МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №9**

з дисципліни «**Нейронні мережі**»

# на тему: «ОСНОВИ TRANSFER LEARNING»

**Виконав:**

студент гр. БС-03

Затуловський Г. А.

**Перевірив:**

ас. каф. БМК Дюмін О.Д.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2023

# Варіант 6

# Практична частина

1. Дослідження наборів даних: виберіть датасет відповідно до варіанту завдання.

2. Проведіть первинний аналіз даних: розподіл класів, кількість зображень у кожному класі тощо.

3. Попередня обробка: здійсніть попередню обробку зображень: масштабування, аугментація (збільшення даних шляхом випадкових перетворень зображення) і поділ на тренувальні та тестові датасети.

4. Моделювання: завантажте попередньо навчену модель з Keras.

5. Здійсніть адаптацію моделі до вашого набору даних шляхом зміни верхніх шарів мережі.

6. Навчіть модель, використовуючи техніку передачі навчання (Feature extraction або Fine-tuning).

7. Оцінка: протестуйте вашу модель на тестовому наборі даних, обчисліть різні метрики якості (точність, відгук, F1-score тощо).

8. Візуалізація: візуалізуйте криві навчання (точність і втрати на тренувальних та валідаційних датасетах під час ітерацій).

9. Висновки: сформулюйте висновки з приводу ефективності використаної стратегії передачі навчання для вашого конкретного набору даних. Запропонуйте можливі підходи до поліпшення результатів.

10. Додаткове завдання (необов'язкове): спробуйте інші попередньо навчені моделі і порівняйте їх результати.

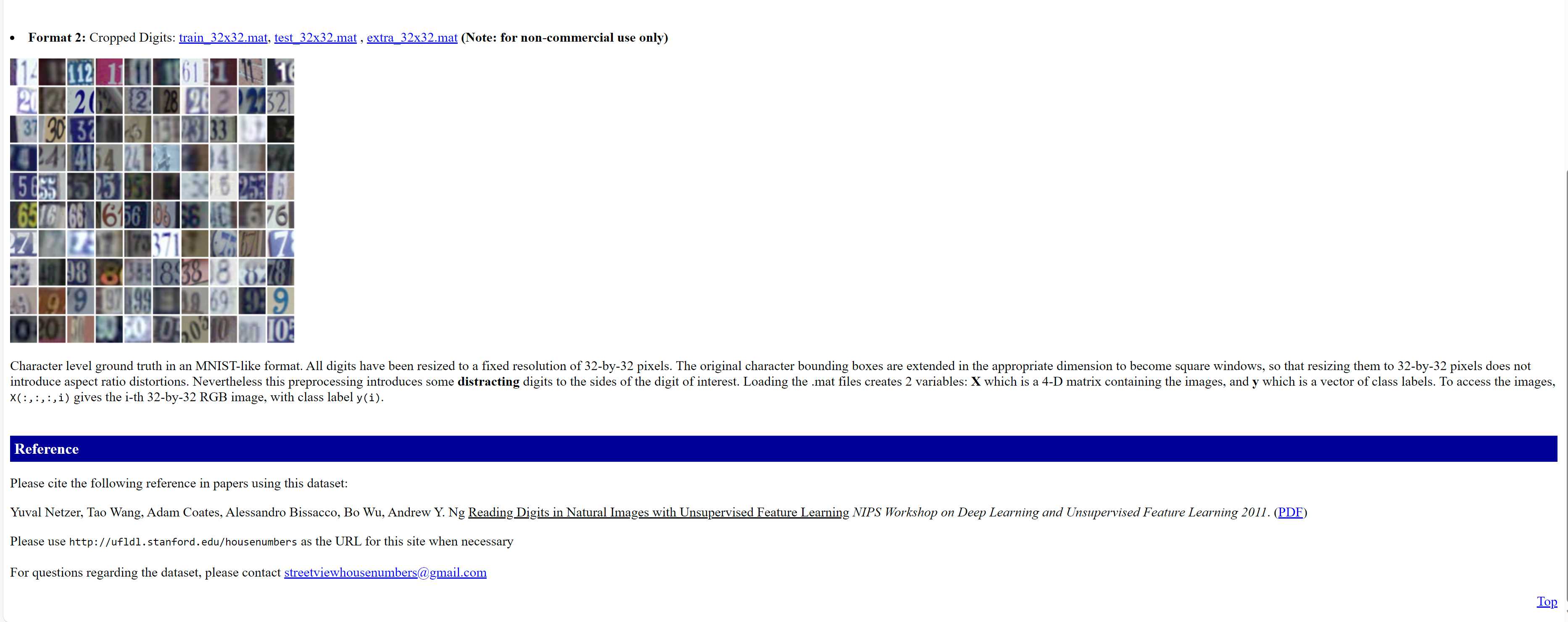
Варіант

[2, **6**, 10, 14, 18, 22] SVHN (Street View House Numbers):

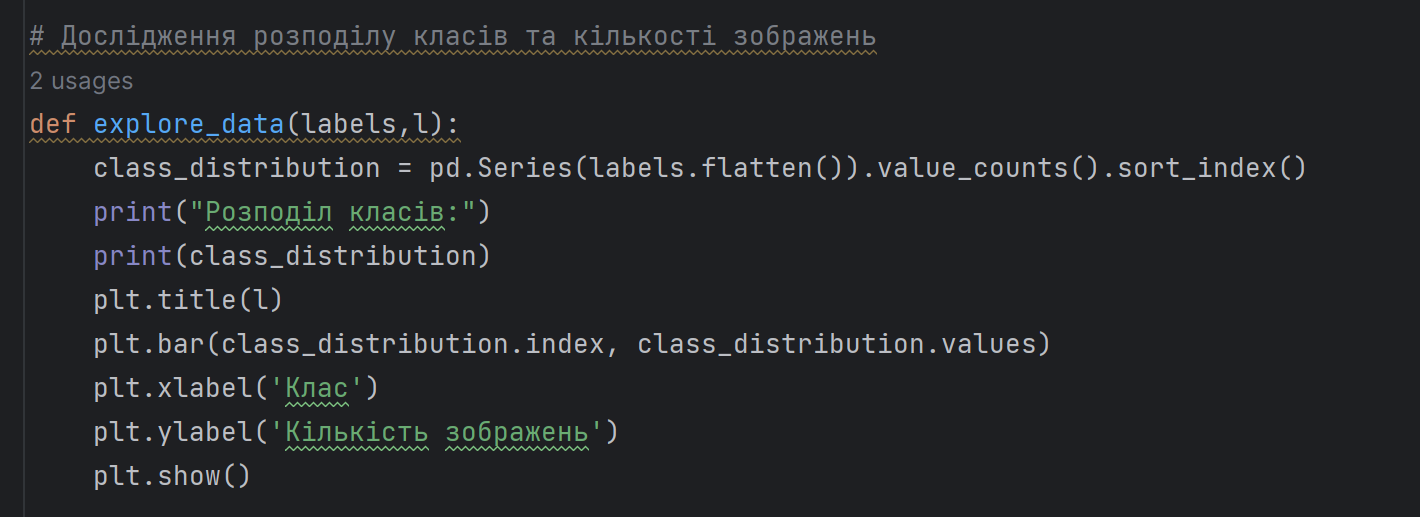
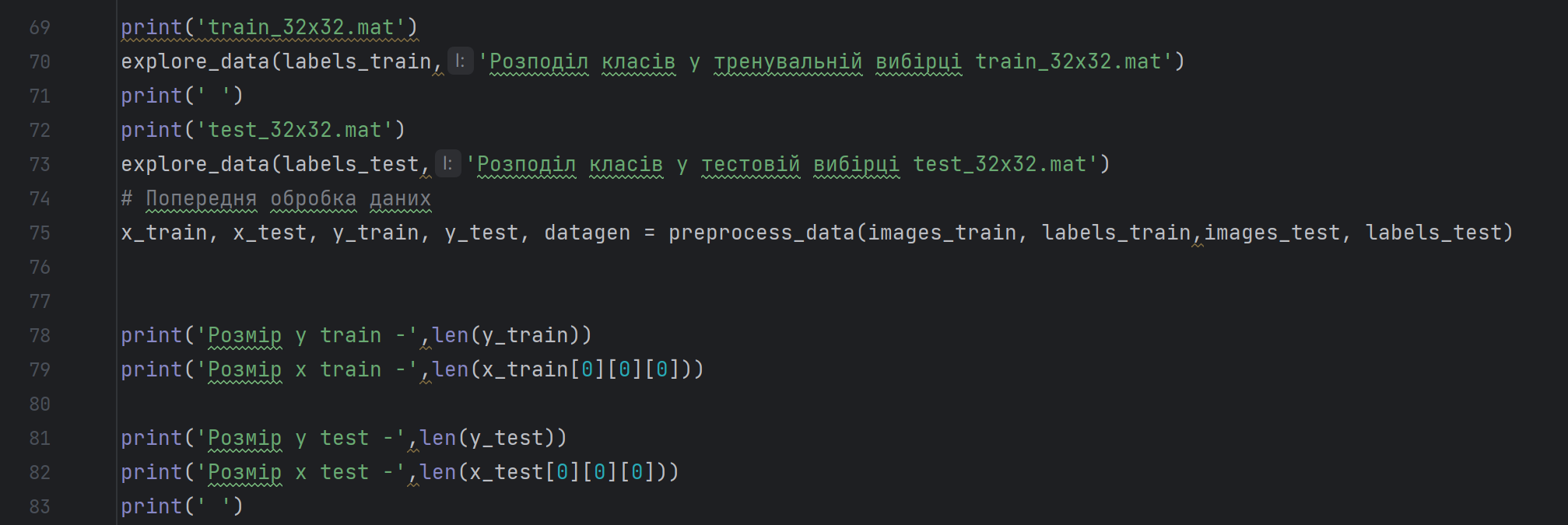
Характеристики: Цей набір даних містить більше 600 000 реальних зображень номерів будинків, взятих із Google Street View. Скачати: <http://ufldl.stanford.edu/housenumbers/>

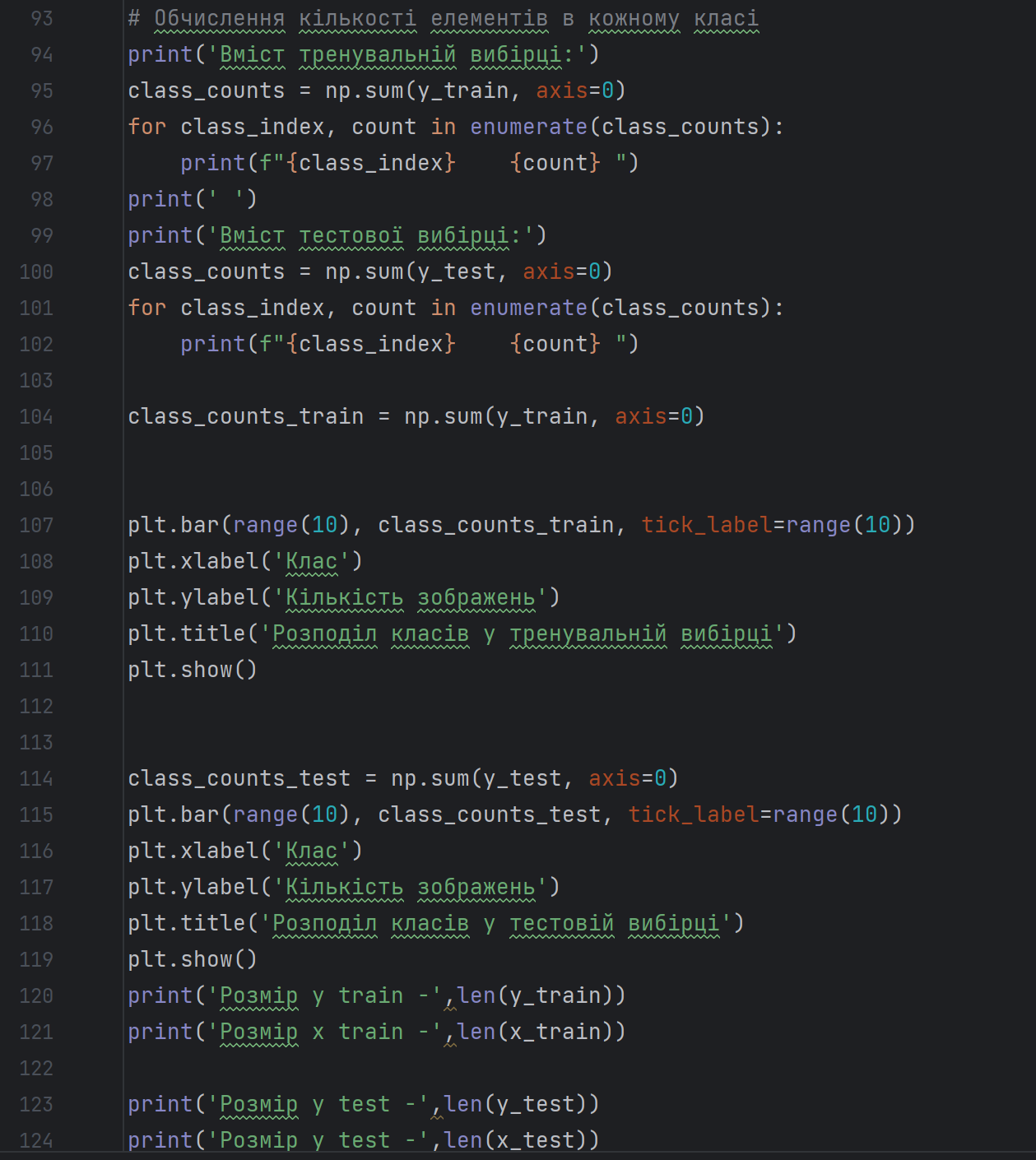
**Хід роботи:**

Завантаження даних з сайту

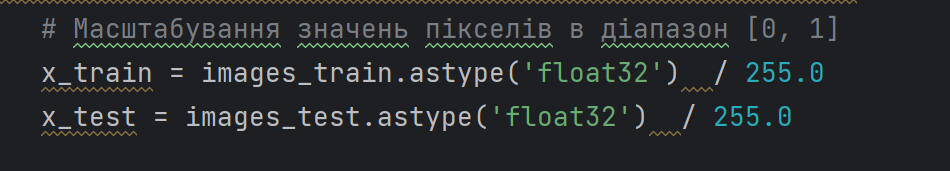
****

Проведемо первиний аналіз

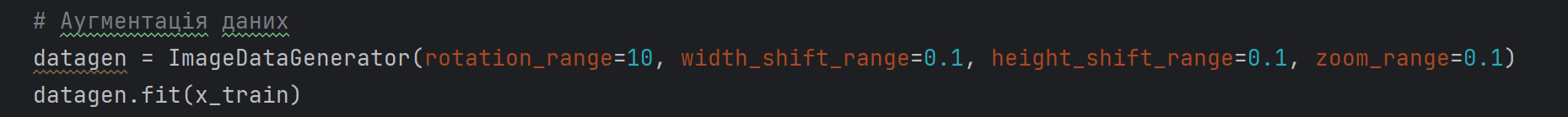
 



Масштабування

****

Аугментація

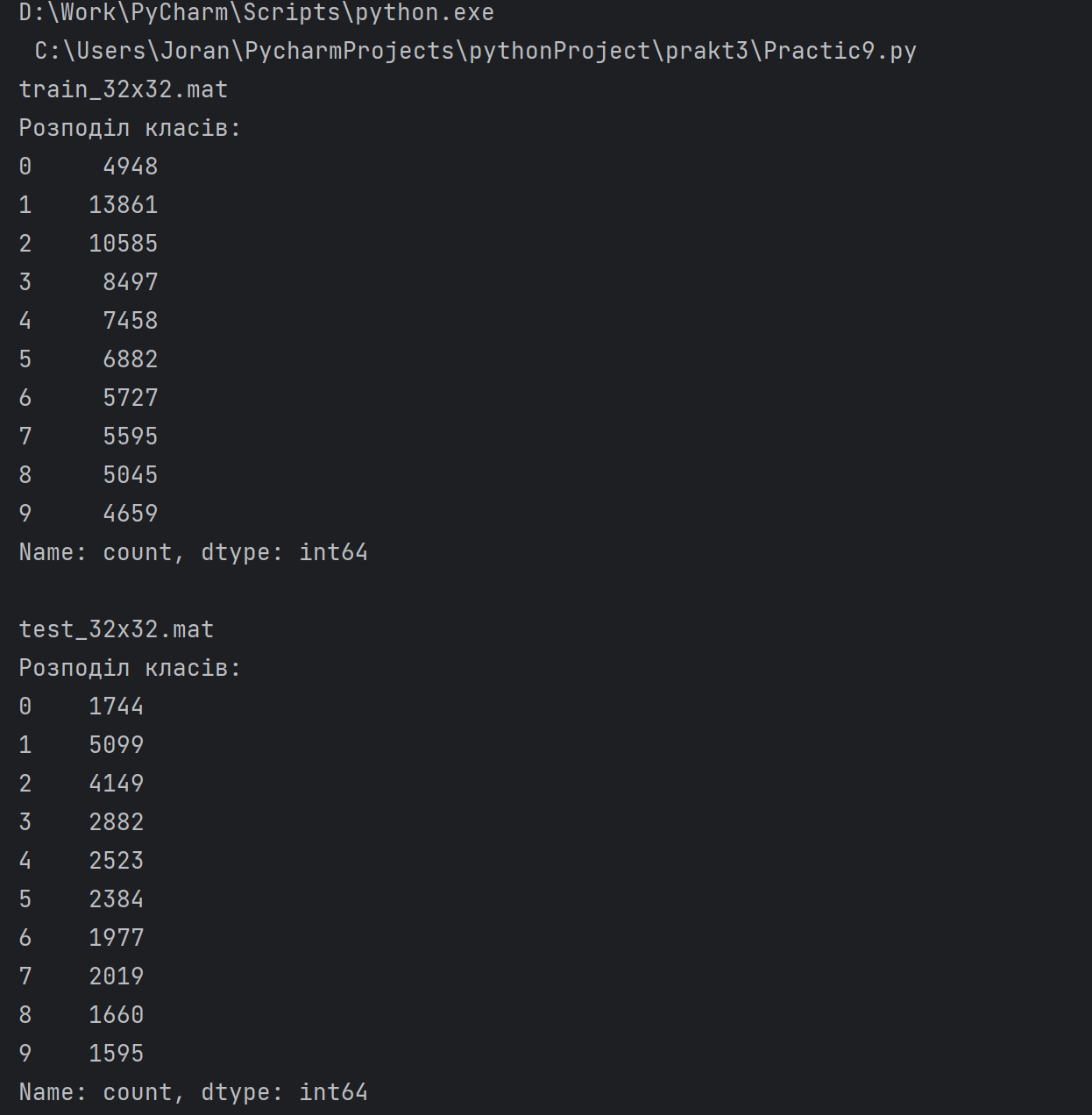


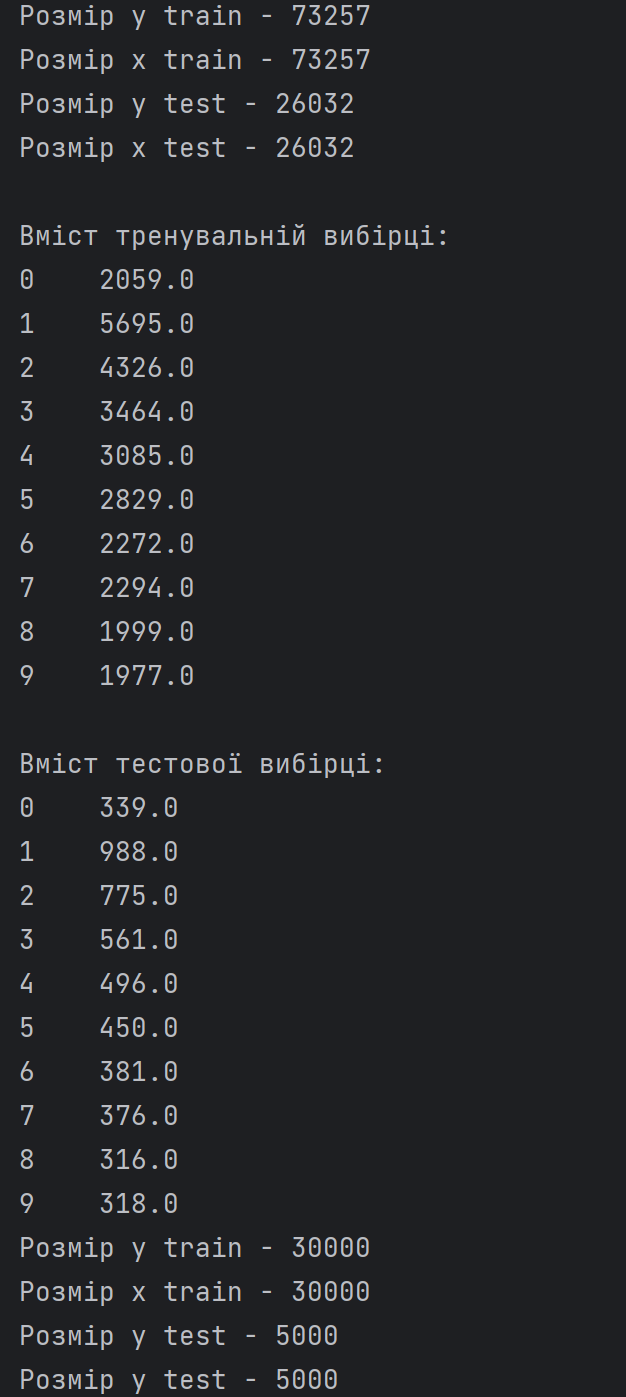
Використовуємо налаштування налаштування Feature extraction та завантажемо попередню навчену модель VGG16

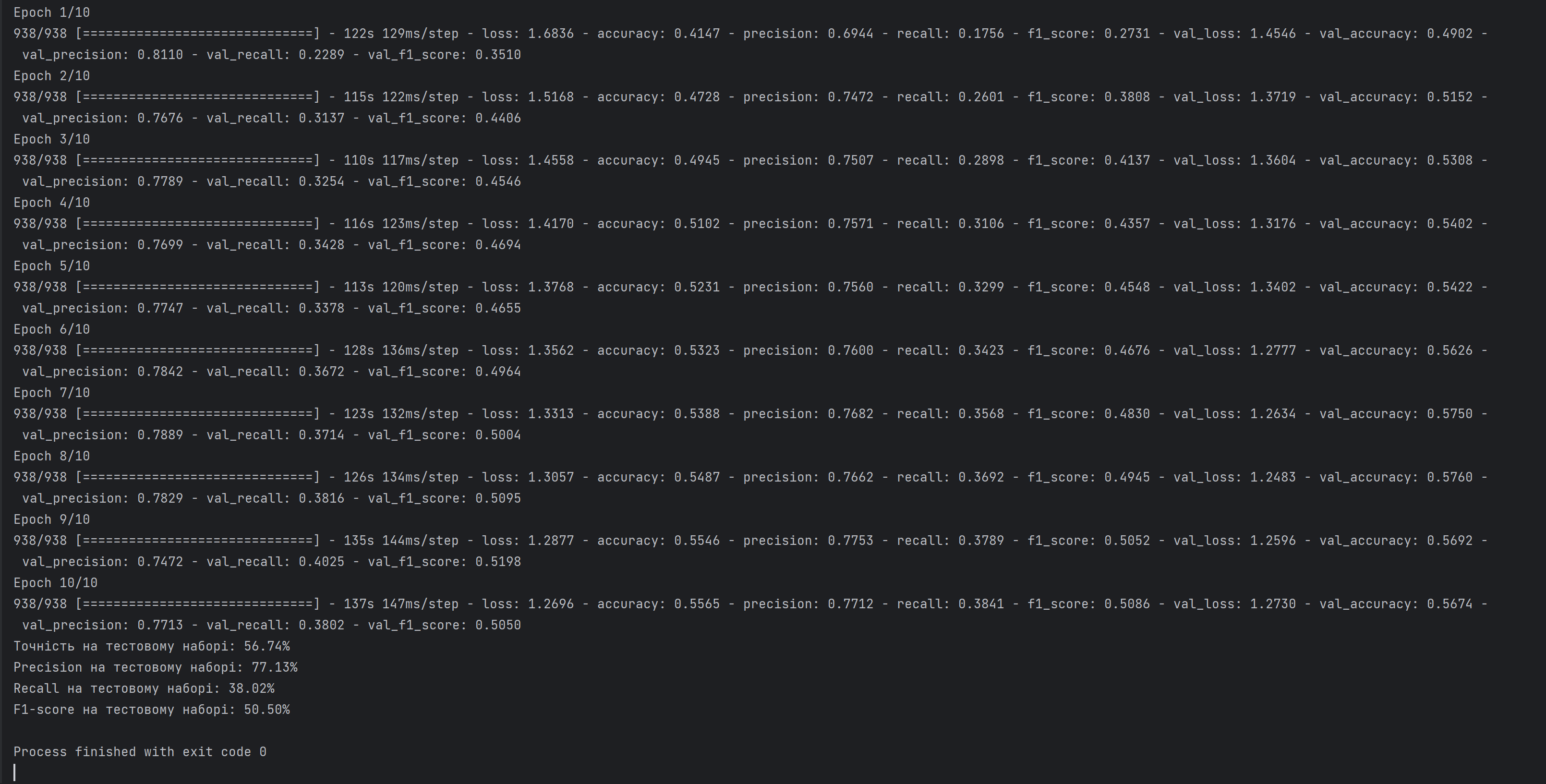
**Код програми:**

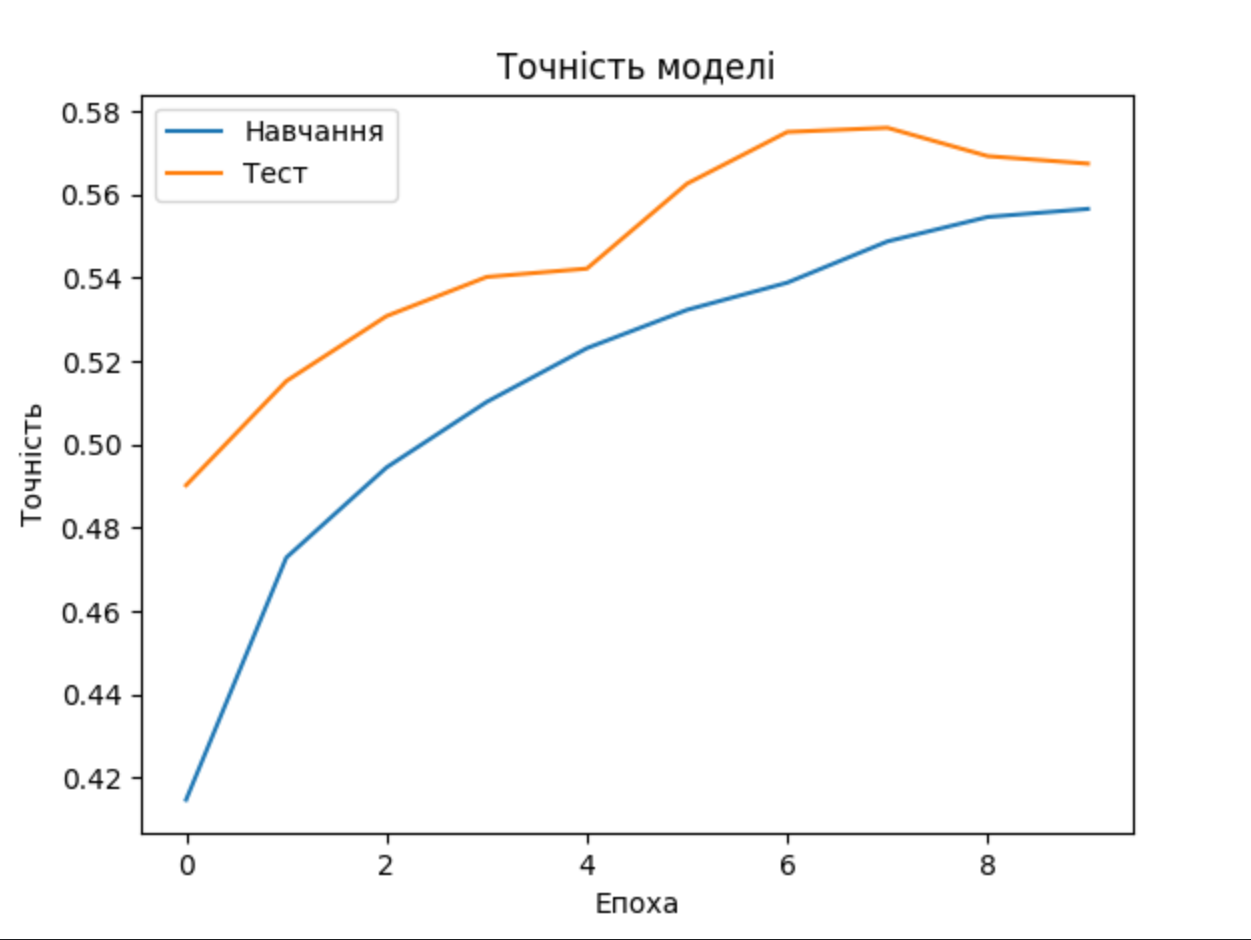
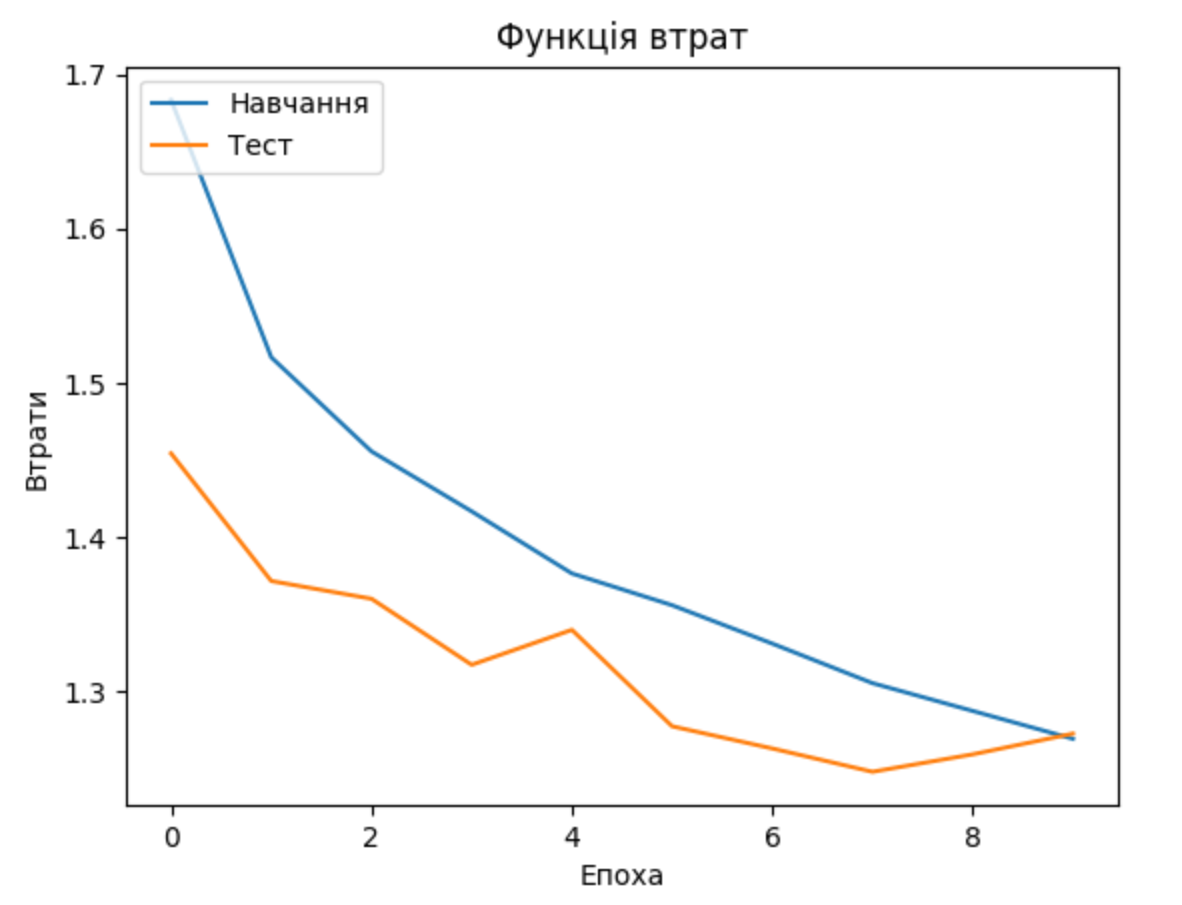
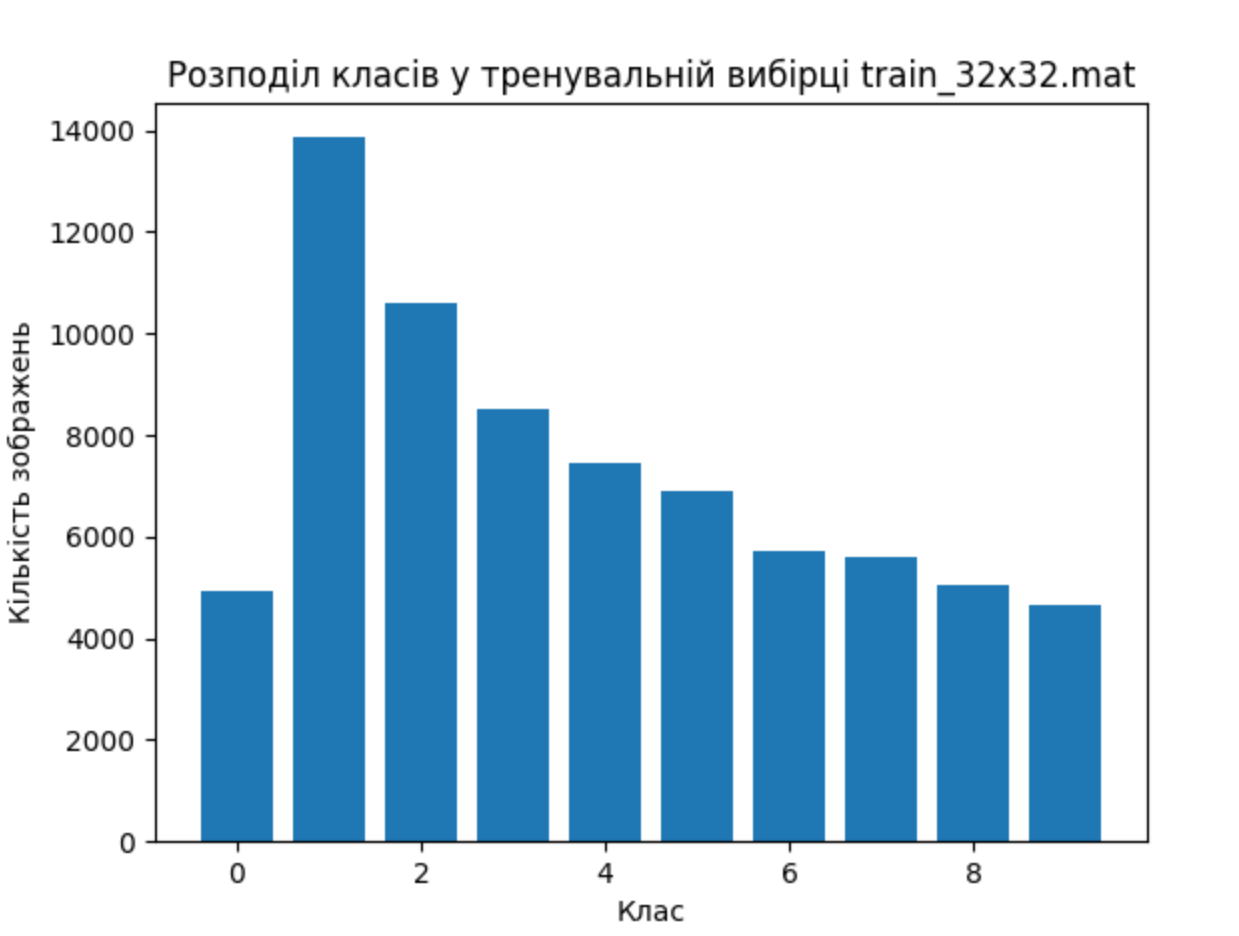
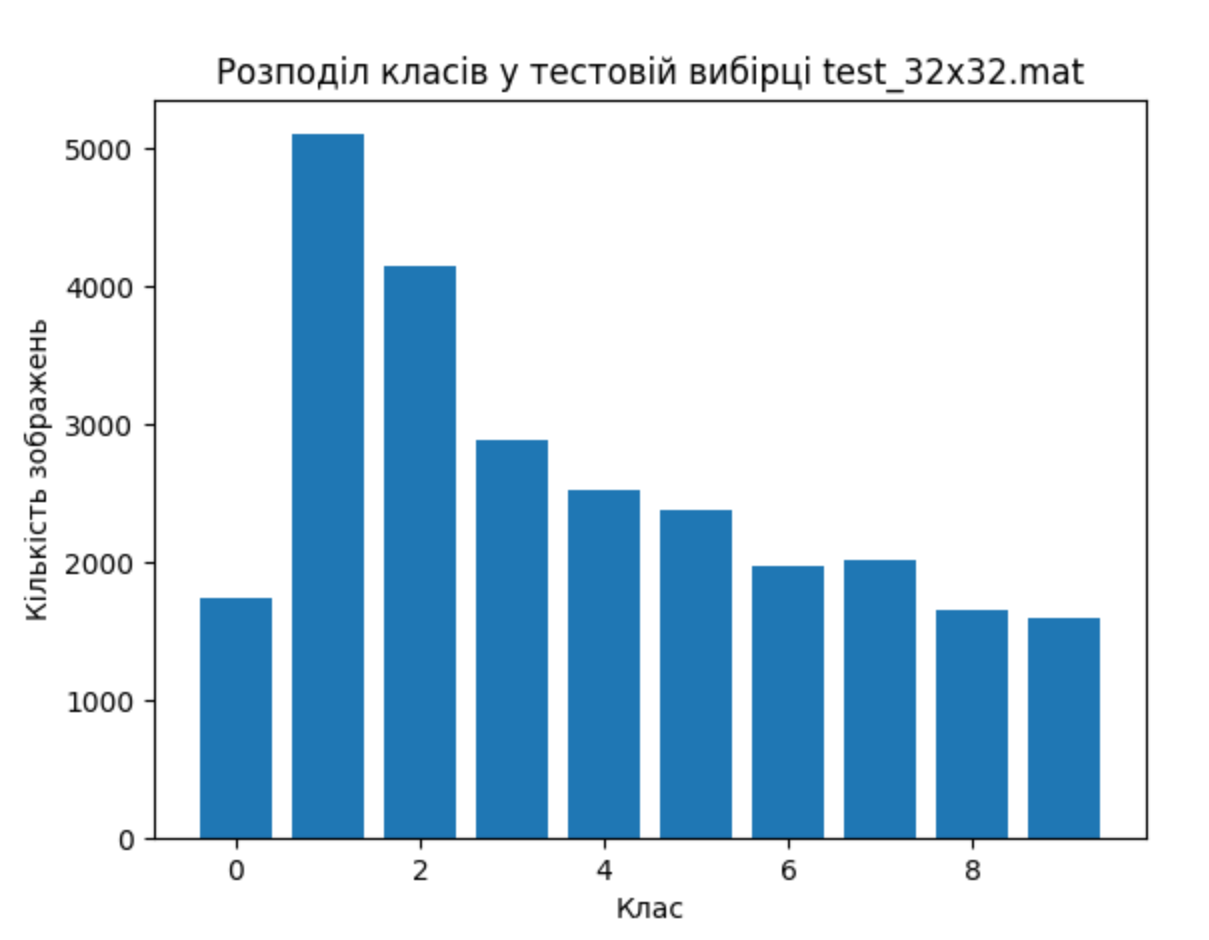
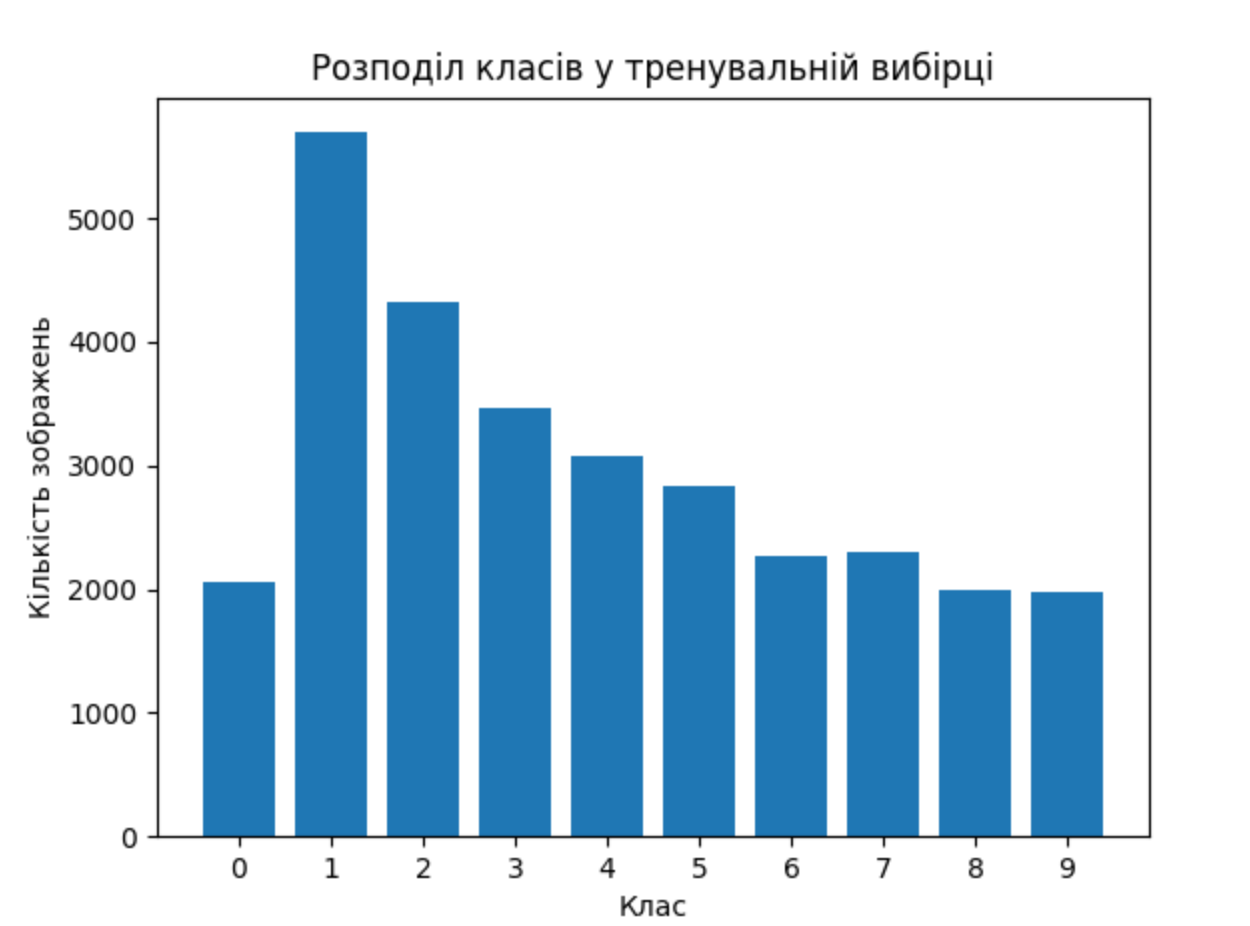
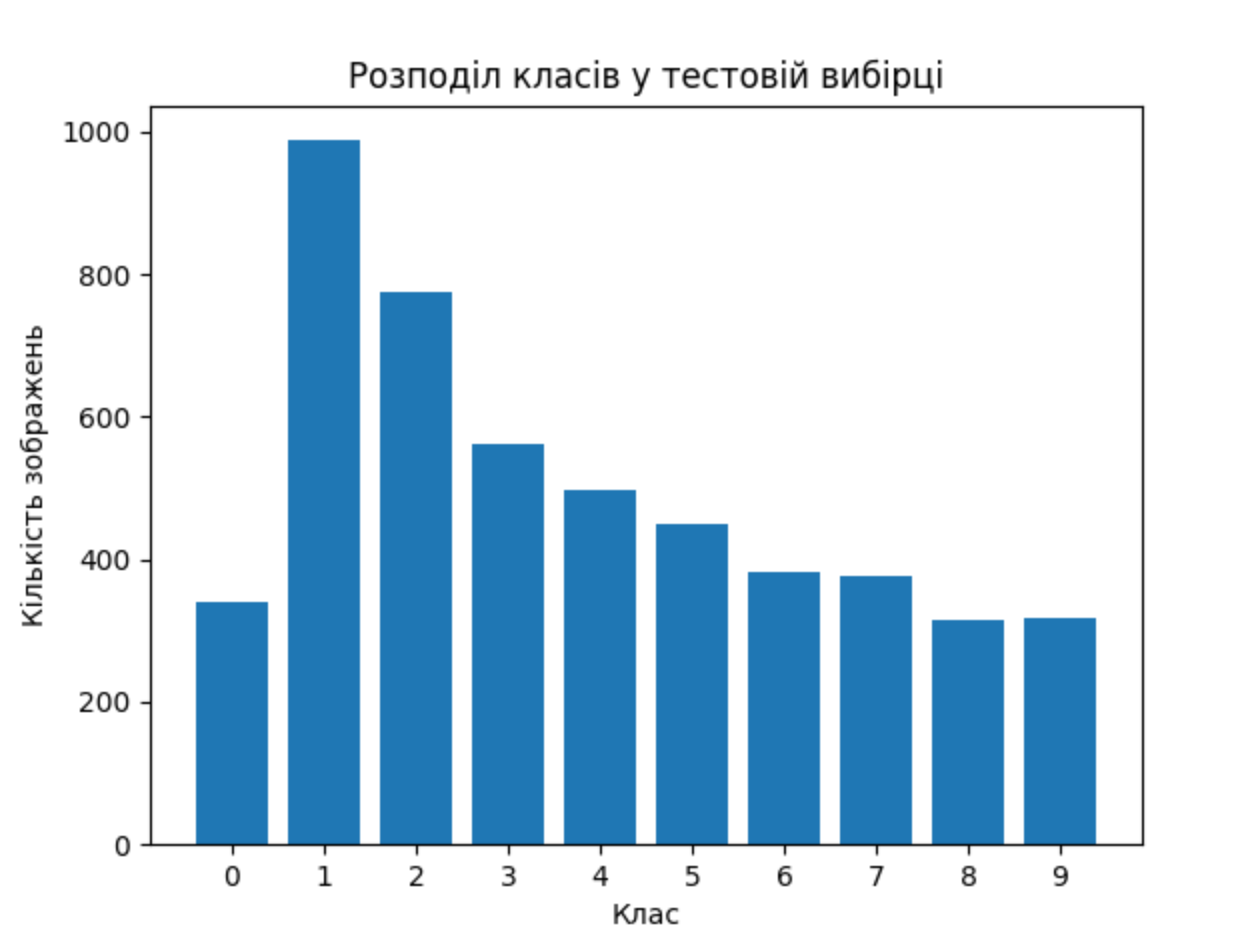
mport numpy as np  
import pandas as pd  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from tensorflow.keras.utils import to\_categorical  
from scipy.io import loadmat  
from keras.applications import VGG16  
import matplotlib.pyplot as plt  
from keras import backend as K  
from keras.models import Model  
from keras.layers import Dense, Flatten  
  
  
# Завантаження датасету SVHN  
def load\_svhn\_data():  
 svhn\_train = loadmat('train\_32x32.mat')  
 images\_train = svhn\_train['X']  
 labels\_train = svhn\_train['y']  
 svhn\_test = loadmat('test\_32x32.mat')  
 images\_test = svhn\_test['X']  
 labels\_test = svhn\_test['y']  
 # Заміна міток 10 на 0  
 labels\_test[labels\_test == 10] = 0  
 labels\_train[labels\_train == 10] = 0  
  
  
  
 return images\_train, labels\_train,images\_test, labels\_test  
  
  
  
  
  
# Дослідження розподілу класів та кількості зображень  
def explore\_data(labels,l):  
 class\_distribution = pd.Series(labels.flatten()).value\_counts().sort\_index()  
 print("Розподіл класів:")  
 print(class\_distribution)  
 plt.title(l)  
 plt.bar(class\_distribution.index, class\_distribution.values)  
 plt.xlabel('Клас')  
 plt.ylabel('Кількість зображень')  
 plt.show()  
  
# Попередня обробка даних  
def preprocess\_data(x\_train, y\_train, x\_test , y\_test):  
 # Масштабування значень пікселів в діапазон [0, 1]  
 x\_train = images\_train.astype('float32') / 255.0  
 x\_test = images\_test.astype('float32') / 255.0  
  
  
 # Кодування міток у формат one-hot  
 y\_train = to\_categorical(y\_train, num\_classes=10)  
 y\_test = to\_categorical(y\_test, num\_classes=10)  
  
 # Аугментація даних  
 datagen = ImageDataGenerator(rotation\_range=10, width\_shift\_range=0.1, height\_shift\_range=0.1, zoom\_range=0.1)  
 datagen.fit(x\_train)  
  
  
 return x\_train, x\_test, y\_train, y\_test, datagen  
  
# Завантаження та дослідження даних  
images\_train, labels\_train,images\_test, labels\_test = load\_svhn\_data()  
  
  
  
  
  
print('train\_32x32.mat')  
explore\_data(labels\_train,'Розподіл класів у тренувальній вибірці train\_32x32.mat')  
print(' ')  
print('test\_32x32.mat')  
explore\_data(labels\_test,'Розподіл класів у тестовій вибірці test\_32x32.mat')  
# Попередня обробка даних  
x\_train, x\_test, y\_train, y\_test, datagen = preprocess\_data(images\_train, labels\_train,images\_test, labels\_test)  
  
  
print('Розмір y train -',len(y\_train))  
print('Розмір x train -',len(x\_train[0][0][0]))  
  
print('Розмір y test -',len(y\_test))  
print('Розмір x test -',len(x\_test[0][0][0]))  
print(' ')  
x\_train = np.moveaxis(x\_train, -1, 0)  
x\_test = np.moveaxis(x\_test, -1, 0)  
  
  
  
x\_train = x\_train[:30000]  
x\_test = x\_test[:5000]  
y\_train = y\_train[:30000]  
y\_test = y\_test[:5000]  
# Обчислення кількості елементів в кожному класі  
print('Вміст тренувальній вибірці:')  
class\_counts = np.sum(y\_train, axis=0)  
for class\_index, count in enumerate(class\_counts):  
 print(f"{class\_index} {count} ")  
print(' ')  
print('Вміст тестової вибірці:')  
class\_counts = np.sum(y\_test, axis=0)  
for class\_index, count in enumerate(class\_counts):  
 print(f"{class\_index} {count} ")  
  
class\_counts\_train = np.sum(y\_train, axis=0)  
  
  
plt.bar(range(10), class\_counts\_train, tick\_label=range(10))  
plt.xlabel('Клас')  
plt.ylabel('Кількість зображень')  
plt.title('Розподіл класів у тренувальній вибірці')  
plt.show()  
  
  
class\_counts\_test = np.sum(y\_test, axis=0)  
plt.bar(range(10), class\_counts\_test, tick\_label=range(10))  
plt.xlabel('Клас')  
plt.ylabel('Кількість зображень')  
plt.title('Розподіл класів у тестовій вибірці')  
plt.show()  
print('Розмір y train -',len(y\_train))  
print('Розмір x train -',len(x\_train))  
  
print('Розмір y test -',len(y\_test))  
print('Розмір y test -',len(x\_test))  
  
  
base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(32, 32, 3))  
  
# Додавання нових верхніх шарів  
x = base\_model.output  
x = Flatten()(x)  
x = Dense(512, activation='relu')(x)  
predictions = Dense(10, activation='softmax')(x)  
# З'єднання базової моделі та нових верхніх шарів  
model = Model(inputs=base\_model.input, outputs=predictions)  
# Заморожування ваг базової моделі  
for layer in base\_model.layers:  
 layer.trainable = False  
# Компіляція моделі  
  
  
  
def precision(y\_true, y\_pred):  
  
 true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))  
 predicted\_positives = K.sum(K.round(K.clip(y\_pred, 0, 1)))  
 precision = true\_positives / (predicted\_positives + K.epsilon())  
 return precision  
  
  
def recall(y\_true, y\_pred):  
  
 true\_positives = K.sum(K.round(K.clip(y\_true \* y\_pred, 0, 1)))  
 possible\_positives = K.sum(K.round(K.clip(y\_true, 0, 1)))  
 recall = true\_positives / (possible\_positives + K.epsilon())  
 return recall  
  
  
def f1\_score(y\_true, y\_pred):  
 precision\_val = precision(y\_true, y\_pred)  
 recall\_val = recall(y\_true, y\_pred)  
 return 2 \* ((precision\_val \* recall\_val) / (precision\_val + recall\_val + K.epsilon()))  
  
  
model.compile(optimizer='adam', loss='categorical\_crossentropy',  
 metrics=['accuracy', precision, recall, f1\_score])  
history = model.fit(datagen.flow(x\_train, y\_train, batch\_size=32),epochs=10,validation\_data=(x\_test, y\_test) )  
# Графік точності  
plt.plot(history.history['accuracy'])  
plt.plot(history.history['val\_accuracy'])  
plt.title('Точність моделі')  
plt.ylabel('Точність')  
plt.xlabel('Епоха')  
plt.legend(['Навчання', 'Тест'], loc='upper left')  
plt.show()  
# Графік функції втрат  
plt.plot(history.history['loss'])  
plt.plot(history.history['val\_loss'])  
plt.title('Функція втрат')  
plt.ylabel('Втрати')  
plt.xlabel('Епоха')  
plt.legend(['Навчання', 'Тест'], loc='upper left')  
plt.show()  
  
print(f"Точність на тестовому наборі: {history.history['val\_accuracy'][-1]\*100:.2f}%")  
print(f"Precision на тестовому наборі: {history.history['val\_precision'][-1]\*100:.2f}%")  
print(f"Recall на тестовому наборі: {history.history['val\_recall'][-1]\*100:.2f}%")  
print(f"F1-score на тестовому наборі: {history.history['val\_f1\_score'][-1]\*100:.2f}%")

**Результати:**

****

****

****

****     

Розглянувши результати можна зробити висновок що навчання нашої нейронної мережі з використанням Transfer Learning з використуванням наших даних є не дуже точною (на тестовому наборі - accuracy 56.74%; precision 77.14%; recall 38.02%; F1-score 50.50%), це може свідчити про кілька можливих проблем:

1. Невідповідність завданням
2. Недостатній обсяг нових даних
3. Неправильне розмороження шарів
4. Неправильні параметри Feature extraction

Щоб покращити результат можна використати кілька методів:

1. Збільште обсяг даних
2. Зміними параметри аугментації даних
3. Призначте достатньо часу для Feature extraction
4. Експериментуйте з архітектурою моделі
5. Оптимізація параметрів Feature extraction
6. Використовуйте діагностику моделі

**Контрольні питання**

1. **Що таке Transfer Learning?**

Transfer Learning (передача навчання) – це методика машинного навчання, де модель, розроблена для одного завдання, використовується як вихідний пункт для моделі, розробленої для іншого (зазвичай пов'язаного) завдання. Це відбувається завдяки перевикористанню основної частини моделі та адаптації окремих частин під конкретне завдання.

1. **Які основні стратегії передачі навчання ви знаєте?**

Основні стратегії передачі навчання включають:

1. Повне використання (Full Network Transfer):

Всю навчену нейронну мережу використовують в новій задачі без змін.

Застосовується, коли ресурси для навчання обмежені, а задачі схожі.

1. Фінетюнінг (Fine-tuning):

Попередньо навчена модель (наприклад, на ImageNet) частково перенавчається на нових даних для нового завдання. Зазвичай заморожують попередньо навчені шари і налаштовують лише кілька верхніх шарів для нового завдання.

1. Використання як екстрактора ознак (Feature Extraction):

Використовують попередньо навчену модель як екстрактор ознак для отримання векторів ознак, які потім використовуються для нової задачі. Зазвичай заморожують ваги всіх або певних нижніх шарів.

1. Стекінг (Stacking):

Об'єднання попередньо навчених моделей з додатковими шарами для розв'язання конкретного завдання. Може включати модифікації структури мережі для об'єднання знань з різних джерел.

1. Синергетична передача навчання (Synergistic Transfer):

Об'єднання навчання на декількох задачах для використання синергії між ними для покращення результатів від навчання на конкретній задачі.

1. Обернена передача (Backward Transfer):

Знання, отримане від навчання на новій задачі, використовується для покращення продуктивності на попередньому завданні.

1. **В чому переваги використання Transfer Learning порівняно з навчанням моделі "з нуля"?**

Переваги використання Transfer Learning порівняно з навчанням моделі "з нуля":

1. Зменшення витрат на навчання:

Перенесення ваг моделі, навченої на великому наборі даних, може значно зменшити час і ресурси, необхідні для навчання нової моделі. Велика частина "знань" вже закладена в передбаченнях, отриманих під час попереднього навчання.

1. Покращення продуктивності:

Модель, навчена на великому наборі даних, вже має загальне розуміння природних закономірностей, які можуть бути корисними в інших завданнях. Це може покращити загальні робочі характеристики моделі в порівнянні з моделлю, яка навчалася тільки на обмеженому наборі даних.

1. Ліпше узагальнення:

Модель, яка навчалася на великому наборі даних, зазвичай має кращі уміння узагальнювати в нових ситуаціях. Це особливо важливо, коли доступні обмежені дані для нового завдання.

1. Менше даних для навчання:

Для використання Transfer Learning не завжди потрібно велику кількість даних для нового завдання. Навіть з невеликою кількістю даних можна досягти хороших результатів, використовуючи передбачення моделі, навченої на великому наборі даних.

1. Застосування до різноманітних завдань:

Модель, яка була навчена на одному типі завдань (наприклад, розпізнавання об'єктів на зображеннях), може бути успішно використана для різних завдань (наприклад, визначення сентименту в тексті), за умови правильного перенесення знань**.**

1. **Як вибрати, які шари попередньо навченої моделі слід "розморозити" для Fine-tuning?**

Вибір шарів для розморожування під час Fine-tuning є критичним етапом, і від нього залежить ефективність процесу навчання. Ось кілька загальних порад щодо вибору шарів для розморозки:

1. Верхні шари: Зазвичай розморожують лише верхні шари попередньо навченої моделі. Ці шари відповідають за вищі рівні абстракції та більш загальні ознаки, які можуть бути корисні в новому завданні.
2. Загальні риси та відділення: Розмороження шарів, які відповідають за загальні риси, такі як краї та форми, може бути корисним. Однак шари, які специфічні для попереднього завдання (яке було вирішено попередньо навченою моделлю), можуть залишатися "замороженими".
3. Аналіз архітектури моделі: Аналізуйте архітектуру попередньо навченої моделі і визначте, які шари відповідають за загальні та специфічні ознаки. Зазвичай, перші шари відповідають за загальні ознаки, а останні шари - за більш деталізовані.
4. Уникайте перенавчання: Уникайте розмороження занадто багато шарів, оскільки це може призвести до перенавчання на обмеженому обсязі mаних. Зазвичай декілька верхніх шарів буде достатньо.
5. Розглядайте адаптацію швидкості навчання: Розгляньте використання різних швидкостей навчання для розморожених і "заморожених" шарів. Розморожені шари можуть вимагати меншої швидкості навчання, оскільки вони можуть зазнавати більших змін.
6. Експериментуйте і слідкуйте за метриками: Експериментуйте з різними конфігураціями розморожених шарів і спостерігайте за метриками навчання та валідації. Вибирайте ту конфігурацію, яка працює найкраще для вашого конкретного завдання.

**5. Чи є обмеження або ситуації, коли Transfer Learning може бути неефективним?**

Так, існують обмеження та ситуації, коли Transfer Learning може бути неефективним або менше ефективним. Ось деякі з них:

1. Не завжди корисний: Якщо джерельний і цільовий датасети занадто відрізняються, передача навчання може не принести бажаного ефекту.
2. Попередній досвід: Інколи попередньо навчена модель може внести певний "спотворений" вигляд даних, який може бути не доцільним для нової задачі.
3. Обчислювальні витрати: Деякі попередньо навчені моделі можуть бути великими і вимагати значних обчислювальних ресурсів, навіть якщо вони не повністю використовуються.
4. Гнучкість: Ви обмежені архітектурою попередньо навченої моделі, хоча і можна модифікувати її кінцеві шари.