шуму в мітках, оскільки вона штрафує навіть малі відхилення від істинних міток.

Було протестовано різні структури моделі:

- З/без шарів Dropout`ів
- З оптимізатором Adam
- З оптимізатором SGD
- З виводом таких метрик на кожній епосі (accuracy, precision, recall)

## Клас Callback

Основна ідея цього класу полягає в відслідковуванні логів після кожної епохи. Якщо точність на валідаційному наборі ('val_accuracy') буде більше 95 %, то

навчання зупиниться, навіть якщо всі епохи ще не пройшли. Тобто критерієм зупинки навчання є або проходження всіх епох, або досягнення відповідної точності.

## Навчання моделі CNN

Спочатку програма намагається знайти та завантажити модель із файлу 'CNN_model.keras' із тієї самої директорії, де знаходиться файл програми.

Якщо цього не вдається зробити, модель переходить до навчання. Кількість епох навчання = 150. Коли модель зупинить навчання по одному з критеріїв, то вона збереже файл моделі у файлі 'CNN_model.keras'

Результати епох навчання наведені в ***додатку Б***, оскільки вони займають багато місця. Якщо уважно придивитись до тих результатів, то можна помітити, що модель завершила, навчання по критерію кількості епох. Але точність валідаційного набору найвища не на останні епосі.

```
Epoch 150/150
```

```
1563/1563 [==============================] - 55s 35ms/step - loss: 0.6374
- accuracy: 0.7793 - val_loss: 0.7106 - val_accuracy: 0.7642
```

Найвища точність по валідаційних даних була досягнута тут:

```
Epoch 138/150
```

```
1563/1563 [==============================] - 55s 35ms/step - loss: 0.6419
- accuracy: 0.7776 - val_loss: 0.7128 - val_accuracy: 0.7724
```

Далі в коді модель приймала результати останньої, епохи, але за необхідності можна перемістити до 138-ї епохи звернувшись до змінної `history`

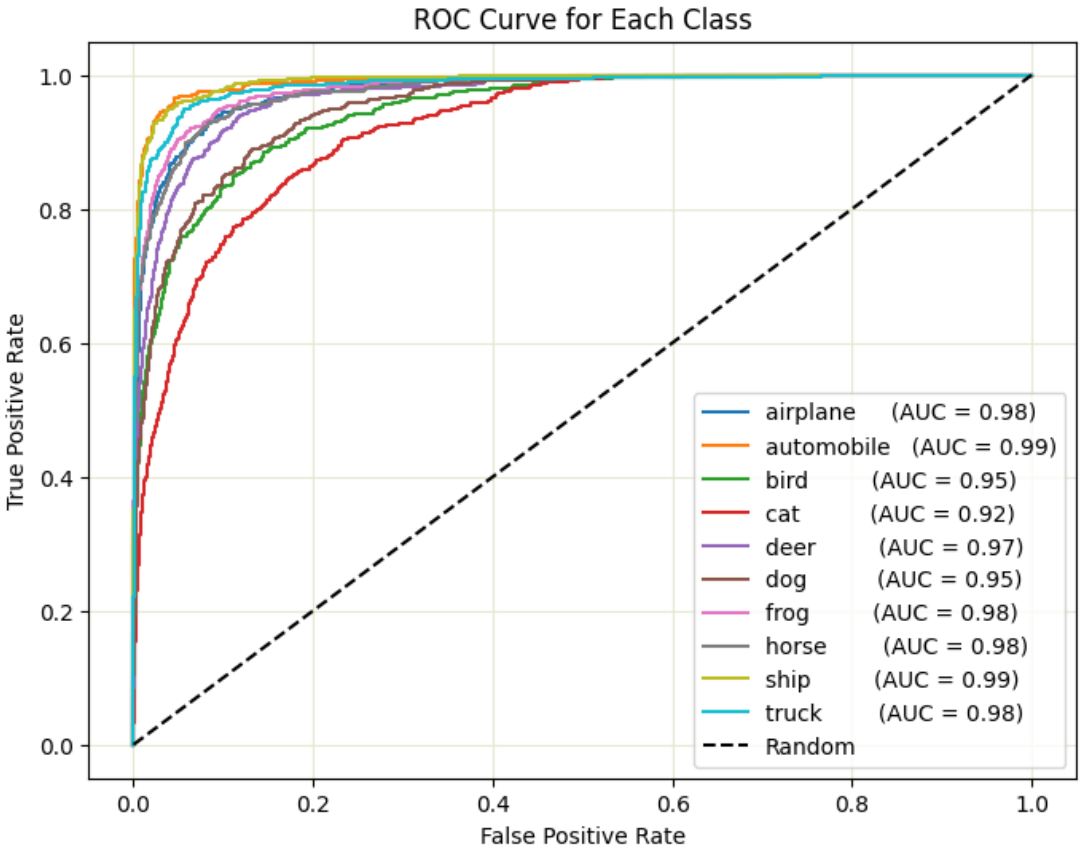
```
callbacks = myCallback()

try:
    model = models.load_model('CNN_model.keras')
except:
    history = model.fit(X_train, y_train,
                        epochs=150,
                        batch_size=batch_size,
                        validation_data=(X_val, y_val),
                        callbacks=[callbacks])
    model.save('CNN_model.keras') # Зберігаємо модель
```

## Тестування моделі та знаходження метрик

```
Accuracy  моделі на тестових даних: 76.58 %
Precision моделі на тестових даних: 76.42 %
Recall    моделі на тестових даних: 76.58 %
F1-score  моделі на тестових даних: 76.34 %
```

## Побудова ROC-кривої для кожного класу

Оскільки ROC-крива показує результати бінарної класифікації, побудуємо 10 кривих для 10 класів.




## Час роботи програми та характеристики ПК

```
Час роботи: 2 год, 22 хв, 36 сек
Процесор:   AMD Ryzen 5 3550H with Radeon Vega
ОЗП:        8.00 ГБ
ОС:         Windows 10 Pro
```

# Додаток А – Код програми

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

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, f1_score, roc_curve, auc

from keras import layers, models
from keras.optimizers import SGD
from keras.datasets import cifar10
from keras.utils import to_categorical

print(f'TensorFlow version: {tf.__version__}')
```

```
print(tf.config.list_physical_devices('GPU'))
print(f'Порожній список означає, що TensorFlow не може знайти жодного доступного GPU')
```

```
batch_size = 32

(X_train, y_train), (X_test, y_test) = cifar10.load_data()

X_train = X_train / 255.0
X_test  = X_test / 255.0

y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test,  10)

X_test, X_val, y_test, y_val = train_test_split(X_test, y_test, test_size=0.5, random_state=42)
```

```
print(f'Кількість записів у \'X_train\': {len(X_train)}')
print(f'Кількість записів у \'X_test\':  {len(X_test)}')
print(f'Кількість записів у \'X_val\':   {len(X_val)}')
```

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

```python
# Вибір по одному зображенню для кожного класу
selected_images = []
selected_labels = []

for class_index in range(10):
    # Знаходження першого елемента для кожного класу
    index = np.where(np.argmax(y_train, axis=1) == class_index)[0][0]
    selected_images.append(X_train[index])
    selected_labels.append(y_train[index])

# Відображення зображень
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()
```

```python
model = models.Sequential()

# Додавання згорткових шарів та шарів пулінгу
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))  # Додаємо Dropout

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

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))  # Додаємо Dropout
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'])



# Оптимізатор Adam
```

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

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if logs.get('val_accuracy') is not None and logs.get('val_accuracy') >
0.95:
            self.model.stop_training = True
            print(f'\n\nAccuracy моделі на тестових даних > 95 %, зупинка навчання
після епохи № {epoch+1}')
```

```
callbacks = myCallback()

try:
    model = models.load_model('CNN_model.keras')
except:
    history = model.fit(X_train, y_train,
                        epochs=150,
                        batch_size=batch_size,
                        validation_data=(X_val, y_val),
                        callbacks=[callbacks])
    model.save('CNN_model.keras') # Зберігаємо модель


# print(f'Введіть 0 для створення моделі, 1 для читання з файлу')
# chose = int(input('Введіть 0 для створення моделі, 1 для читання з файлу:  '))
# print(f'Введено: {chose}\n')

# if chose == 1:
#     model = models.load_model('CNN_model.keras')
# else:
#     history = model.fit(X_train, y_train,
#                         epochs=150,
#                         batch_size=batch_size,
#                         validation_data=(X_val, y_val),
#                         callbacks=[callbacks])
#     model.save('CNN_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')

print(f'Accuracy  моделі на тестових даних: {test_accuracy * 100:.2f} %')
print(f'Precision моделі на тестових даних: {test_precision * 100:.2f} %')
print(f'Recall    моделі на тестових даних: {test_recall * 100:.2f} %')
print(f'F1-score  моделі на тестових даних: {test_f1_score * 100:.2f} %')
```

```
predictions = model.predict(X_test, verbose=0)
fpr = dict()
tpr = dict()
roc_auc = dict()

for i in range(10):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], predictions[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Побудова ROC-кривих
plt.figure(figsize=(8, 6))

for i in range(10):
    label = f'{class_names[i]:<12} (AUC = {roc_auc[i]:.2f})'
    plt.plot(fpr[i], tpr[i], label=label)

plt.plot([0, 1], [0, 1], 'k--', label='Random')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve for Each Class')
plt.legend(loc='best')
plt.grid(c='#E7E8D2')
plt.show()
```

```
elapsed_time = time.time() - start_timer
hours, remainder = divmod(elapsed_time, 3600)
minutes, seconds = divmod(remainder, 60)

if hours > 0:
    print(f'Час роботи: {int(hours)} год, {int(minutes)} хв, {int(seconds)} сек')
elif minutes > 0:
    print(f'Час роботи: {int(minutes)} хв, {int(seconds)} сек')
else:
    print(f'Час роботи: {int(seconds)} сек')
print(f'Процесор:   AMD Ryzen 5 3550H with Radeon Vega')
print(f'ОЗП:        8.00 ГБ')
print(f'ОС:         Windows 10 Pro')
```

# Додаток Б – Епохи навчання моделі

```
Epoch 1/150

1563/1563 [==============================] - 66s 39ms/step - loss: 1.7596
- accuracy: 0.3466 - val_loss: 1.4314 - val_accuracy: 0.5006

Epoch 2/150

1563/1563 [==============================] - 55s 35ms/step - loss: 1.4282
- accuracy: 0.4890 - val_loss: 1.2293 - val_accuracy: 0.5632

Epoch 3/150

1563/1563 [==============================] - 55s 35ms/step - loss: 1.2998
- accuracy: 0.5379 - val_loss: 1.1151 - val_accuracy: 0.6078

Epoch 4/150

1563/1563 [==============================] - 56s 36ms/step - loss: 1.2183
- accuracy: 0.5725 - val_loss: 1.0496 - val_accuracy: 0.6292

Epoch 5/150

1563/1563 [==============================] - 56s 36ms/step - loss: 1.1586
- accuracy: 0.5930 - val_loss: 1.0628 - val_accuracy: 0.6302

Epoch 6/150

1563/1563 [==============================] - 57s 36ms/step - loss: 1.1126
- accuracy: 0.6084 - val_loss: 1.0092 - val_accuracy: 0.6444

Epoch 7/150

1563/1563 [==============================] - 57s 37ms/step - loss: 1.0697
- accuracy: 0.6266 - val_loss: 0.9770 - val_accuracy: 0.6568

Epoch 8/150

1563/1563 [==============================] - 57s 36ms/step - loss: 1.0415
- accuracy: 0.6376 - val_loss: 0.9174 - val_accuracy: 0.6844

Epoch 9/150

1563/1563 [==============================] - 57s 37ms/step - loss: 1.0143
- accuracy: 0.6494 - val_loss: 0.9113 - val_accuracy: 0.6824

Epoch 10/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.9939
- accuracy: 0.6559 - val_loss: 0.8885 - val_accuracy: 0.6916

Epoch 11/150
```

```
1563/1563 [==============================] - 57s 36ms/step - loss: 0.9664
- accuracy: 0.6647 - val_loss: 0.8679 - val_accuracy: 0.7024

Epoch 12/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.9599
- accuracy: 0.6679 - val_loss: 1.0460 - val_accuracy: 0.6614

Epoch 13/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.9382
- accuracy: 0.6740 - val_loss: 0.8777 - val_accuracy: 0.7008

Epoch 14/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.9215
- accuracy: 0.6801 - val_loss: 0.8579 - val_accuracy: 0.7032

Epoch 15/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.9096
- accuracy: 0.6881 - val_loss: 0.9694 - val_accuracy: 0.6692

Epoch 16/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8959
- accuracy: 0.6908 - val_loss: 0.9354 - val_accuracy: 0.6834

Epoch 17/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.8929
- accuracy: 0.6932 - val_loss: 0.8985 - val_accuracy: 0.6916

Epoch 18/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.8812
- accuracy: 0.6943 - val_loss: 0.8355 - val_accuracy: 0.7146

Epoch 19/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.8658
- accuracy: 0.6987 - val_loss: 0.8210 - val_accuracy: 0.7176

Epoch 20/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8567
- accuracy: 0.7056 - val_loss: 0.8628 - val_accuracy: 0.7134

Epoch 21/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8542
- accuracy: 0.7039 - val_loss: 0.8479 - val_accuracy: 0.7074

Epoch 22/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8443
- accuracy: 0.7097 - val_loss: 0.8515 - val_accuracy: 0.7086
```

```
Epoch 23/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8389
- accuracy: 0.7096 - val_loss: 0.8402 - val_accuracy: 0.7226

Epoch 24/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.8314
- accuracy: 0.7113 - val_loss: 0.7930 - val_accuracy: 0.7286

Epoch 25/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8254
- accuracy: 0.7154 - val_loss: 0.7951 - val_accuracy: 0.7322

Epoch 26/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.8145
- accuracy: 0.7194 - val_loss: 0.8140 - val_accuracy: 0.7240

Epoch 27/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.8089
- accuracy: 0.7204 - val_loss: 0.7997 - val_accuracy: 0.7268

Epoch 28/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8056
- accuracy: 0.7222 - val_loss: 0.8037 - val_accuracy: 0.7304

Epoch 29/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.7920
- accuracy: 0.7257 - val_loss: 0.8290 - val_accuracy: 0.7252

Epoch 30/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.8006
- accuracy: 0.7240 - val_loss: 0.7849 - val_accuracy: 0.7378

Epoch 31/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7884
- accuracy: 0.7264 - val_loss: 0.8527 - val_accuracy: 0.7210

Epoch 32/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7867
- accuracy: 0.7287 - val_loss: 0.7971 - val_accuracy: 0.7304

Epoch 33/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7790
- accuracy: 0.7316 - val_loss: 0.8025 - val_accuracy: 0.7354

Epoch 34/150
```

```
1563/1563 [==============================] - 56s 36ms/step - loss: 0.7716
- accuracy: 0.7317 - val_loss: 0.7669 - val_accuracy: 0.7396

Epoch 35/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7674
- accuracy: 0.7322 - val_loss: 0.7502 - val_accuracy: 0.7436

Epoch 36/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7689
- accuracy: 0.7350 - val_loss: 0.7792 - val_accuracy: 0.7402

Epoch 37/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7667
- accuracy: 0.7344 - val_loss: 0.8053 - val_accuracy: 0.7310

Epoch 38/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7623
- accuracy: 0.7348 - val_loss: 0.7667 - val_accuracy: 0.7484

Epoch 39/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7563
- accuracy: 0.7389 - val_loss: 0.7831 - val_accuracy: 0.7390

Epoch 40/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7528
- accuracy: 0.7388 - val_loss: 0.7671 - val_accuracy: 0.7482

Epoch 41/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7523
- accuracy: 0.7402 - val_loss: 0.7559 - val_accuracy: 0.7404

Epoch 42/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7507
- accuracy: 0.7405 - val_loss: 0.7438 - val_accuracy: 0.7520

Epoch 43/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7438
- accuracy: 0.7420 - val_loss: 0.7911 - val_accuracy: 0.7404

Epoch 44/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7427
- accuracy: 0.7434 - val_loss: 0.7661 - val_accuracy: 0.7478

Epoch 45/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7373
- accuracy: 0.7459 - val_loss: 0.7618 - val_accuracy: 0.7460
```

```
Epoch 46/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7295
- accuracy: 0.7475 - val_loss: 0.7706 - val_accuracy: 0.7450

Epoch 47/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7320
- accuracy: 0.7463 - val_loss: 0.7555 - val_accuracy: 0.7490

Epoch 48/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7264
- accuracy: 0.7480 - val_loss: 0.7803 - val_accuracy: 0.7386

Epoch 49/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7269
- accuracy: 0.7494 - val_loss: 0.7686 - val_accuracy: 0.7442

Epoch 50/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.7183
- accuracy: 0.7519 - val_loss: 0.7379 - val_accuracy: 0.7520

Epoch 51/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7265
- accuracy: 0.7452 - val_loss: 0.7607 - val_accuracy: 0.7464

Epoch 52/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7237
- accuracy: 0.7493 - val_loss: 0.7401 - val_accuracy: 0.7510

Epoch 53/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.7245
- accuracy: 0.7499 - val_loss: 0.7777 - val_accuracy: 0.7434

Epoch 54/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7144
- accuracy: 0.7504 - val_loss: 0.7815 - val_accuracy: 0.7368

Epoch 55/150

1563/1563 [==============================] - 61s 39ms/step - loss: 0.7192
- accuracy: 0.7511 - val_loss: 0.7632 - val_accuracy: 0.7430

Epoch 56/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7102
- accuracy: 0.7542 - val_loss: 0.7878 - val_accuracy: 0.7382

Epoch 57/150
```

```
1563/1563 [==============================] - 57s 36ms/step - loss: 0.7207
- accuracy: 0.7463 - val_loss: 0.7686 - val_accuracy: 0.7452

Epoch 58/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.7128
- accuracy: 0.7539 - val_loss: 0.7277 - val_accuracy: 0.7588

Epoch 59/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.7085
- accuracy: 0.7539 - val_loss: 0.7759 - val_accuracy: 0.7428

Epoch 60/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7055
- accuracy: 0.7566 - val_loss: 0.7280 - val_accuracy: 0.7564

Epoch 61/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.7042
- accuracy: 0.7543 - val_loss: 0.7610 - val_accuracy: 0.7506

Epoch 62/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.7031
- accuracy: 0.7544 - val_loss: 0.8034 - val_accuracy: 0.7428

Epoch 63/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7019
- accuracy: 0.7577 - val_loss: 0.7999 - val_accuracy: 0.7480

Epoch 64/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.6958
- accuracy: 0.7587 - val_loss: 0.7433 - val_accuracy: 0.7566

Epoch 65/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6993
- accuracy: 0.7575 - val_loss: 0.7468 - val_accuracy: 0.7514

Epoch 66/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.7006
- accuracy: 0.7586 - val_loss: 0.7693 - val_accuracy: 0.7454

Epoch 67/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6954
- accuracy: 0.7601 - val_loss: 0.7177 - val_accuracy: 0.7536

Epoch 68/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6958
- accuracy: 0.7578 - val_loss: 0.7208 - val_accuracy: 0.7606
```

```
Epoch 69/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6926
- accuracy: 0.7610 - val_loss: 0.7690 - val_accuracy: 0.7510

Epoch 70/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6918
- accuracy: 0.7607 - val_loss: 0.7430 - val_accuracy: 0.7586

Epoch 71/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6912
- accuracy: 0.7603 - val_loss: 0.7345 - val_accuracy: 0.7554

Epoch 72/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6919
- accuracy: 0.7610 - val_loss: 0.7219 - val_accuracy: 0.7604

Epoch 73/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6888
- accuracy: 0.7603 - val_loss: 0.7408 - val_accuracy: 0.7560

Epoch 74/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.6865
- accuracy: 0.7626 - val_loss: 0.7559 - val_accuracy: 0.7556

Epoch 75/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6839
- accuracy: 0.7629 - val_loss: 0.7222 - val_accuracy: 0.7586

Epoch 76/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6876
- accuracy: 0.7613 - val_loss: 0.7314 - val_accuracy: 0.7548

Epoch 77/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.6804
- accuracy: 0.7633 - val_loss: 0.7206 - val_accuracy: 0.7622

Epoch 78/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6887
- accuracy: 0.7600 - val_loss: 0.7843 - val_accuracy: 0.7482

Epoch 79/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6773
- accuracy: 0.7656 - val_loss: 0.8329 - val_accuracy: 0.7394

Epoch 80/150
```

```
1563/1563 [==============================] - 57s 36ms/step - loss: 0.6817
- accuracy: 0.7610 - val_loss: 0.7870 - val_accuracy: 0.7430

Epoch 81/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6832
- accuracy: 0.7635 - val_loss: 0.7105 - val_accuracy: 0.7634

Epoch 82/150

1563/1563 [==============================] - 57s 36ms/step - loss: 0.6823
- accuracy: 0.7615 - val_loss: 0.7497 - val_accuracy: 0.7602

Epoch 83/150

1563/1563 [==============================] - 62s 39ms/step - loss: 0.6754
- accuracy: 0.7664 - val_loss: 0.7705 - val_accuracy: 0.7528

Epoch 84/150

1563/1563 [==============================] - 67s 43ms/step - loss: 0.6781
- accuracy: 0.7637 - val_loss: 0.8707 - val_accuracy: 0.7276

Epoch 85/150

1563/1563 [==============================] - 61s 39ms/step - loss: 0.6722
- accuracy: 0.7638 - val_loss: 0.7331 - val_accuracy: 0.7586

Epoch 86/150

1563/1563 [==============================] - 63s 40ms/step - loss: 0.6756
- accuracy: 0.7632 - val_loss: 0.7682 - val_accuracy: 0.7504

Epoch 87/150

1563/1563 [==============================] - 63s 41ms/step - loss: 0.6727
- accuracy: 0.7669 - val_loss: 0.7536 - val_accuracy: 0.7558

Epoch 88/150

1563/1563 [==============================] - 62s 40ms/step - loss: 0.6688
- accuracy: 0.7670 - val_loss: 0.7243 - val_accuracy: 0.7650

Epoch 89/150

1563/1563 [==============================] - 65s 42ms/step - loss: 0.6713
- accuracy: 0.7679 - val_loss: 0.7196 - val_accuracy: 0.7610

Epoch 90/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6706
- accuracy: 0.7689 - val_loss: 0.7541 - val_accuracy: 0.7538

Epoch 91/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6707
- accuracy: 0.7691 - val_loss: 0.7452 - val_accuracy: 0.7590
```

```
Epoch 92/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6677
- accuracy: 0.7690 - val_loss: 0.7133 - val_accuracy: 0.7628

Epoch 93/150

1563/1563 [==============================] - 64s 41ms/step - loss: 0.6651
- accuracy: 0.7697 - val_loss: 0.7490 - val_accuracy: 0.7596

Epoch 94/150

1563/1563 [==============================] - 59s 38ms/step - loss: 0.6689
- accuracy: 0.7678 - val_loss: 0.7298 - val_accuracy: 0.7616

Epoch 95/150

1563/1563 [==============================] - 59s 38ms/step - loss: 0.6638
- accuracy: 0.7695 - val_loss: 0.7181 - val_accuracy: 0.7642

Epoch 96/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.6610
- accuracy: 0.7688 - val_loss: 0.7465 - val_accuracy: 0.7604

Epoch 97/150

1563/1563 [==============================] - 60s 38ms/step - loss: 0.6765
- accuracy: 0.7656 - val_loss: 0.7201 - val_accuracy: 0.7660

Epoch 98/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.6667
- accuracy: 0.7701 - val_loss: 0.7599 - val_accuracy: 0.7596

Epoch 99/150

1563/1563 [==============================] - 60s 38ms/step - loss: 0.6616
- accuracy: 0.7709 - val_loss: 0.7230 - val_accuracy: 0.7568

Epoch 100/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6642
- accuracy: 0.7708 - val_loss: 0.7215 - val_accuracy: 0.7624

Epoch 101/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6634
- accuracy: 0.7692 - val_loss: 0.7301 - val_accuracy: 0.7628

Epoch 102/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6694
- accuracy: 0.7691 - val_loss: 0.7686 - val_accuracy: 0.7532

Epoch 103/150
```

```
1563/1563 [==============================] - 56s 36ms/step - loss: 0.6628
- accuracy: 0.7720 - val_loss: 0.7396 - val_accuracy: 0.7602

Epoch 104/150

1563/1563 [==============================] - 57s 37ms/step - loss: 0.6640
- accuracy: 0.7703 - val_loss: 0.7300 - val_accuracy: 0.7554

Epoch 105/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6668
- accuracy: 0.7703 - val_loss: 0.7248 - val_accuracy: 0.7614

Epoch 106/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6638
- accuracy: 0.7711 - val_loss: 0.7263 - val_accuracy: 0.7560

Epoch 107/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6569
- accuracy: 0.7727 - val_loss: 0.7292 - val_accuracy: 0.7614

Epoch 108/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6521
- accuracy: 0.7730 - val_loss: 0.7389 - val_accuracy: 0.7590

Epoch 109/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6571
- accuracy: 0.7719 - val_loss: 0.7872 - val_accuracy: 0.7536

Epoch 110/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6575
- accuracy: 0.7700 - val_loss: 0.7508 - val_accuracy: 0.7606

Epoch 111/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6543
- accuracy: 0.7716 - val_loss: 0.7184 - val_accuracy: 0.7630

Epoch 112/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6504
- accuracy: 0.7733 - val_loss: 0.7303 - val_accuracy: 0.7638

Epoch 113/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6480
- accuracy: 0.7764 - val_loss: 0.7311 - val_accuracy: 0.7600

Epoch 114/150

1563/1563 [==============================] - 62s 40ms/step - loss: 0.6485
- accuracy: 0.7764 - val_loss: 0.7472 - val_accuracy: 0.7646
```

```
Epoch 115/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6579
- accuracy: 0.7736 - val_loss: 0.7133 - val_accuracy: 0.7644

Epoch 116/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6488
- accuracy: 0.7752 - val_loss: 0.7170 - val_accuracy: 0.7670

Epoch 117/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6471
- accuracy: 0.7756 - val_loss: 0.7210 - val_accuracy: 0.7648

Epoch 118/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6503
- accuracy: 0.7758 - val_loss: 0.7214 - val_accuracy: 0.7616

Epoch 119/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6456
- accuracy: 0.7775 - val_loss: 0.7071 - val_accuracy: 0.7684

Epoch 120/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6435
- accuracy: 0.7765 - val_loss: 0.7245 - val_accuracy: 0.7664

Epoch 121/150

1563/1563 [==============================] - 58s 37ms/step - loss: 0.6562
- accuracy: 0.7714 - val_loss: 0.7289 - val_accuracy: 0.7654

Epoch 122/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6428
- accuracy: 0.7768 - val_loss: 0.7334 - val_accuracy: 0.7700

Epoch 123/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6499
- accuracy: 0.7757 - val_loss: 0.7430 - val_accuracy: 0.7616

Epoch 124/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6537
- accuracy: 0.7756 - val_loss: 0.6992 - val_accuracy: 0.7602

Epoch 125/150

1563/1563 [==============================] - 55s 36ms/step - loss: 0.6507
- accuracy: 0.7718 - val_loss: 0.7262 - val_accuracy: 0.7652

Epoch 126/150
```

```
1563/1563 [==============================] - 55s 35ms/step - loss: 0.6513
- accuracy: 0.7777 - val_loss: 0.7537 - val_accuracy: 0.7578

Epoch 127/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6396
- accuracy: 0.7787 - val_loss: 0.7321 - val_accuracy: 0.7610

Epoch 128/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6536
- accuracy: 0.7746 - val_loss: 0.6964 - val_accuracy: 0.7698

Epoch 129/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6448
- accuracy: 0.7761 - val_loss: 0.7179 - val_accuracy: 0.7602

Epoch 130/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6483
- accuracy: 0.7748 - val_loss: 0.7507 - val_accuracy: 0.7628

Epoch 131/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6490
- accuracy: 0.7751 - val_loss: 0.7255 - val_accuracy: 0.7640

Epoch 132/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6422
- accuracy: 0.7793 - val_loss: 0.7124 - val_accuracy: 0.7648

Epoch 133/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6365
- accuracy: 0.7793 - val_loss: 0.7479 - val_accuracy: 0.7622

Epoch 134/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6437
- accuracy: 0.7765 - val_loss: 0.7230 - val_accuracy: 0.7674

Epoch 135/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6388
- accuracy: 0.7773 - val_loss: 0.7350 - val_accuracy: 0.7658

Epoch 136/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6485
- accuracy: 0.7768 - val_loss: 0.7116 - val_accuracy: 0.7682

Epoch 137/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6454
- accuracy: 0.7761 - val_loss: 0.7386 - val_accuracy: 0.7608
```

```
Epoch 138/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6419
- accuracy: 0.7776 - val_loss: 0.7128 - val_accuracy: 0.7724

Epoch 139/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6470
- accuracy: 0.7767 - val_loss: 0.7074 - val_accuracy: 0.7696

Epoch 140/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6455
- accuracy: 0.7785 - val_loss: 0.7354 - val_accuracy: 0.7696

Epoch 141/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6386
- accuracy: 0.7791 - val_loss: 0.7290 - val_accuracy: 0.7636

Epoch 142/150

1563/1563 [==============================] - 56s 36ms/step - loss: 0.6408
- accuracy: 0.7767 - val_loss: 0.7034 - val_accuracy: 0.7698

Epoch 143/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6386
- accuracy: 0.7814 - val_loss: 0.7478 - val_accuracy: 0.7506

Epoch 144/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6414
- accuracy: 0.7772 - val_loss: 0.7752 - val_accuracy: 0.7596

Epoch 145/150

1563/1563 [==============================] - 53s 34ms/step - loss: 0.6367
- accuracy: 0.7801 - val_loss: 0.7173 - val_accuracy: 0.7716

Epoch 146/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6398
- accuracy: 0.7770 - val_loss: 0.7681 - val_accuracy: 0.7548

Epoch 147/150

1563/1563 [==============================] - 54s 34ms/step - loss: 0.6419
- accuracy: 0.7774 - val_loss: 0.7256 - val_accuracy: 0.7652

Epoch 148/150

1563/1563 [==============================] - 54s 35ms/step - loss: 0.6381
- accuracy: 0.7776 - val_loss: 0.7199 - val_accuracy: 0.7716

Epoch 149/150
```

```
1563/1563 [==============================] - 54s 34ms/step - loss: 0.6370
- accuracy: 0.7787 - val_loss: 0.7350 - val_accuracy: 0.7672

Epoch 150/150

1563/1563 [==============================] - 55s 35ms/step - loss: 0.6374
- accuracy: 0.7793 - val_loss: 0.7106 - val_accuracy: 0.7642
```