МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №10**

з дисципліни «**Нейронні мережі**»

# на тему: «РОЗРОБКА РЕКУРЕНТНОЇ НЕЙРОННОЇ МЕРЕЖІ»

**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.Вивчення датасету: завантажте датасет відповідно до варіанту завдання. Проведіть його аналіз: розподіл категорій, кількість слів у текстах, часті слова.

2. Попередня обробка тексту:

• Токенізуйте (tokenize) текст: перетворіть слова в цілочисельні ідентифікатори. • Проведіть padding послідовностей до однакової довжини.

• Розділіть датасет на навчальний та тестовий піднабори.

3. Моделювання:

• Створіть модель, використовуючи рекурентні шари (LSTM, GRU чи BiRNN).

• Додайте Dropouts для запобігання перенавчанню.

• Визначтеся із кількістю нейронів у шарах та іншими гіперпараметрами.

4. Тренування та оцінка:

• Навчіть модель на навчальному датасеті.

• Оцініть якість моделі на тестовому датасеті, використовуючи такі метрики як точність, F1-метрика та інші.

• Візуалізуйте графіки втрат та точності в процесі навчання.

5. Оптимізація:

• Застосуйте різні методи оптимізації моделі: змінюйте архітектуру, гіперпараметри, додайте регуляризацію тощо.

• Порівняйте результати отриманих моделей.

**Хід роботи:**

[2, 6, 10, 14, 18, 22] MovieLens Sentiment Dataset: Цей набір даних містить текстові відгуки користувачів про фільми, аналогічно IMDb. Він може бути використаний для задач аналізу настрою. Датасет можна знайти тут: <https://www.kaggle.com/competitions/sentiment-analysis-on-movie-reviews/data>.

Набір даних складається з розділених табуляціями файлів із фразами з набору даних Rotten Tomatoes. Поділ train/test було збережено для цілей порівняльного аналізу, але речення перемішано з початкового порядку. Кожне речення було розібрано на багато фраз Стенфордським аналізатором. Кожна фраза має PhraseId. Кожне речення має SentenceId. Фрази, які повторюються (наприклад, короткі/поширені слова), включаються до даних лише один раз.

train.tsv містить фрази та пов’язані з ними позначки настроїв. Ми додатково надали SentenceId, щоб ви могли відстежувати, які фрази належать до одного речення.

test.tsv містить лише фрази.

Ви повинні призначити мітку настрою кожній фразі.

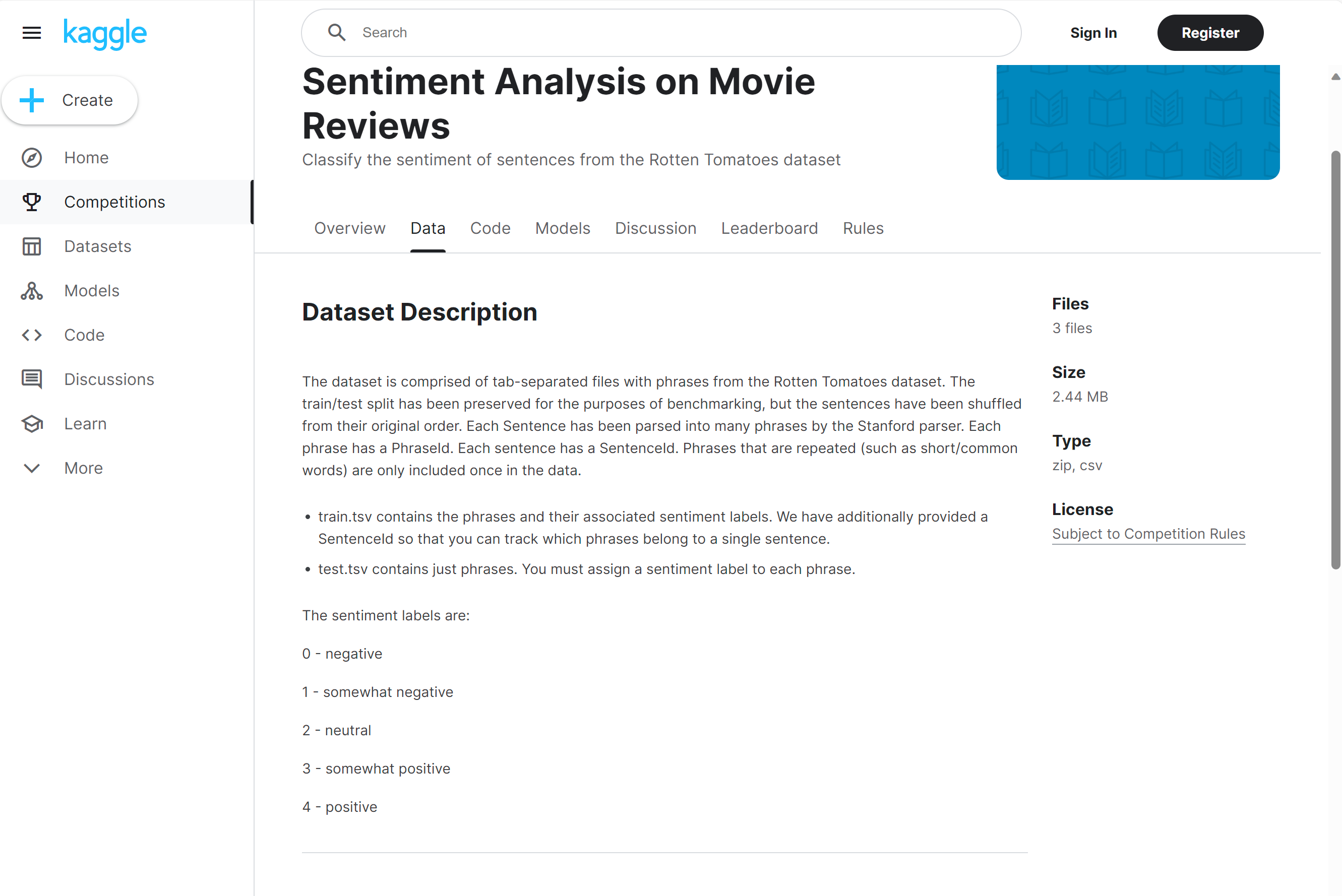
Мітки настроїв:

0 - негативний

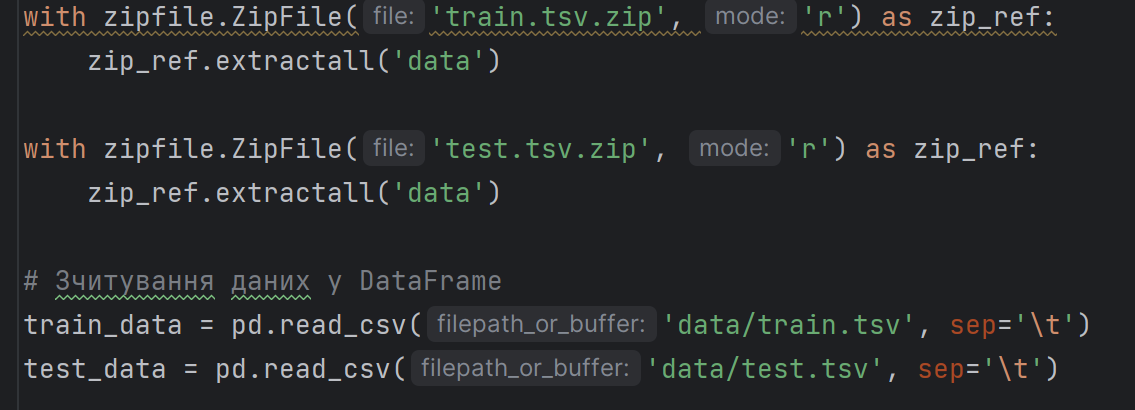
1 - дещо негативно

2 - нейтральний

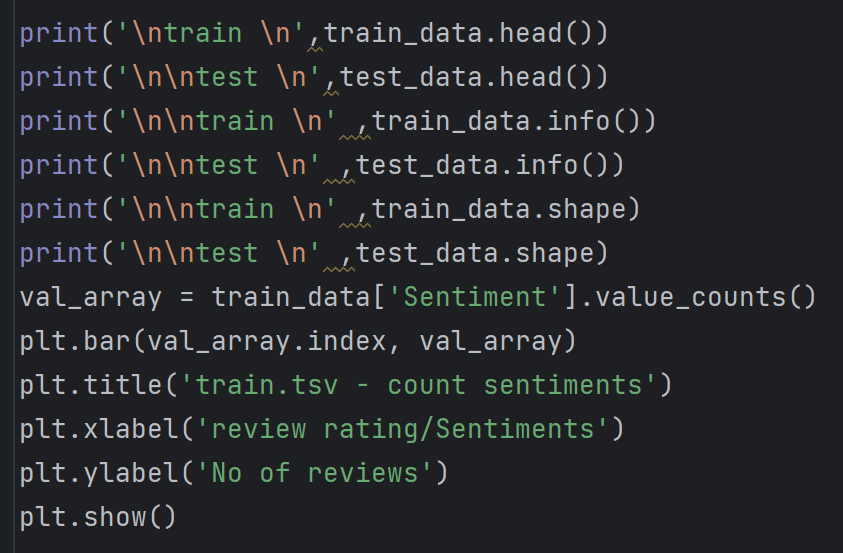
3 - певною мірою позитивно

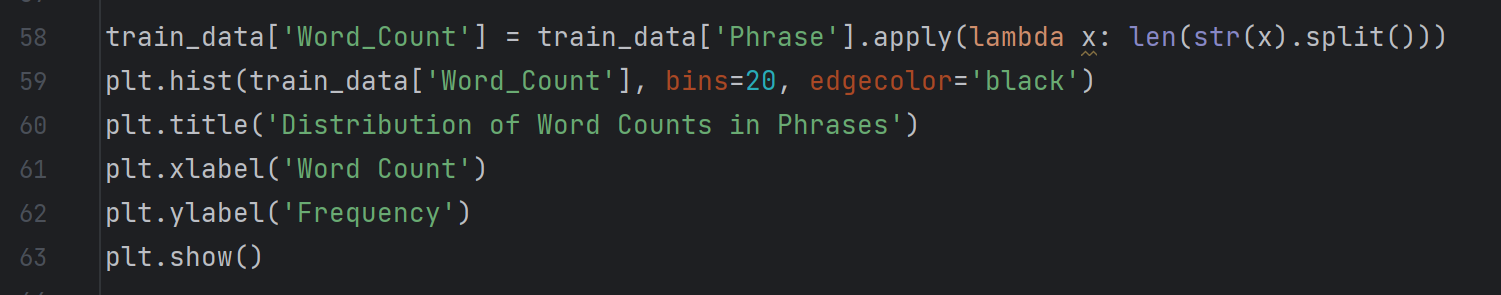
4 - позитивний****

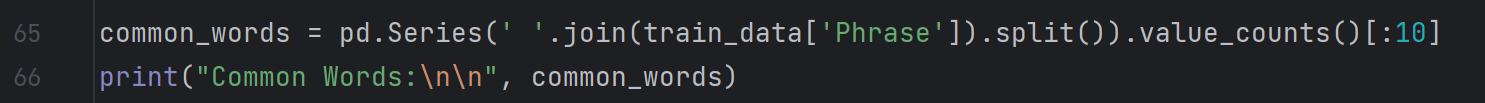
Завантажемо датасет



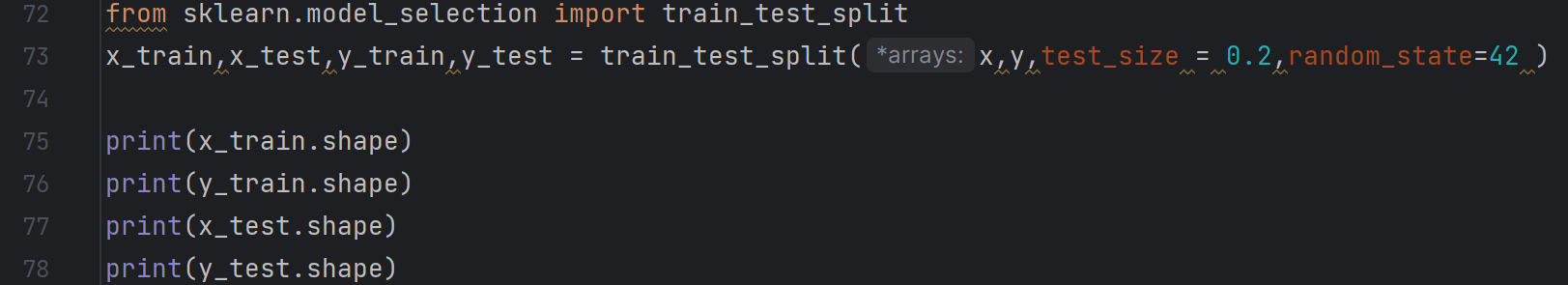
Проведіть аналіз датасету



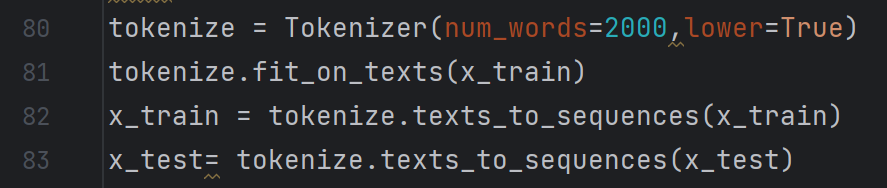




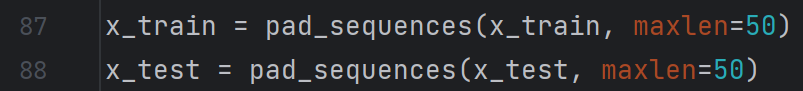
Розділимио датасет на навчальний та тестовий піднабори



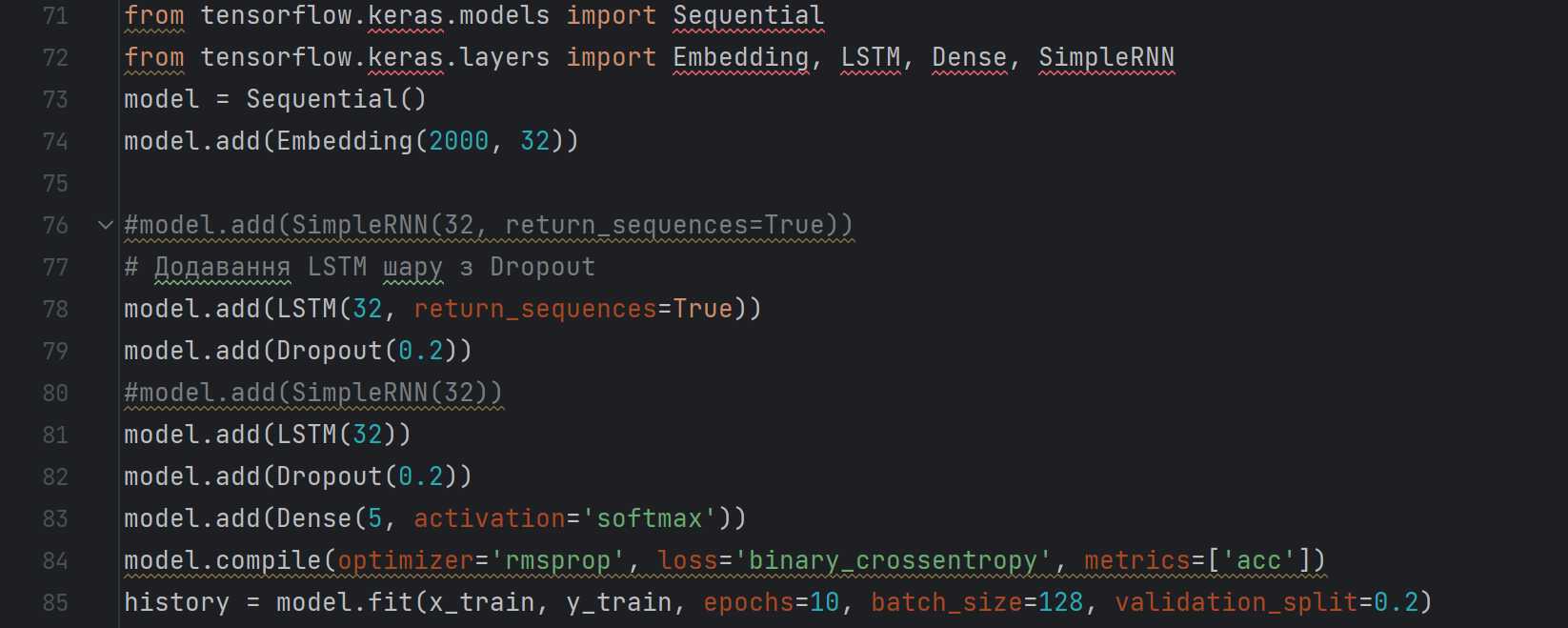
Проведемо Токенізування (tokenize)



Проведемо padding послідовностей до однакової довжини

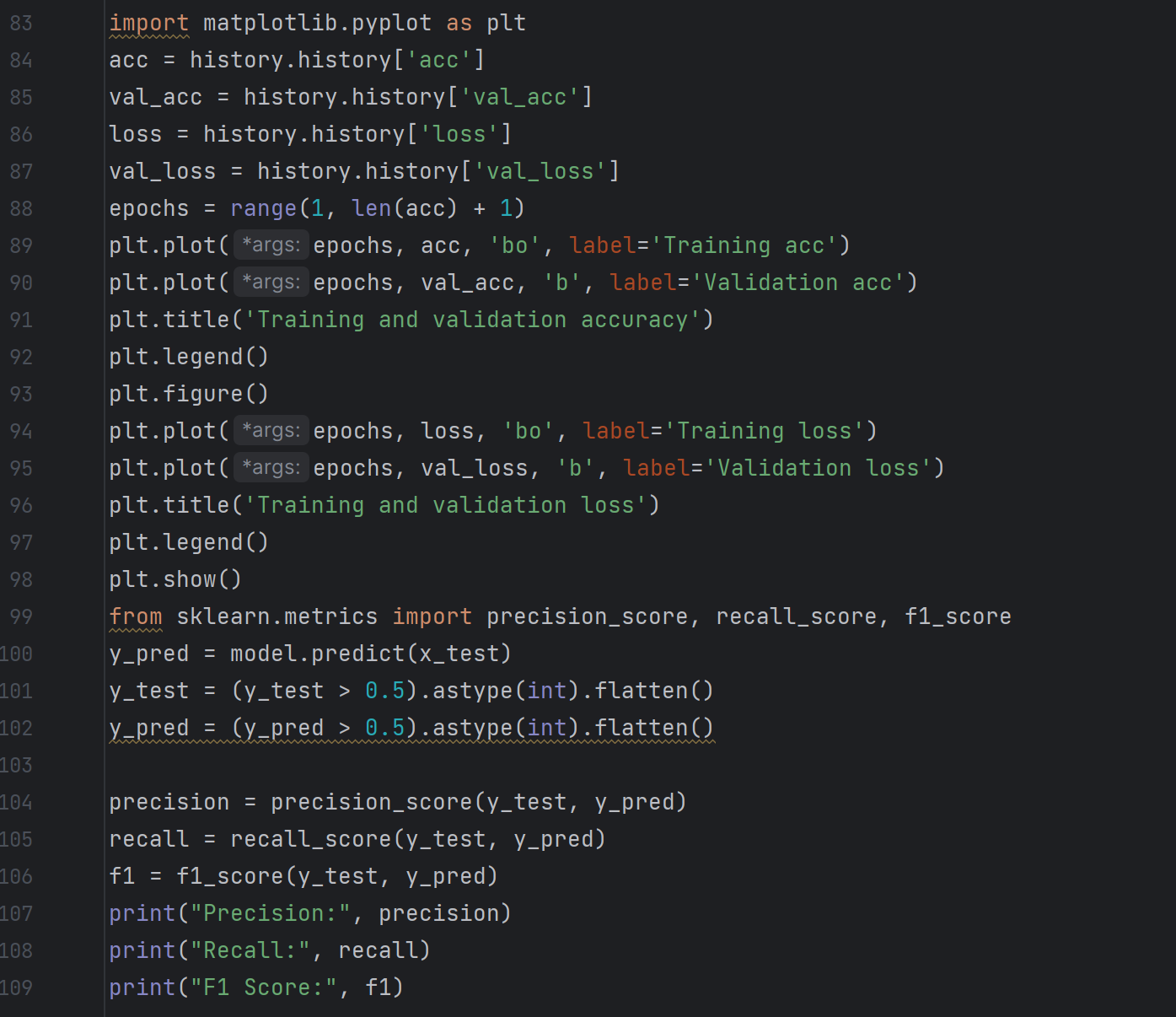


Створимо модель, використовуючи рекурентні шари (LSTM) та додамо Dropouts для запобігання перенавчанню й визначемо із кількістю нейронів у шарах та іншими гіперпараметрами.



4. Тренування та оцінка:

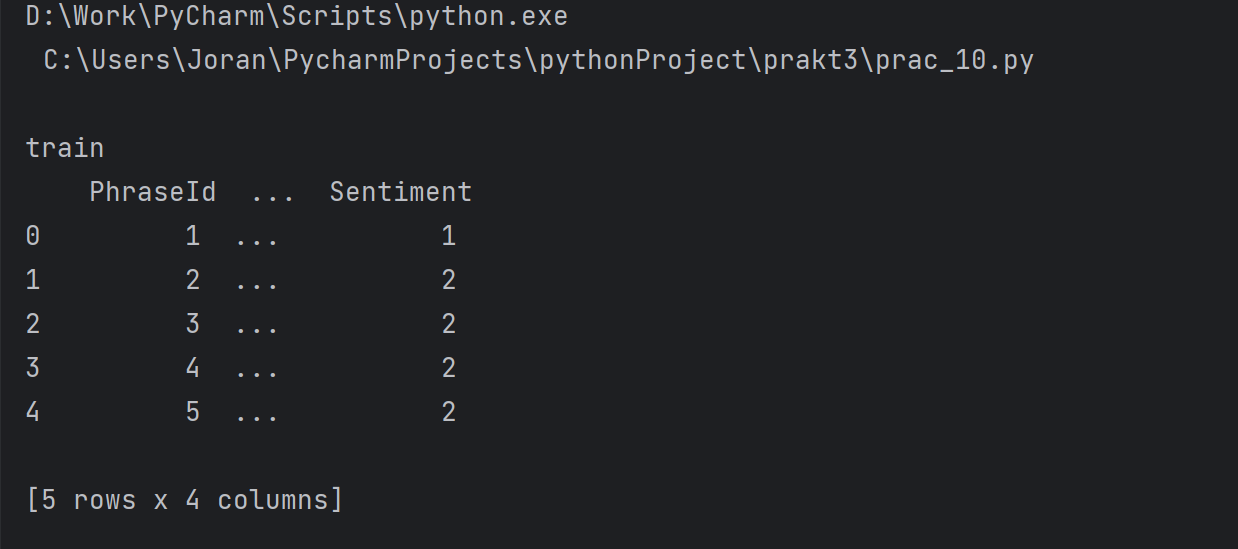
Оцініть якість моделі на тестовому датасеті, використовуючи такі метрики як точність, F1-метрика та інші. Візуалізуйте графіки втрат та точності в процесі

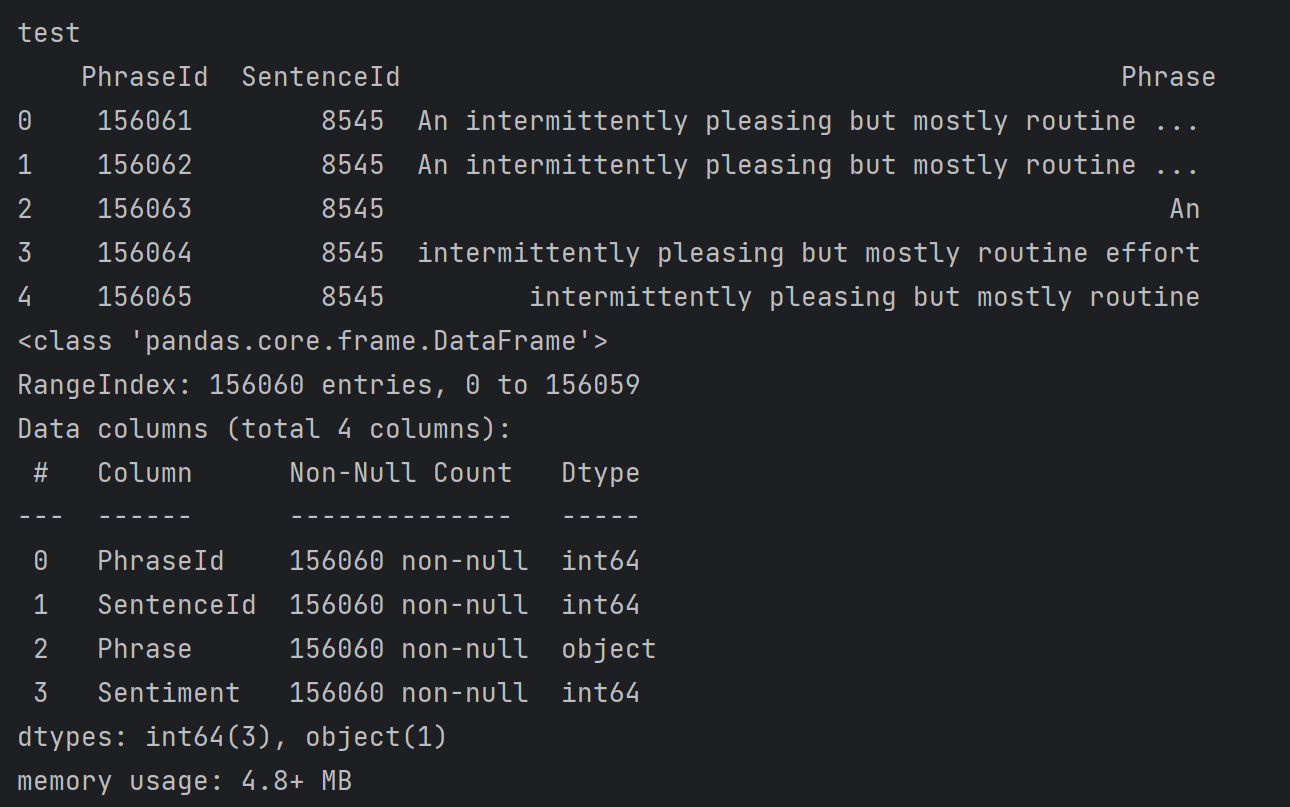


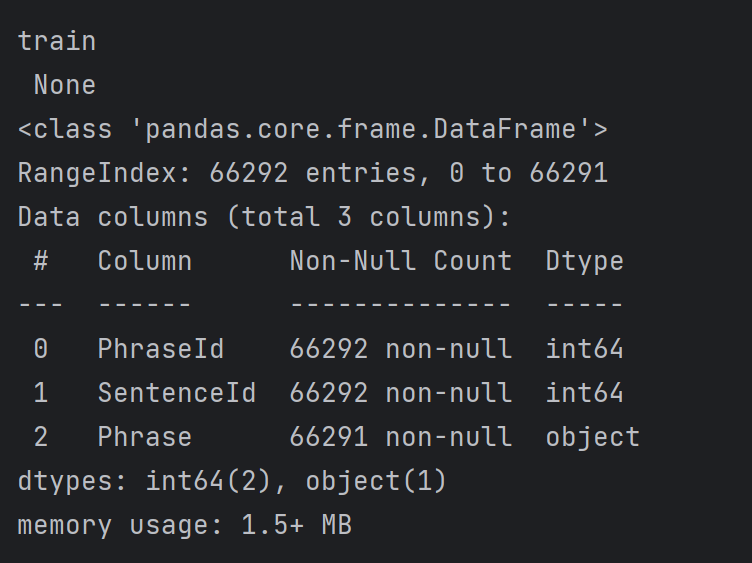
Код програми:

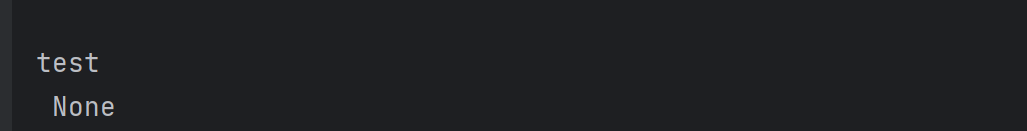
import zipfile  
import pandas as pd  
from tensorflow.keras.preprocessing.sequence import pad\_sequences  
from tensorflow.keras.layers import Dense, Flatten, Dropout, LSTM, Embedding, Bidirectional, BatchNormalization  
import matplotlib.pyplot as plt  
  
with zipfile.ZipFile('train.tsv.zip', 'r') as zip\_ref:  
 zip\_ref.extractall('data')  
  
with zipfile.ZipFile('test.tsv.zip', 'r') as zip\_ref:  
 zip\_ref.extractall('data')  
  
train\_data = pd.read\_csv('data/train.tsv', sep='\t')  
test\_data = pd.read\_csv('data/test.tsv', sep='\t')  
  
print('\ntrain \n',train\_data.head())  
print('\n\ntest \n',test\_data.head())  
print('\n\ntrain \n' ,train\_data.info())  
print('\n\ntest \n' ,test\_data.info())  
print('\n\ntrain \n' ,train\_data.shape)  
print('\n\ntest \n' ,test\_data.shape)  
val\_array = train\_data['Sentiment'].value\_counts()  
plt.bar(val\_array.index, val\_array)  
plt.title('train.tsv - count sentiments')  
plt.xlabel('review rating/Sentiments')  
plt.ylabel('No of reviews')  
plt.show()  
  
train\_data['Word\_Count'] = train\_data['Phrase'].apply(lambda x: len(str(x).split()))  
plt.hist(train\_data['Word\_Count'], bins=20, edgecolor='black')  
plt.title('Distribution of Word Counts in Phrases')  
plt.xlabel('Word Count')  
plt.ylabel('Frequency')  
plt.show()  
  
common\_words = pd.Series(' '.join(train\_data['Phrase']).split()).value\_counts()[:10]  
print("Common Words:\n\n", common\_words)  
  
  
x = train\_data['Phrase']  
y = train\_data['Sentiment']  
from sklearn.model\_selection import train\_test\_split  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.2,random\_state=42 )  
  
  
print(x\_train.shape)  
print(y\_train.shape)  
print(x\_test.shape)  
print(y\_test.shape)  
from keras.preprocessing.text import Tokenizer  
tokenize = Tokenizer(num\_words=2000,lower=True)  
tokenize.fit\_on\_texts(x\_train)  
x\_train = tokenize.texts\_to\_sequences(x\_train)  
x\_test= tokenize.texts\_to\_sequences(x\_test)  
  
x\_train = pad\_sequences(x\_train, maxlen=50)  
x\_test = pad\_sequences(x\_test, maxlen=50)  
  
  
from keras.utils import to\_categorical  
num\_classes = 5  
y\_train = to\_categorical(y\_train,num\_classes)  
y\_test = to\_categorical(y\_test,num\_classes)  
print(y\_train.shape)  
print(x\_train.shape)  
  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Embedding, LSTM, Dense, SimpleRNN  
model = Sequential()  
model.add(Embedding(2000, 32))  
model.add(SimpleRNN(32, return\_sequences=True))  
  
model.add(LSTM(32, return\_sequences=True))  
model.add(Dropout(0.1))  
#model.add(SimpleRNN(32))  
model.add(LSTM(32))  
model.add(Dropout(0.1))  
model.add(Dense(5, activation='sigmoid'))  
  
model.compile(optimizer='rmsprop', loss='binary\_crossentropy', metrics=['acc'])  
history = model.fit(x\_train, y\_train, epochs=10, batch\_size=128, validation\_split=0.2)  
  
import matplotlib.pyplot as plt  
acc = history.history['acc']  
val\_acc = history.history['val\_acc']  
loss = history.history['loss']  
val\_loss = history.history['val\_loss']  
epochs = range(1, len(acc) + 1)  
plt.plot(epochs, acc, 'bo', label='Training acc')  
plt.plot(epochs, val\_acc, 'b', label='Validation acc')  
plt.title('Training and validation accuracy')  
plt.legend()  
plt.figure()  
plt.plot(epochs, loss, 'bo', label='Training loss')  
plt.plot(epochs, val\_loss, 'b', label='Validation loss')  
plt.title('Training and validation loss')  
plt.legend()  
plt.show()  
from sklearn.metrics import precision\_score, recall\_score, f1\_score  
y\_pred = model.predict(x\_test)  
y\_test = (y\_test > 0.5).astype(int).flatten()  
y\_pred = (y\_pred > 0.5).astype(int).flatten()  
  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)  
f1 = f1\_score(y\_test, y\_pred)  
print("Precision:", precision)  
print("Recall:", recall)  
print("F1 Score:", f1)

Результати:

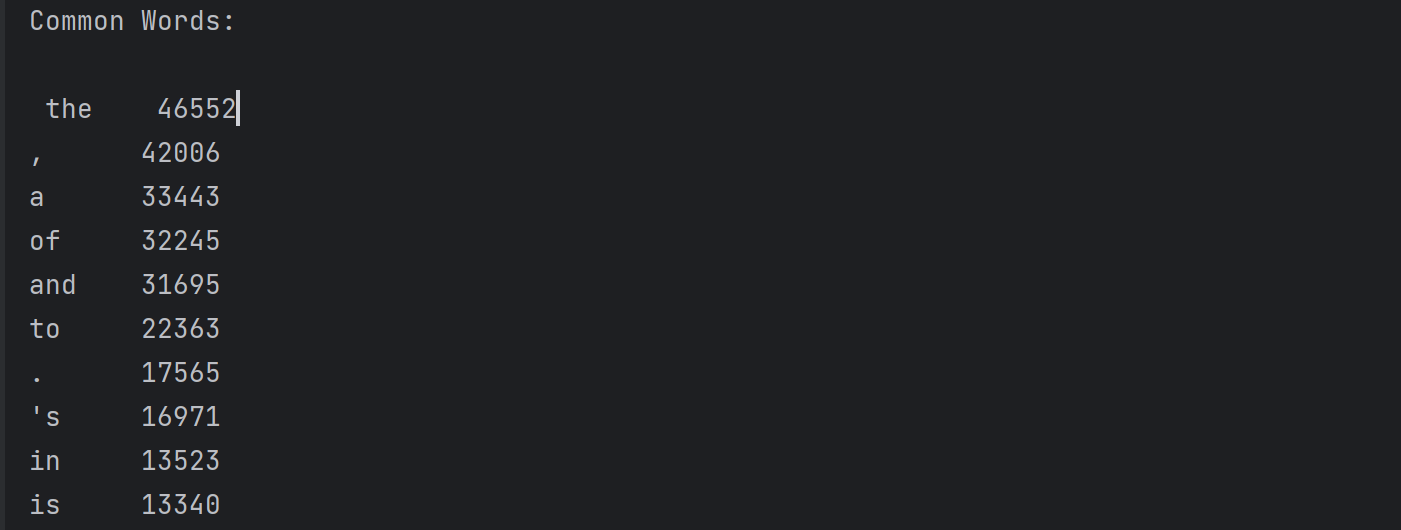




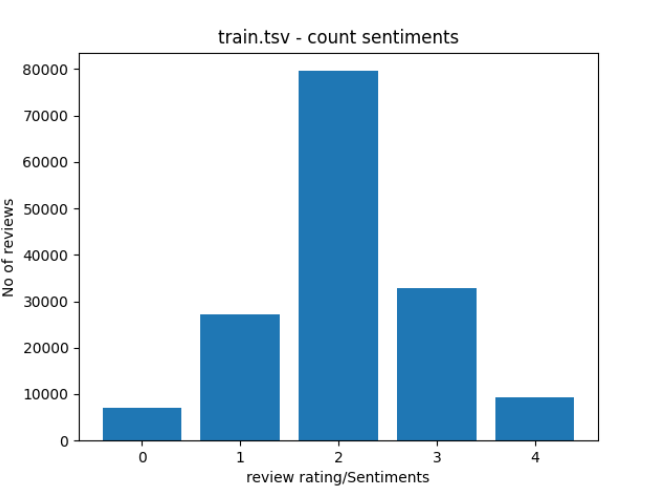


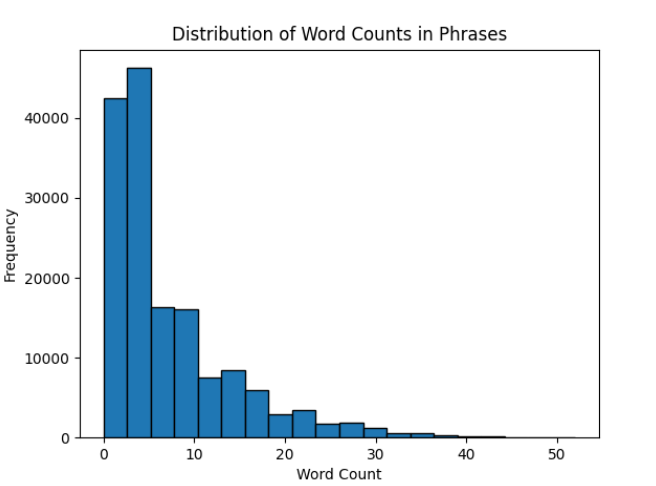


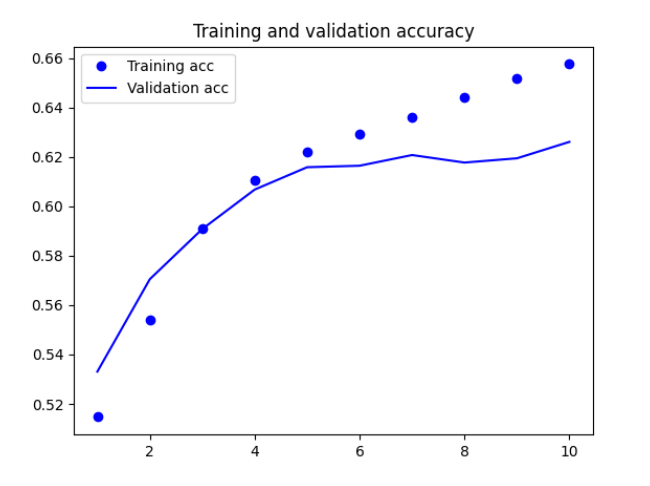


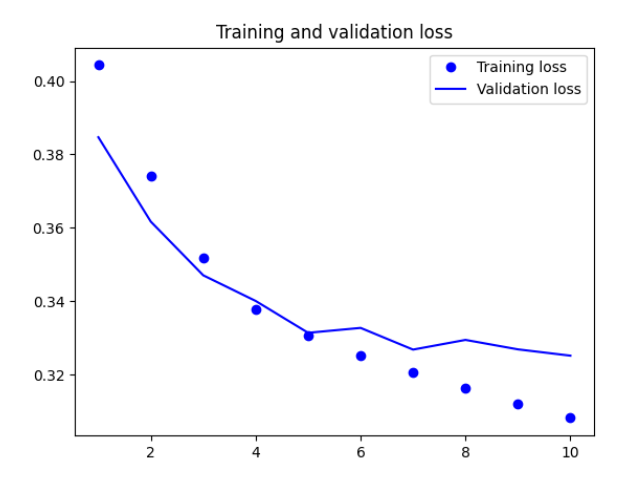


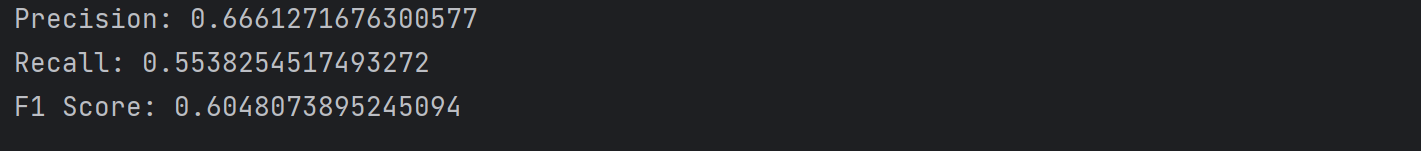






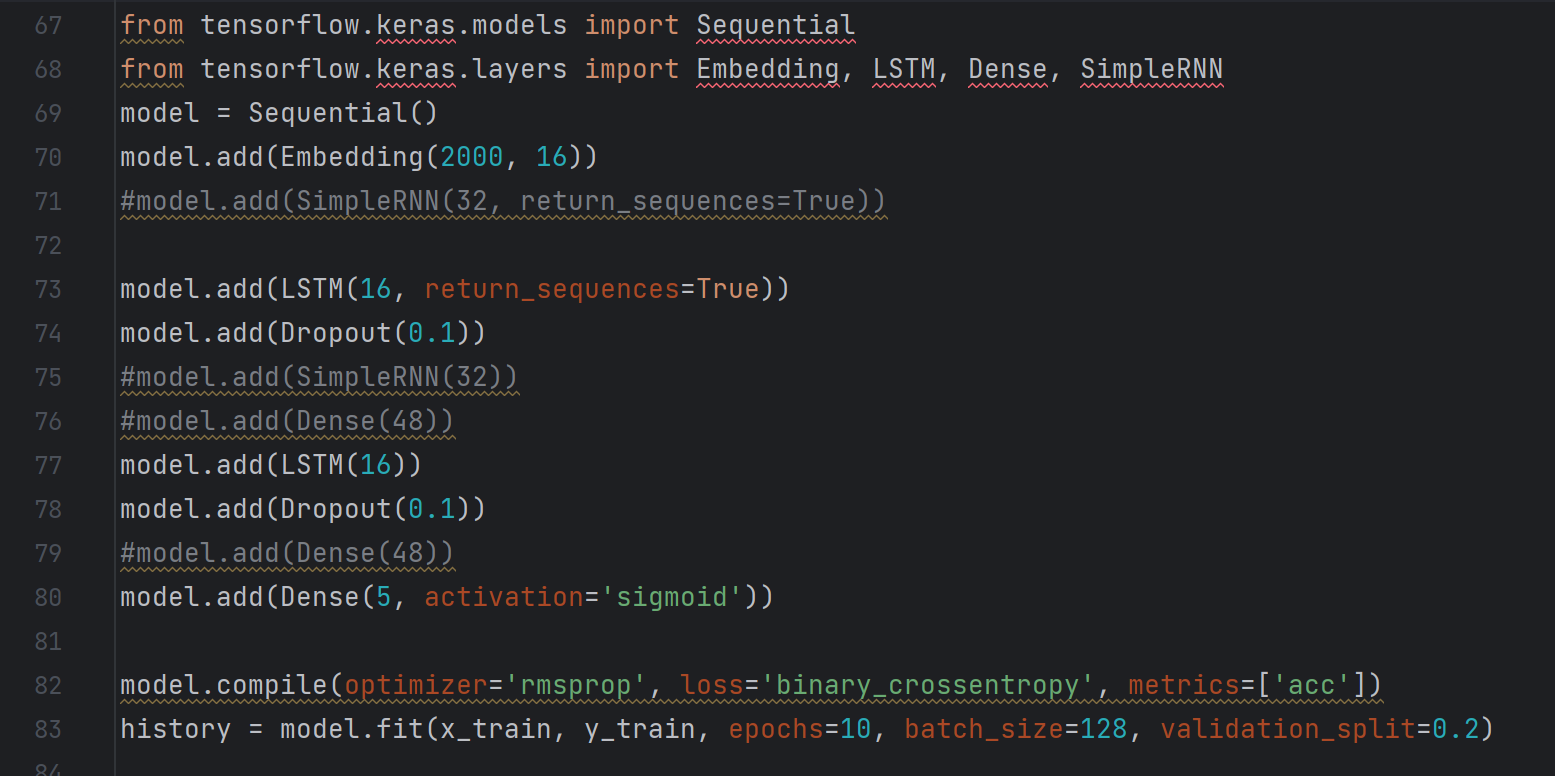
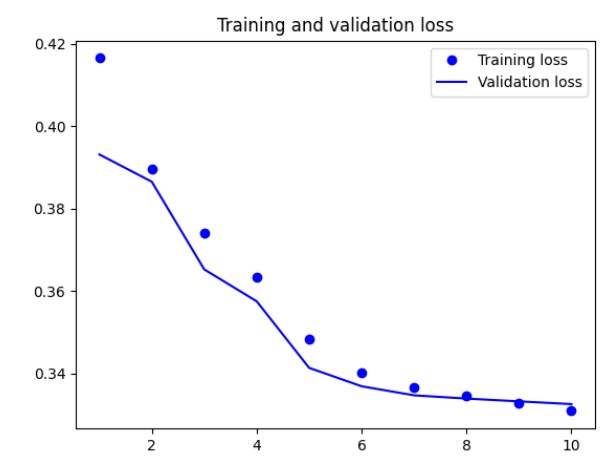




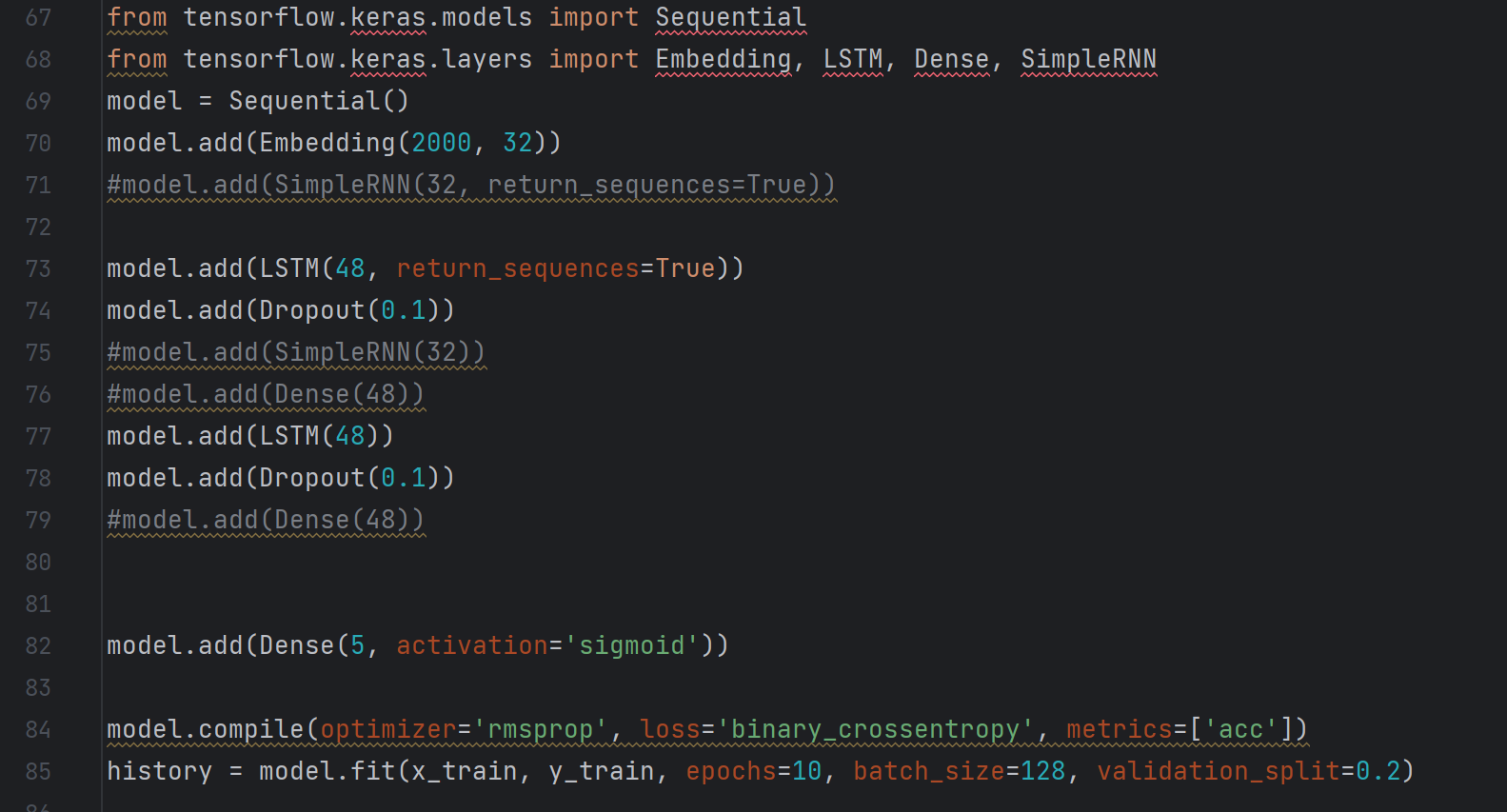
 

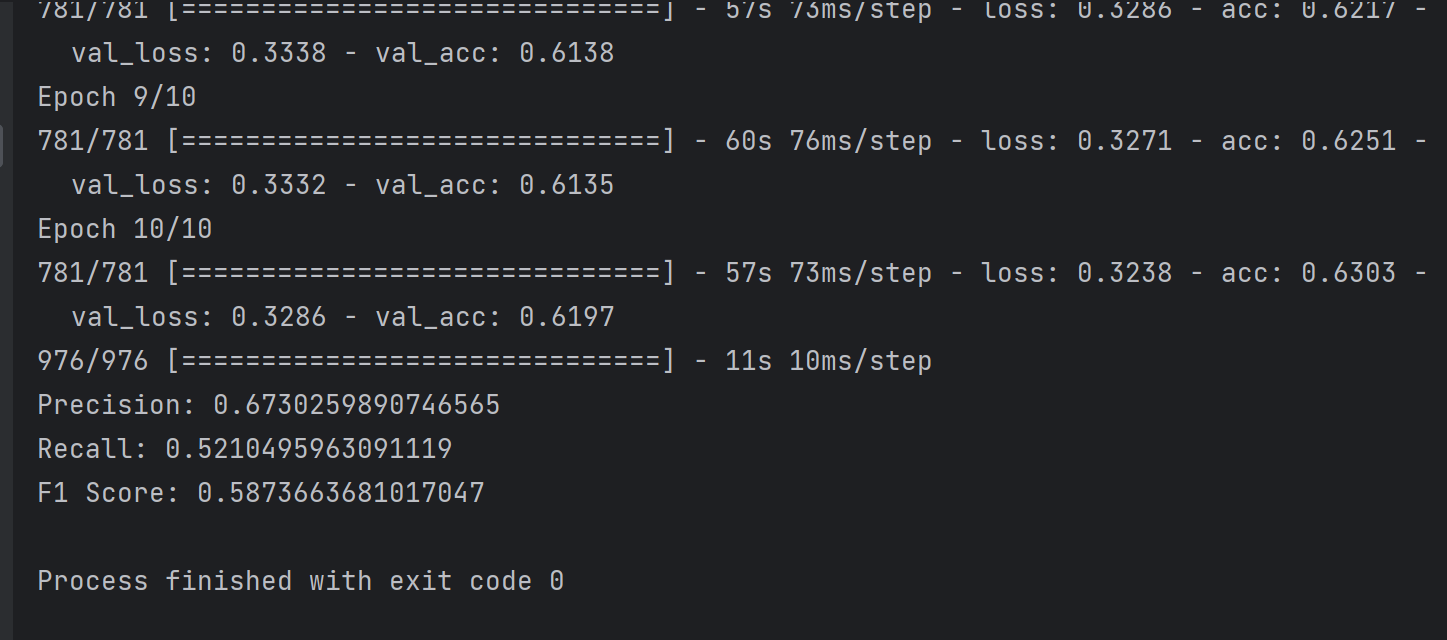
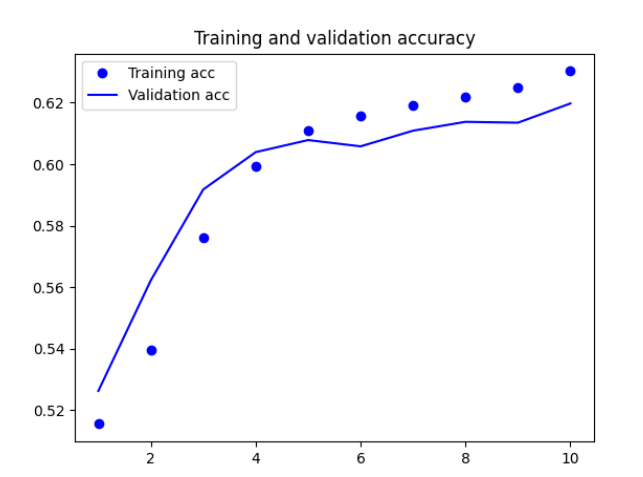
Проведемо оптимізацію нашої нейроної мережі: Застосуємо різні методи оптимізації моделі

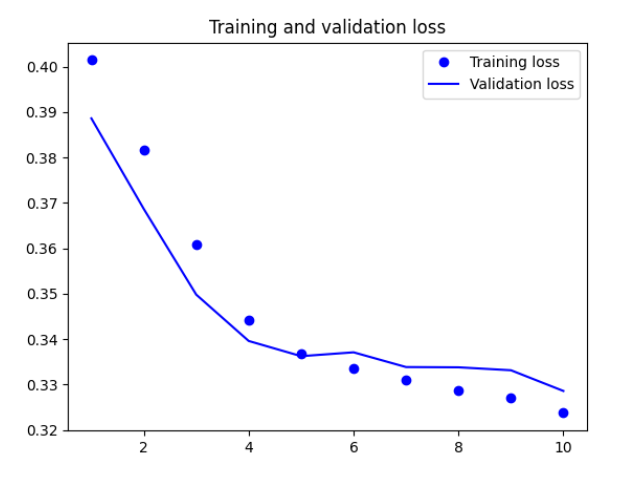
Зменшимо розміри шарів

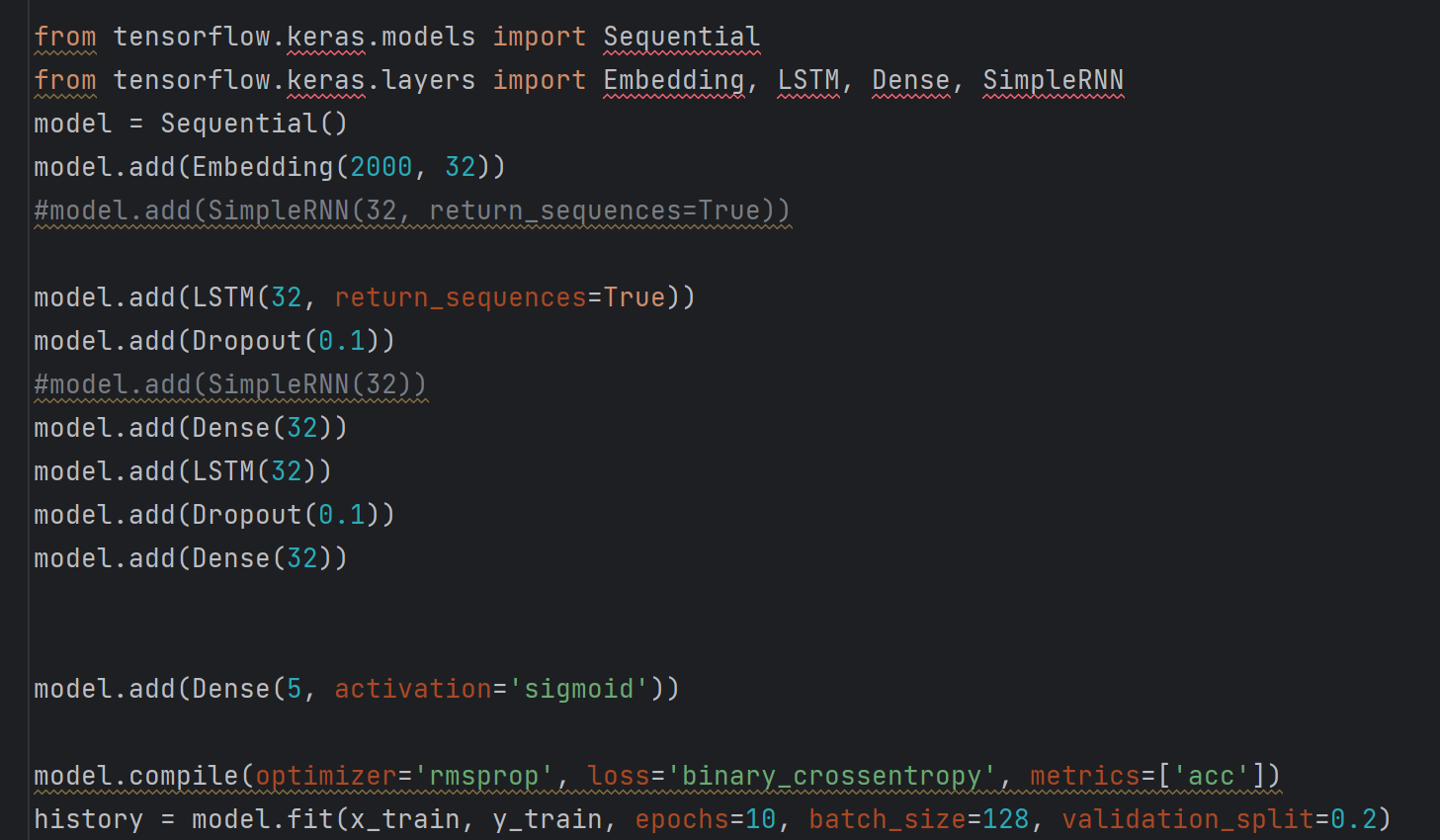
   

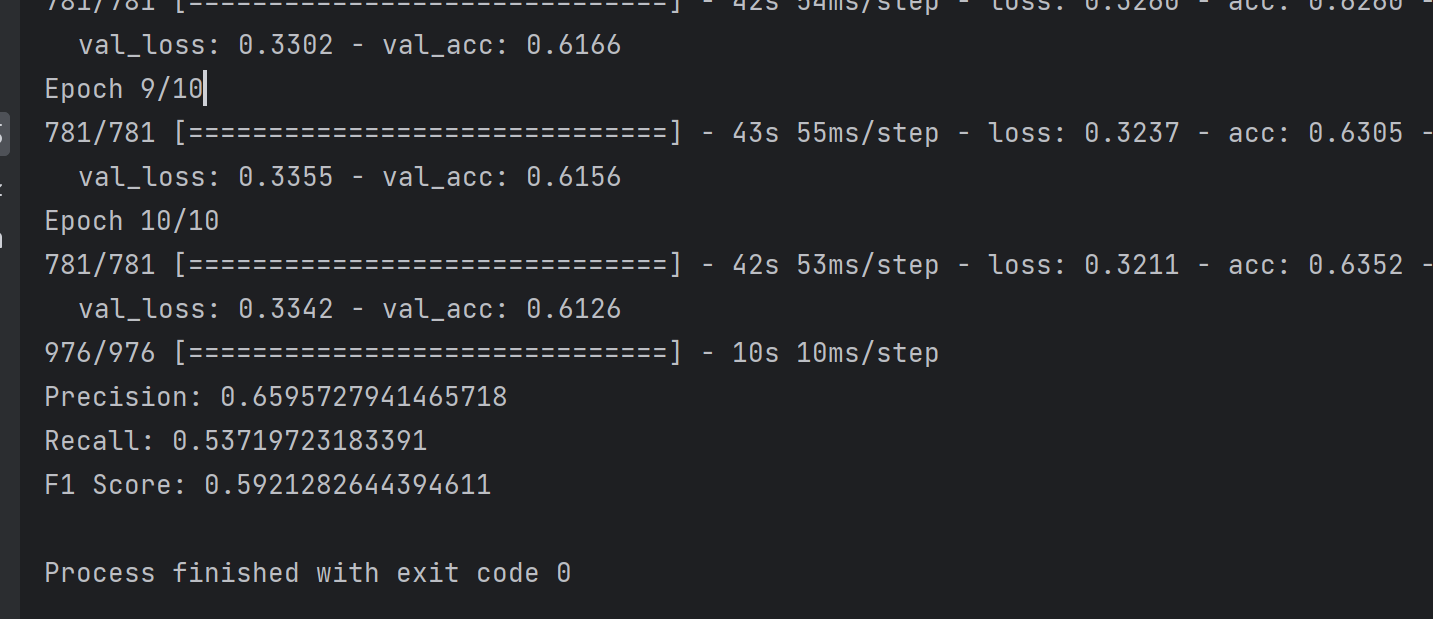
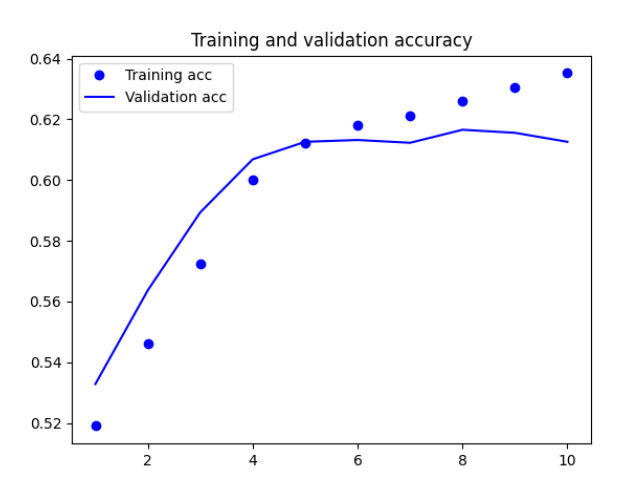
Збільшимо розміри шарів

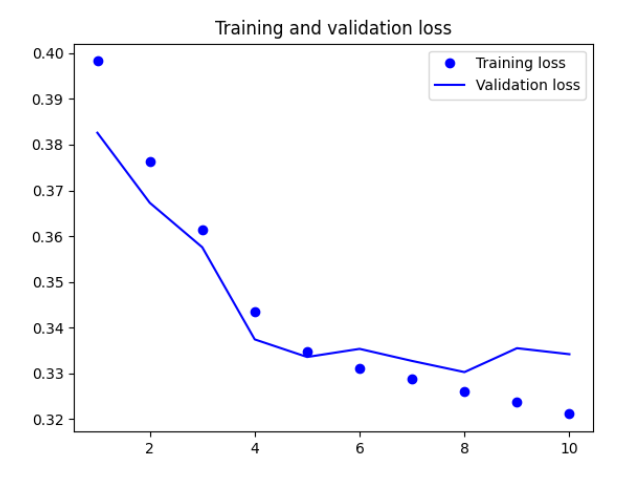


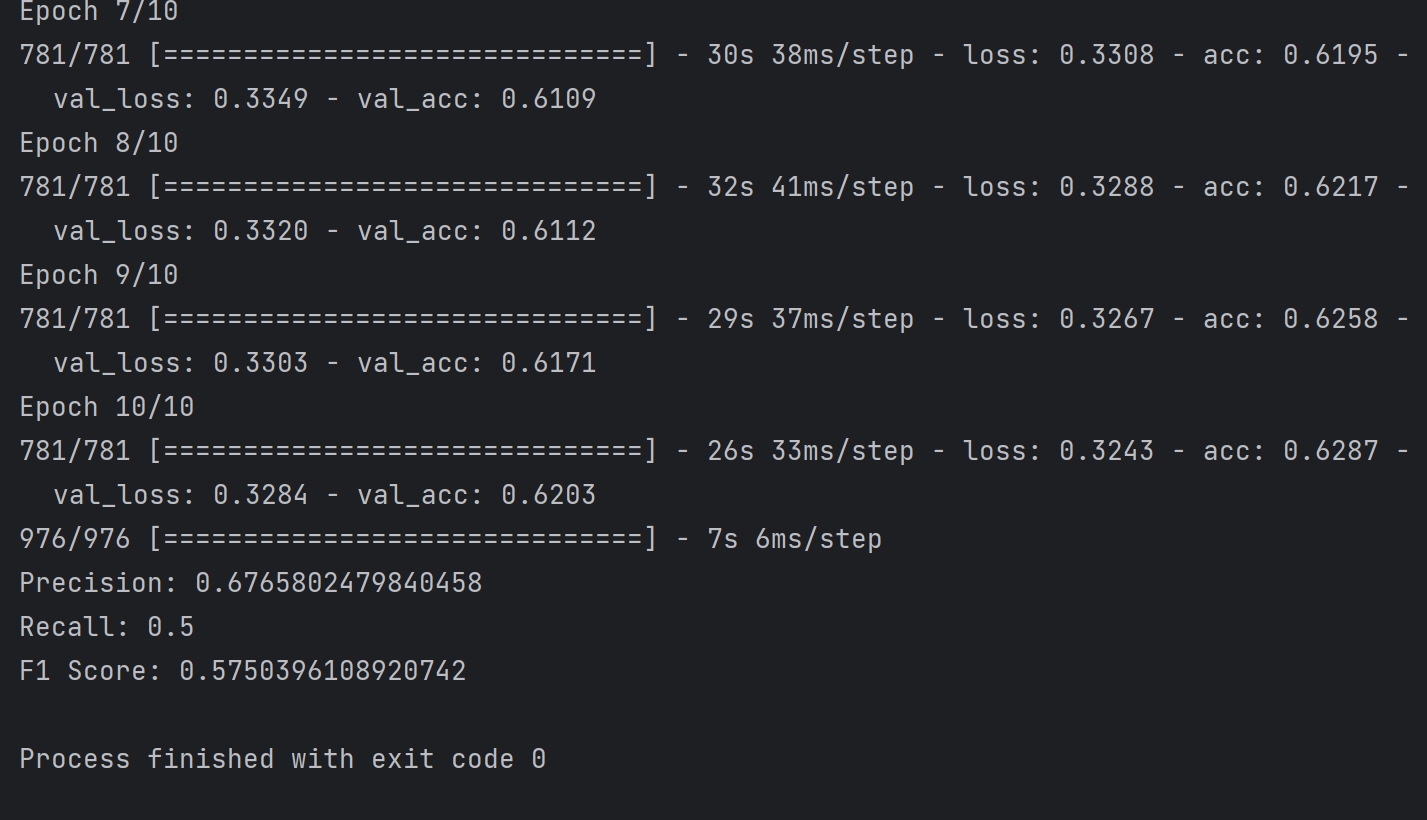
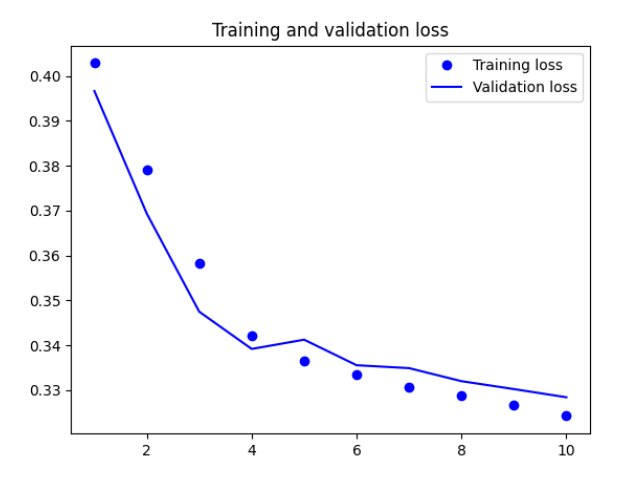
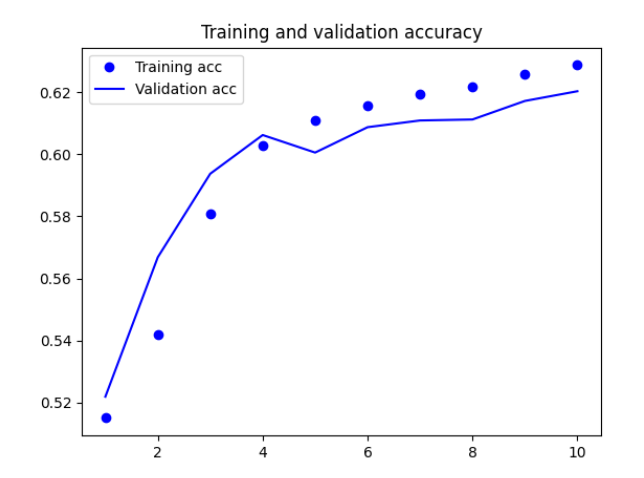


додамо додатковий шари

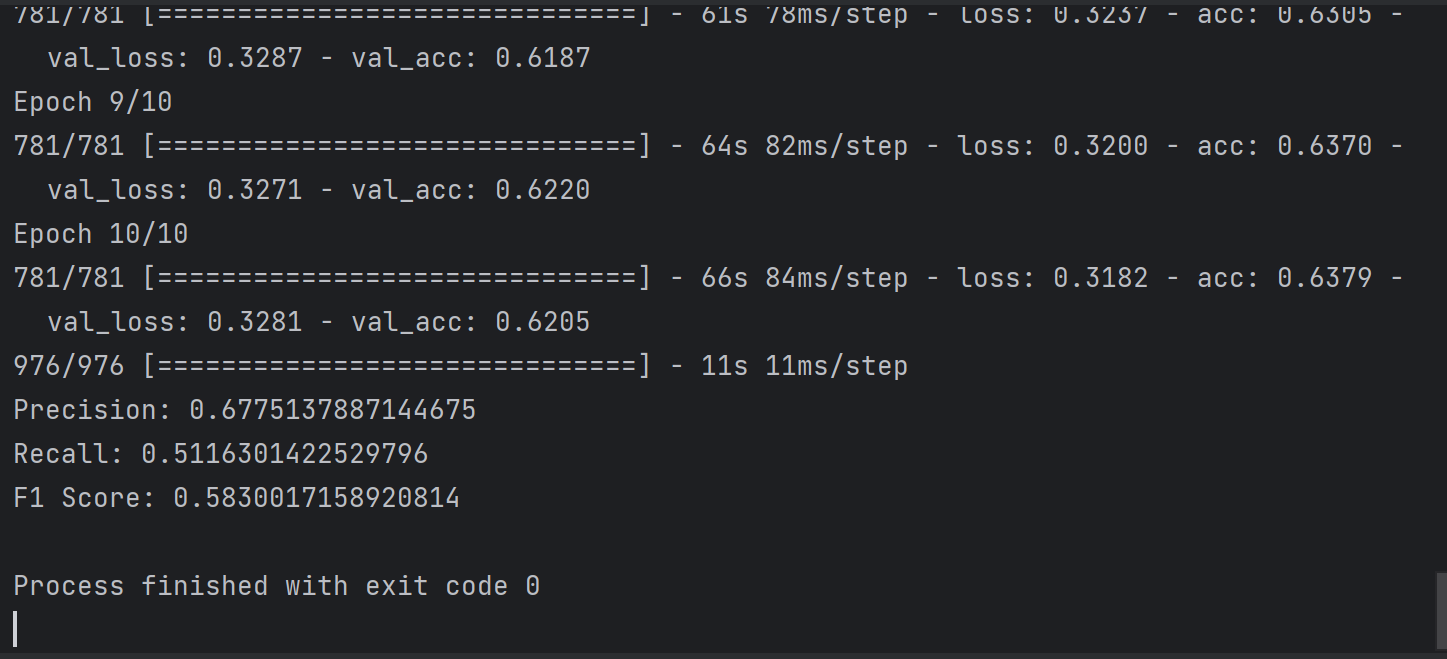
 

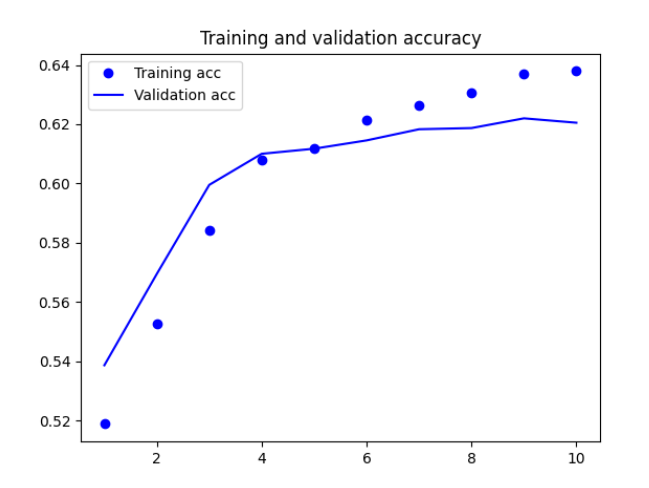
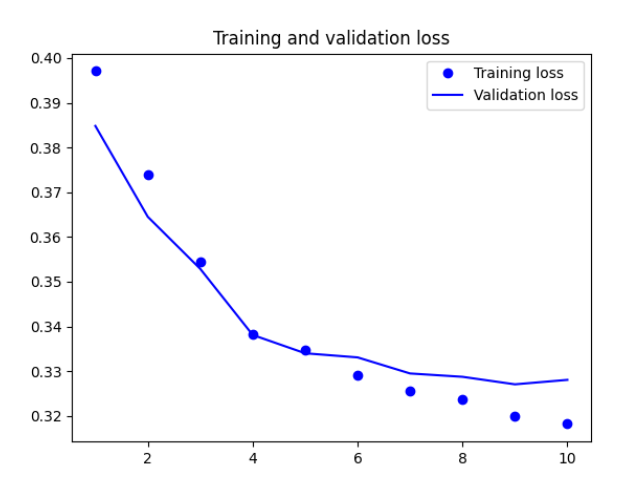


Зменшимо розміри всіх шарів

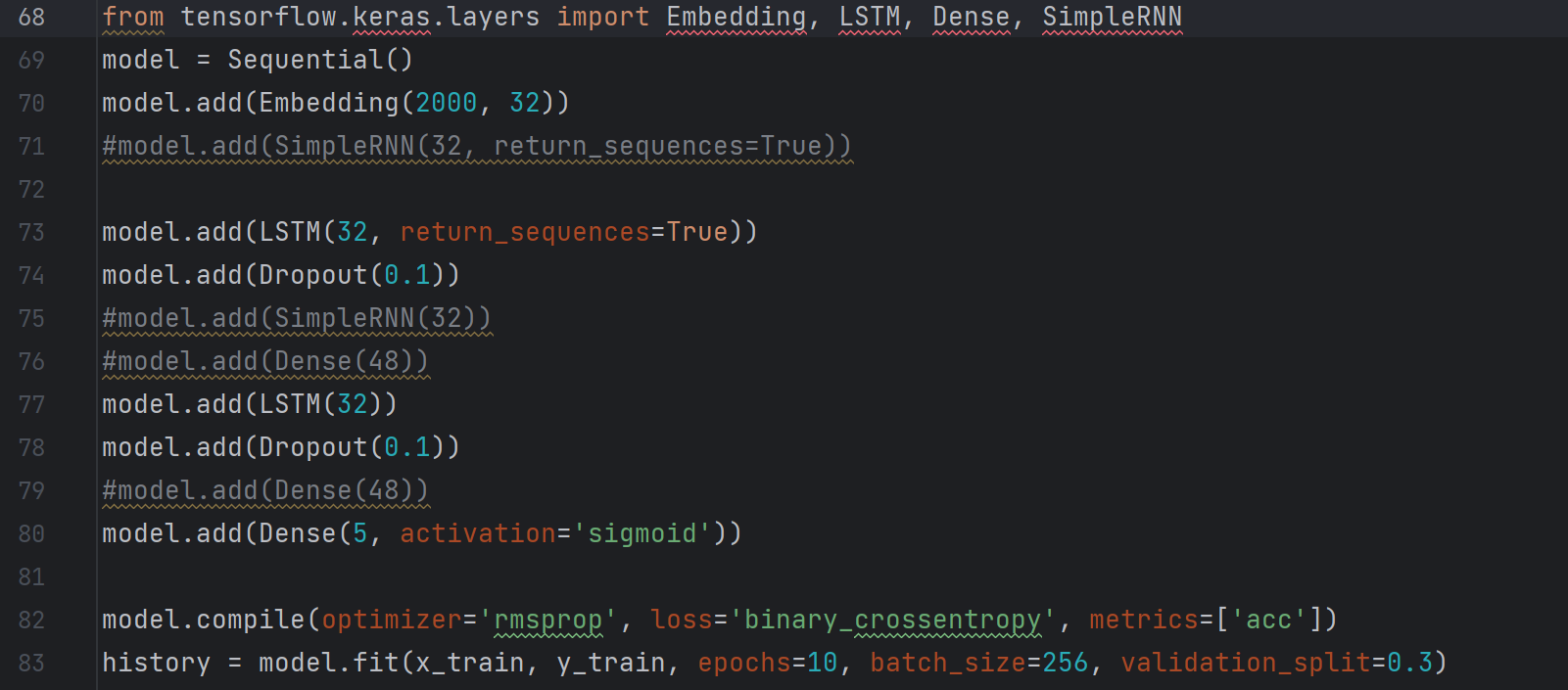
  

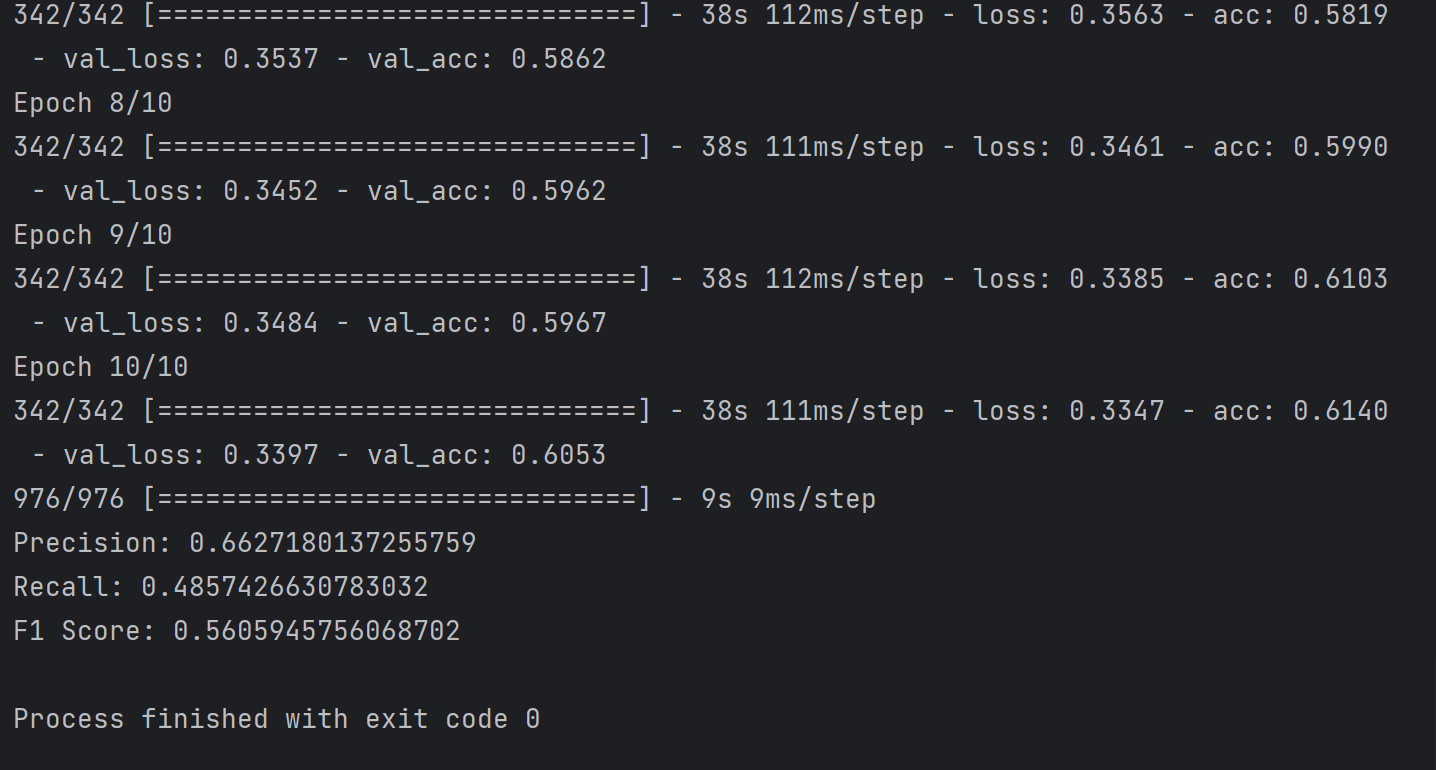
Збільшимо розміри всіх шарів

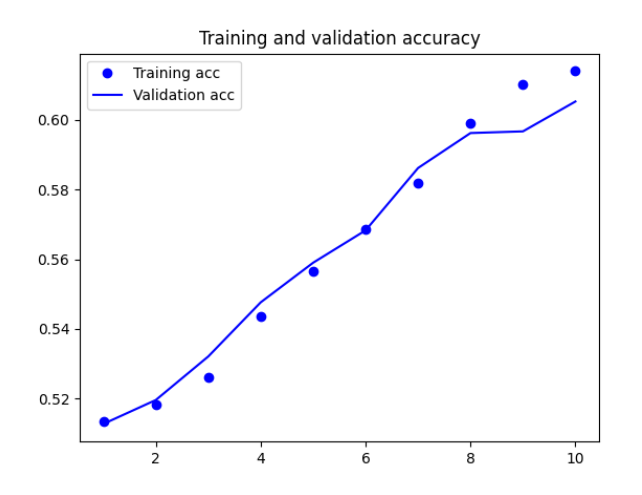


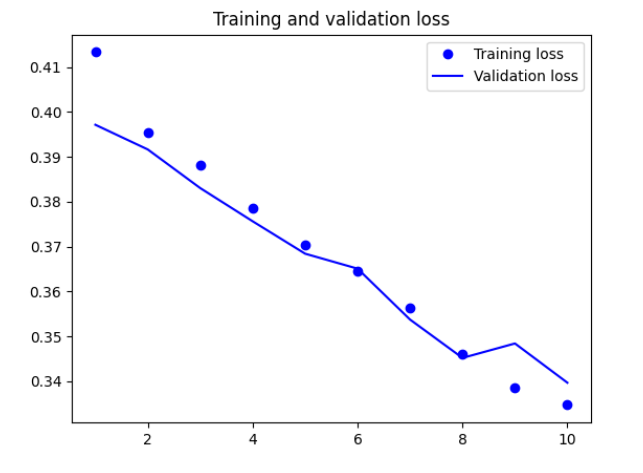
 

Збільшено валідацію та розмір пакета

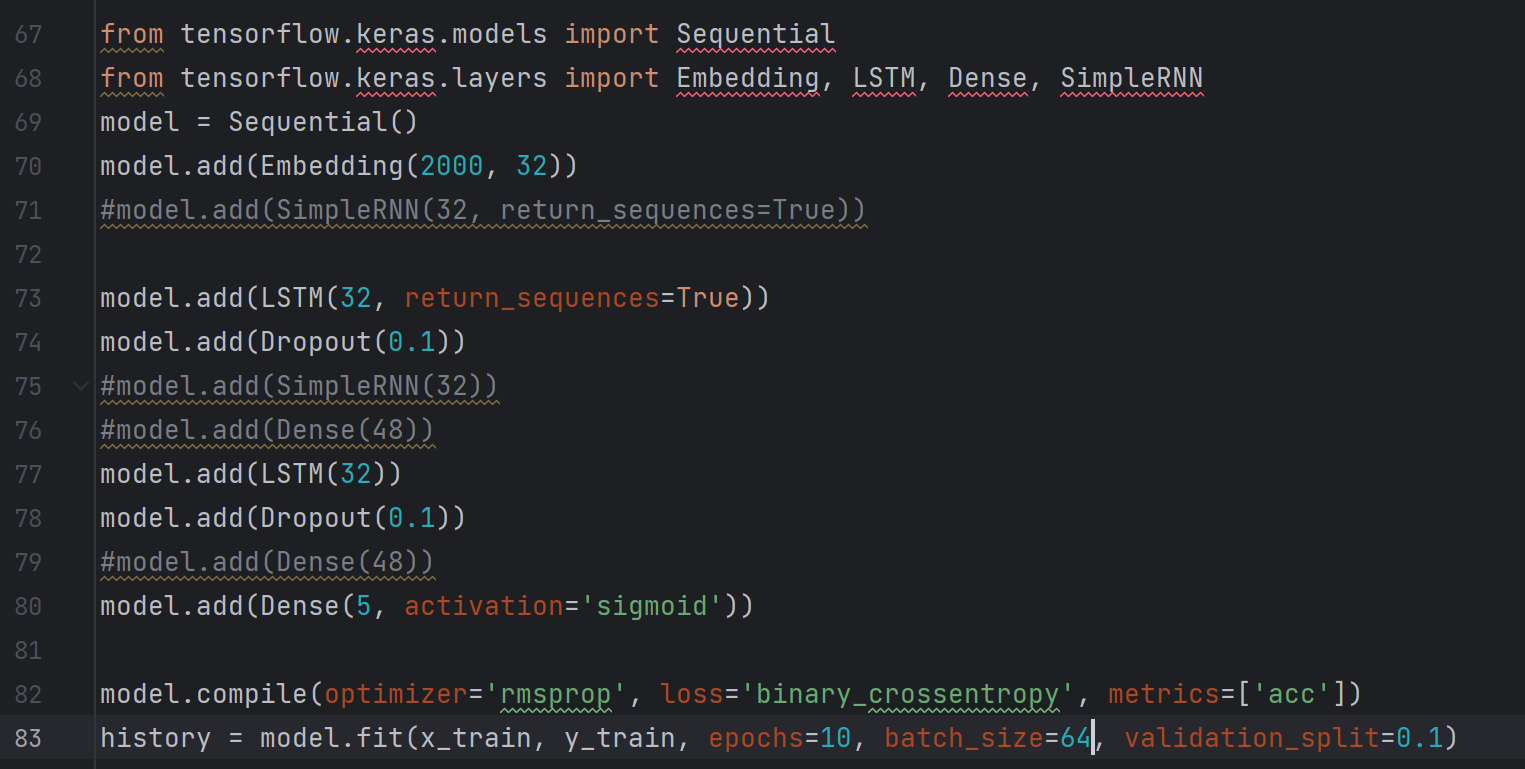


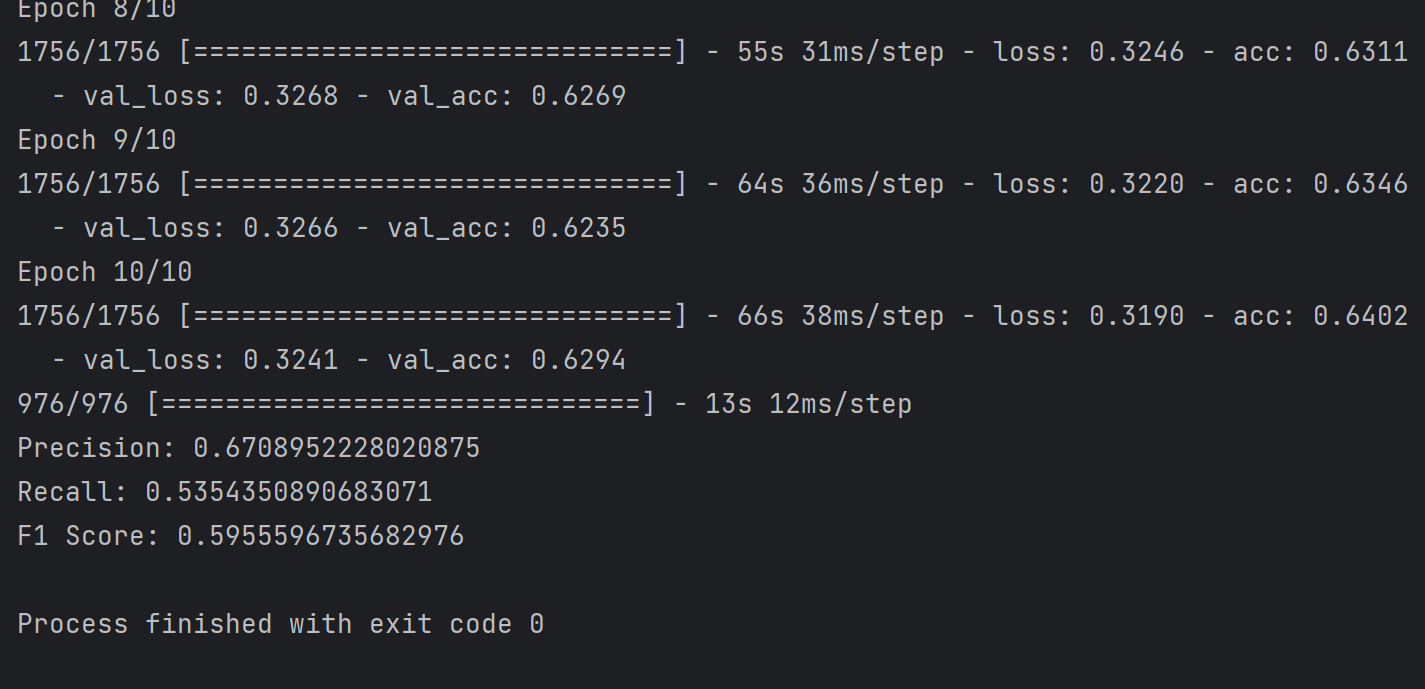


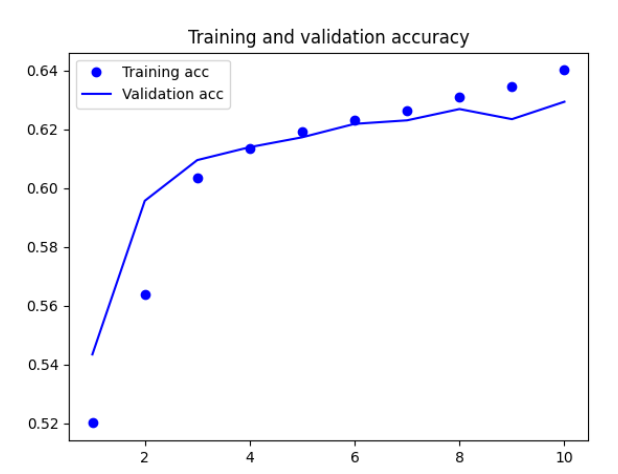


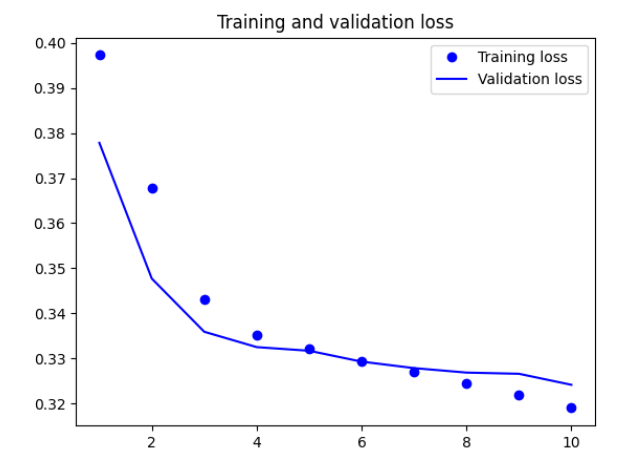


Зменшемо валідацію та розмір пакета

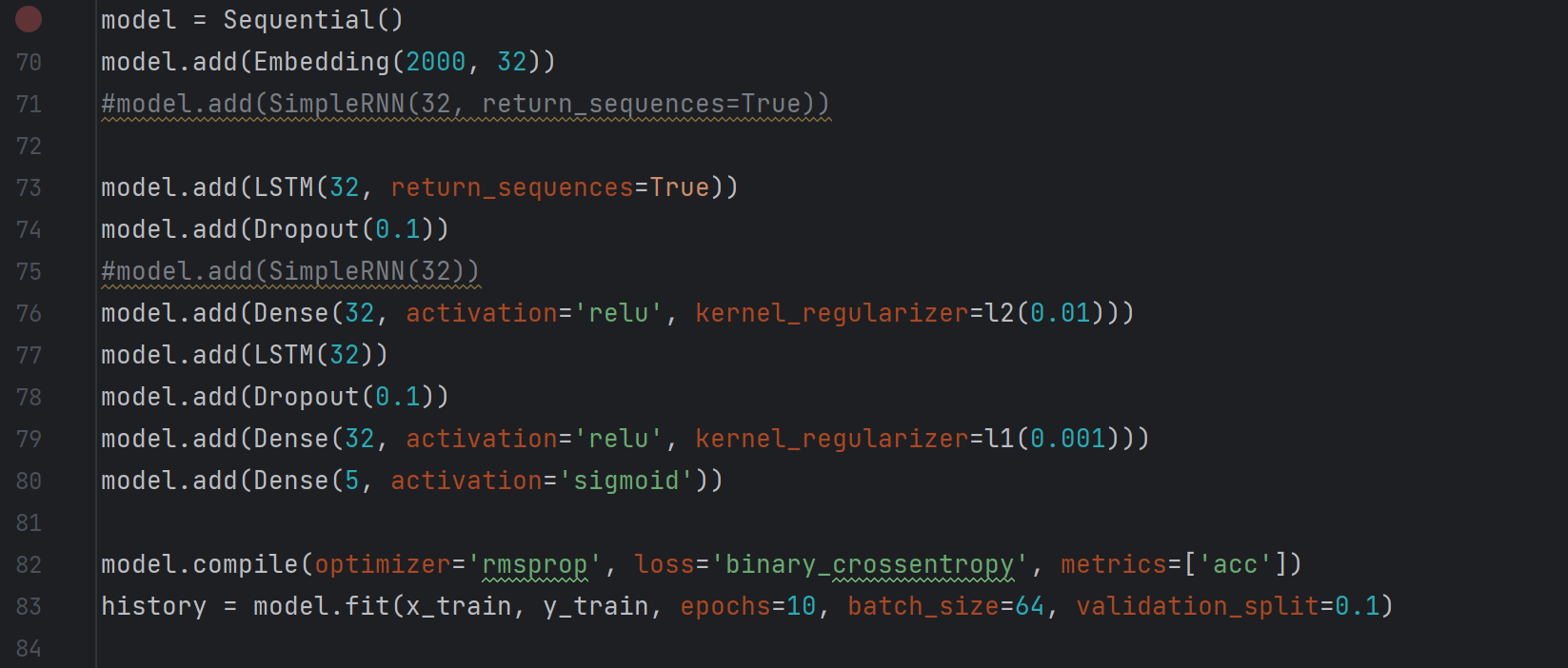






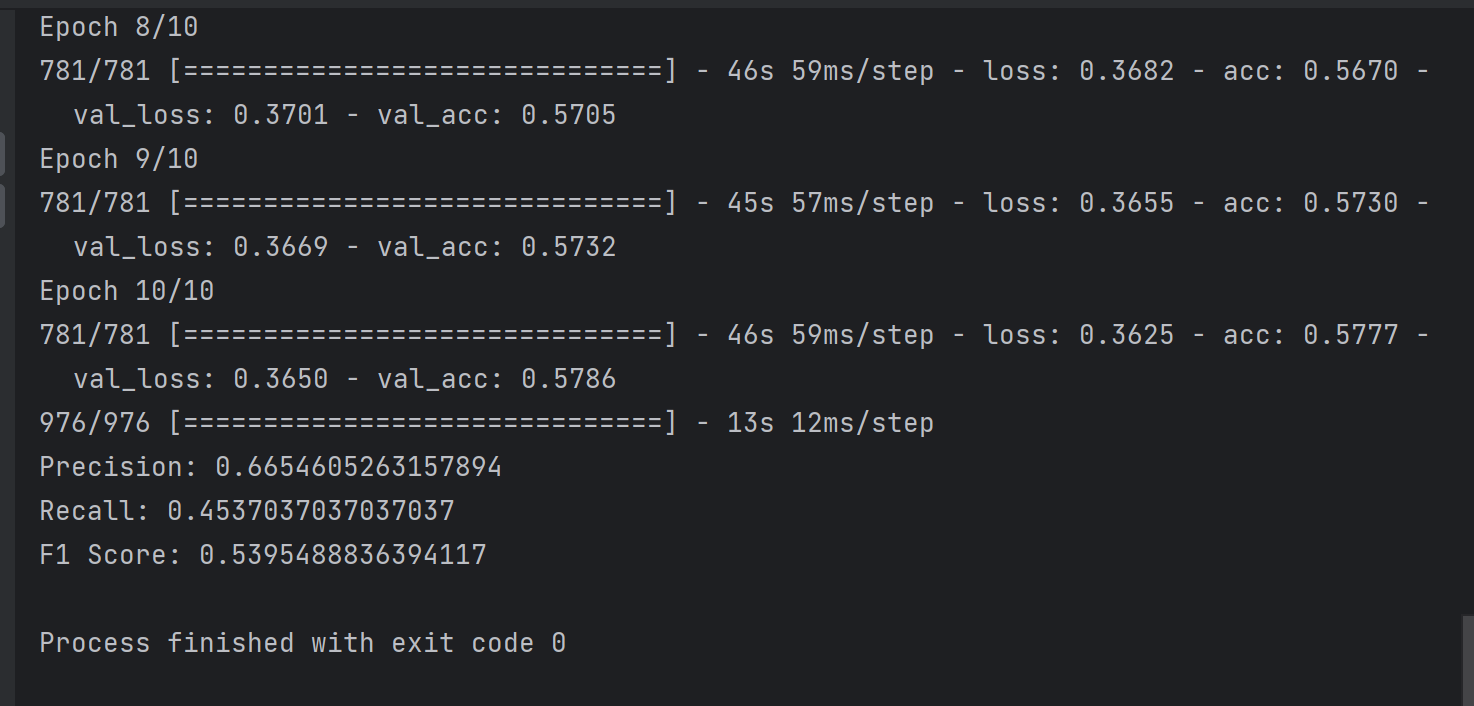


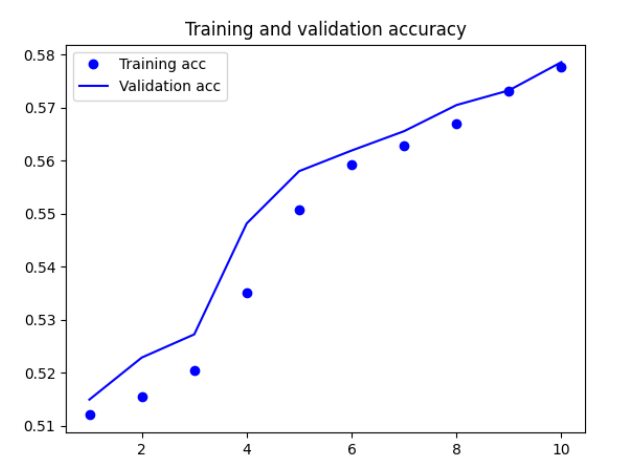
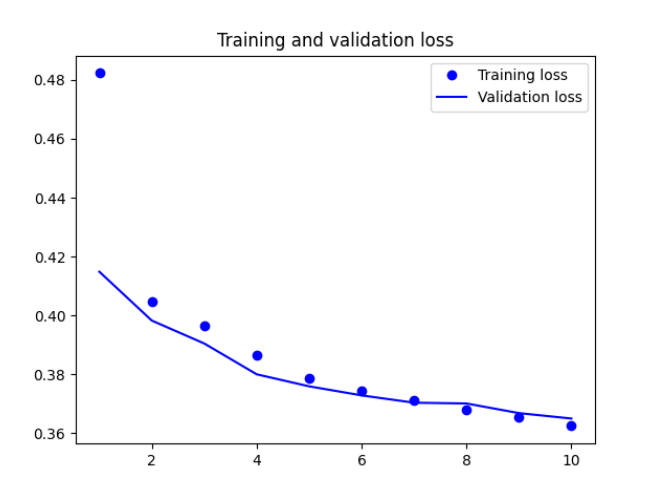
додайте регуляризацію L2,L1

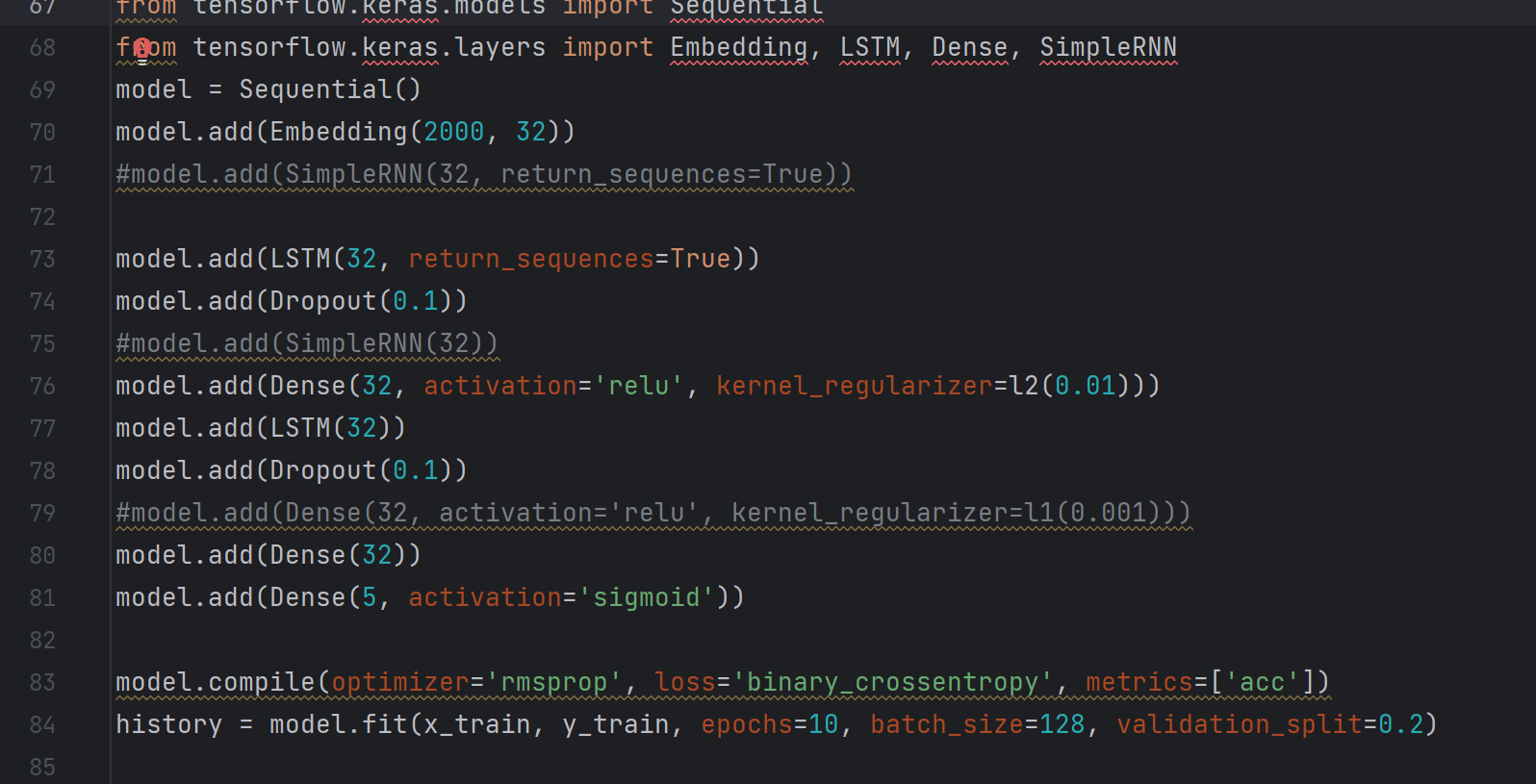


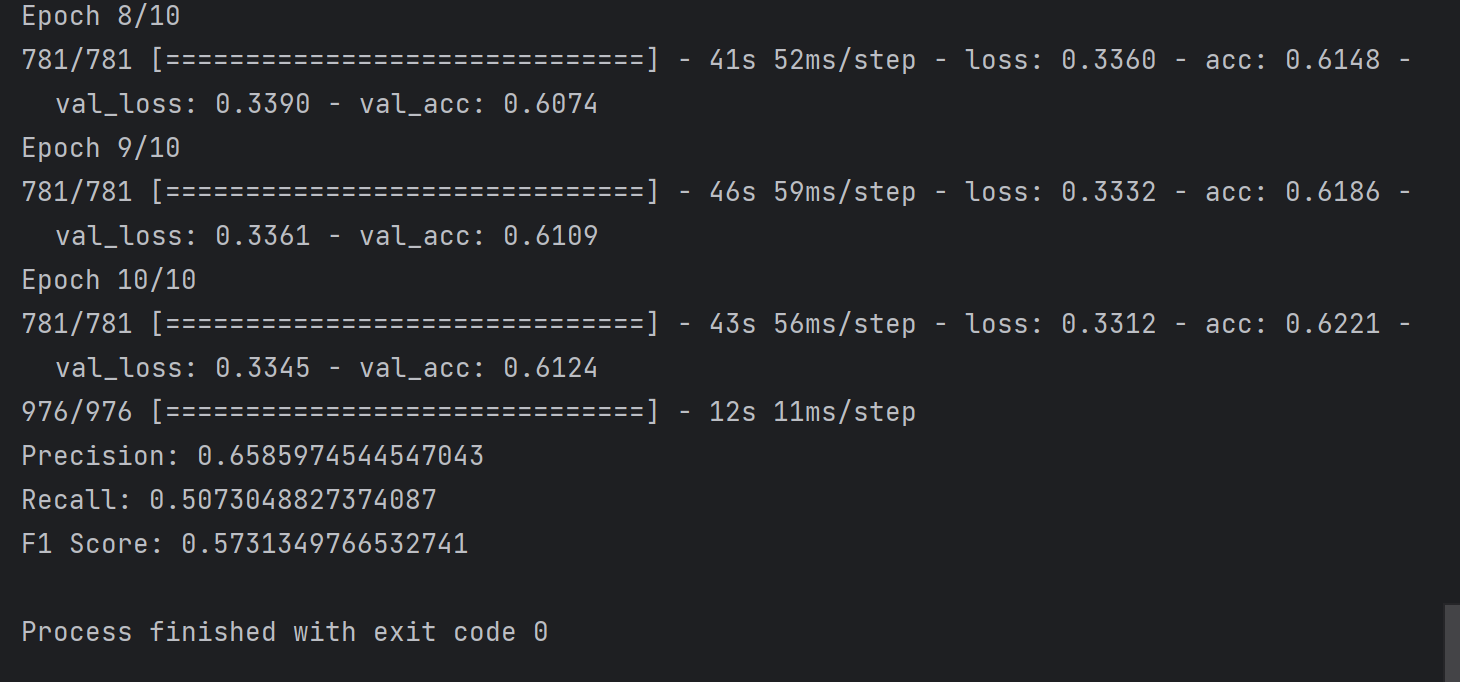
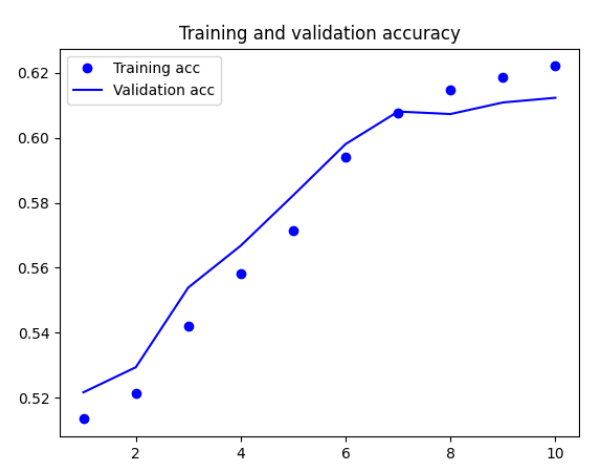
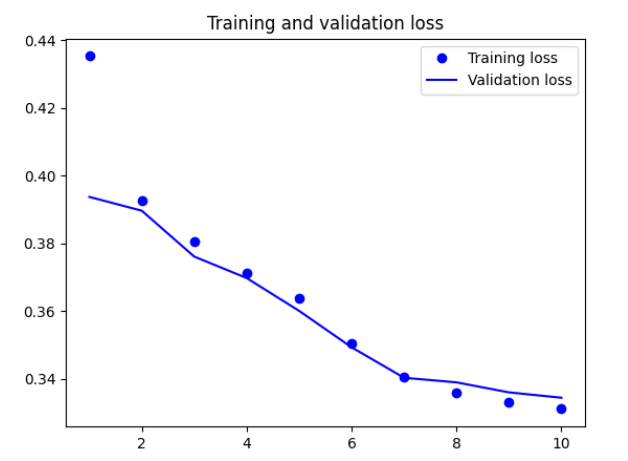
додайте регуляризацію L2, L1

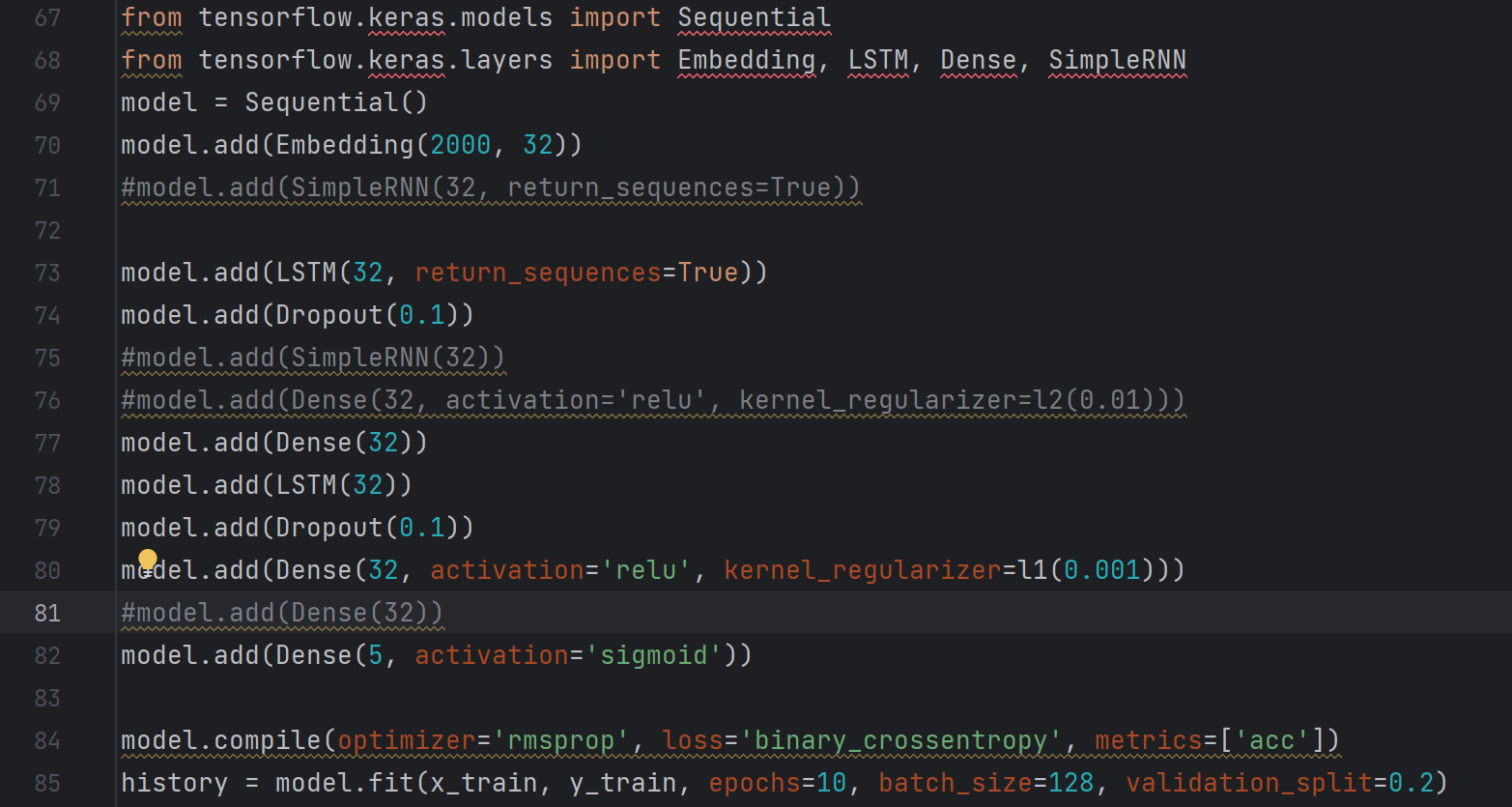


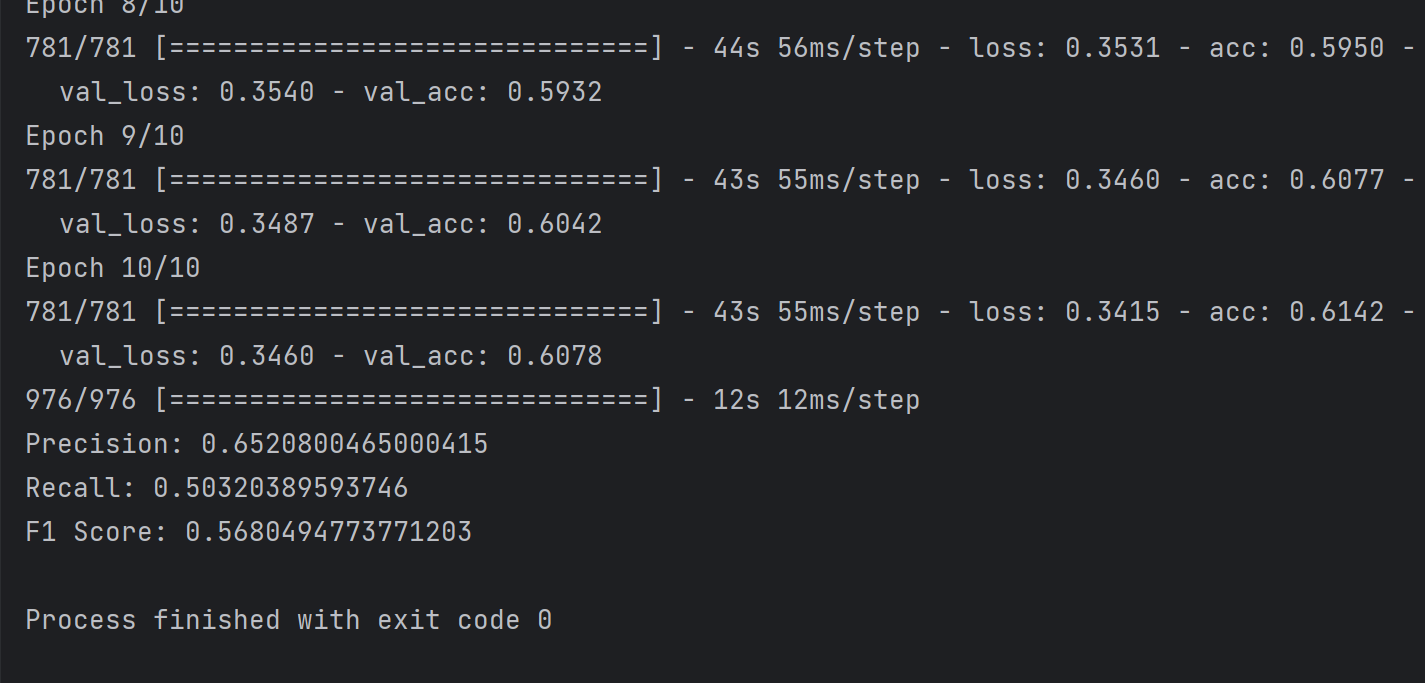
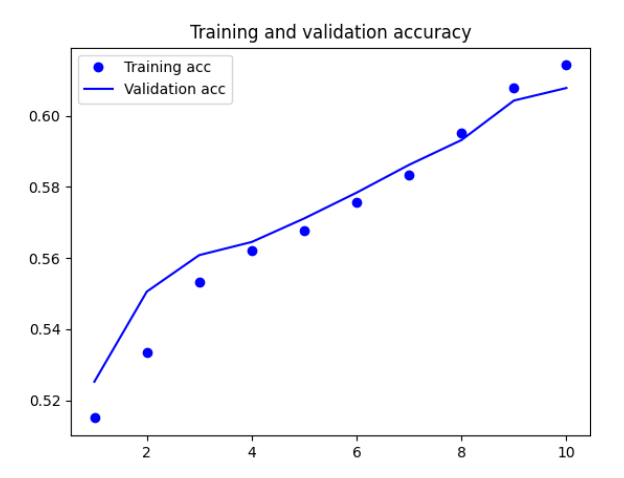


додайте регуляризацію L2

додайте регуляризацію L1 



В результаті проведення експерементів над базою даних MovieLens Sentiment Dataset з використанням рекурентні нейронні мережі (RNN) в варіації аріації RNN: LSTM можна зробити висновки, що зменшемо валідацію та розмір пакета покращюють результати (Accurate 0.6294, precision: 0.671, recall: 0.535; F1 score: 0.5956) та найгірший результат був отриманий під час використання регулізації L1, L2 (Accurate 0.5786, precision: 0.6656, recall: 0.4537; F1 score: 0.5396).

Рекомендації для покращення RNN:

Використання GRU: Замість традиційних RNN, використовуйте інші архітектури, наприклад GRU (Gated Recurrent Unit). Ці архітектури дозволяють ефективніше працювати з довгостроковими залежностями та уникнути проблеми зниклого градієнту.

Bidirectional RNN: Використовуйте бідирекційні RNN, які обробляють послідовності в обох напрямках. Це дозволяє моделі отримувати інформацію з обох кінців послідовності.

Зменшення довжини послідовності: Розгляньте можливість розділення довгих послідовностей на більш короткі або використання підходів, які дозволяють змінювати довжину вхідних послідовностей (наприклад, використання пулінгу).

Dropout та Batch Normalization: Використовуйте методи регуляризації, такі як dropout, щоб уникнути перенавчання. Також можна використовувати шари нормалізації партій (batch normalization) для стабілізації навчання.

Гіперпараметри: Тщательно налаштовуйте гіперпараметри, такі як розмір батча, швидкість навчання, кількість нейронів у шарах та інші параметри моделі.

Зменшення обчислювальної складності: Використовуйте меншу кількість нейронів або шарів, якщо обчислювальна складність є проблемою. Також розгляньте можливість використання покращених оптимізаторів.

Використання більш сучасних архітектур: Розгляньте використання більш сучасних архітектур, таких як Transformer, які стали популярними в обробці послідовностей.

Попередня обробка тексту: Переконайтеся, що попереднє обробка тексту виконана належним чином, включаючи токенізацію, видалення зайвої інформації та очищення даних.

**Контрольні питання**

1. **Що таке рекурентна нейронна мережа і у чому її основні відмінності від звичайних нейронних мереж?**

Рекурентні нейронні мережі (RNN) – це клас нейронних мереж, що були безпосередньо розроблені для обробки послідовностей даних. Ці мережі названі "рекурентними", тому що вони виконують ту саму задачу для кожного елемента послідовності, зберігаючи при цьому інформацію про попередній елемент. Тобто, вони мають "пам'ять", яка зберігає інформацію про те, що сталося раніше в послідовності.

Архітектура RNN: відмінності від традиційних нейронних мереж.

1. Структура з петлями: Найбільша відмінність RNN від звичайних нейронних мереж полягає в їх внутрішній структурі. У RNN є петлі, де вихід з одного нейрона може бути входом для того самого нейрона в наступному кроці часу. Ця петля дозволяє RNN зберігати інформацію (

2. Обробка послідовностей: Традиційні нейронні мережі приймають фіксований розмір входу та видають фіксований розмір виходу, тоді як RNN може працювати з послідовностями різної довжини.

3. Залежність від часу: Вихід RNN в даному часовому кроці залежить не тільки від поточного входу, але й від попередніх входів. Ця "пам'ять" дає можливість RNN враховувати контекст при обробці послідовностей.

4. Параметри: У традиційних нейронних мережах кожен шар може мати свої ваги та зміщення. У RNN, ваги між нейронами, які з'єднуються з петлями, зазвичай спільні на всіх часових кроках

**2. Які основні проблеми пов'язані з навчанням RNN?**

Основні проблеми пов'язані з RNN включають:

* Проблема зниклого градієнту (Vanishing Gradient Problem):

У тривалому часі, при навчанні RNN, градієнти можуть зникати або зростати експоненційно. Це може призводити до того, що нейрони на початку послідовності втрачають здатність вносити внесок у вихід.

* Проблема вибору оптимального розміру кроку (Step Size Problem):

Визначення оптимального кроку для обробки послідовності є непростим завданням, і великі чи малі кроки можуть призводити до проблем, таких як великі обчислювальні витрати або недостатню здатність моделі взагалі сприймати залежності між вхідними даними.

* Наявність збурень та варіантність довжини послідовності:

RNN може бути чутливою до наявності збурень або варіантності довжини вхідної послідовності, що робить їх менш ефективними в деяких завданнях.

* Проблема довгострокових залежностей:

Навчання довгострокових залежностей у RNN може бути важким завданням, оскільки градієнти можуть стають дуже малими або невизначеними під час зворотнього поширення помилки.

* Обчислювальна складність:

Для довгих послідовностей або складних мереж обчислювальна складність може бути дуже великою, що призводить до великих витрат часу на навчання та прогнозування.

* Витрати пам'яті:

RNN може вимагати багато пам'яті для зберігання інформації про попередній стан, особливо при великій кількості часових кроків.

* Артефакти на початку та кінці послідовностей:

За замовчуванням RNN має тенденцію надмірно акцентувати важливість останніх подій у послідовності, і це може призводити до забування важливої інформації на початку.

**3. Що таке LSTM та GRU? Які основні відмінності та схожості між ними?**

LSTM (Long Short-Term Memory) і GRU (Gated Recurrent Unit) є двома типами рекурентних нейронних мереж (RNN), розробленими для вирішення проблеми зниклого градієнту та покращення здатності моделі працювати з довгостроковими залежностями в послідовностях.

1. Схожості:

* Засновані на ідеї вентилів:

Обидві архітектури базуються на ідеї використання вентилів для регулювання інформаційного потоку в нейронних мережах.

* Призначені для роботи з послідовностями:

Як LSTM, так і GRU спеціалізуються на роботі з послідовностями та мають успішне використання в області обробки природної мови, машинного перекладу та інших завдань з обробки послідовностей.

* Здатні до роботи з довгостроковими залежностями:

Обидві архітектури розроблені для ефективної обробки довгострокових залежностей в послідовностях.

1. Відмінності:

* Кількість вентилів:

Основна відмінність полягає у кількості вентилів та загальній архітектурі. LSTM має більше вентилів, що може дозволяти йому ефективніше працювати зі складнішими залежностями, але може вимагати більше обчислювальних ресурсів.

* Витрати ресурсів:

GRU зазвичай вимагає менше обчислювальних ресурсів, оскільки має менше параметрів.

* Зберігання та забування інформації:

У LSTM є окремий вентиль забування, тоді як GRU використовує один вентиль оновлення, що регулює які частини інформації слід зберегти або забути.

1. **У яких задачах рекомендується використовувати RNN?**

Застосування RNN.

• Обробка тексту. Аналіз настрою: Визначення, чи є текст позитивним, негативним чи нейтральним. Наприклад, аналіз відгуків клієнтів про продукти чи послуги. Розпізнавання іменованих сутностей (NER, Named Entity Recognition): Виявлення осіб, організацій, місць тощо у тексті. Машинний переклад: Переклад тексту з однієї мови на іншу за допомогою послідовності до послідовності (Seq2Seq) моделей. Автоматичне відповідання на питання: Генерація відповідей на запитання на основі даного контексту.

• Прогнозування часових рядів: Прогноз погоди: Використання історичних даних про погоду для передбачення майбутніх погодних умов. Прогнозування акційних ринків: Прогнозування цін акцій на основі історичних даних. Виявлення аномалій: Виявлення відхилень у часових рядах, що може вказувати на несправності у системах або шахраї в фінансових транзакціях.

• Генерація тексту: Генерація віршів та прози: Створення вмісту, що наближений до людської мови, на основі навчання з великих текстових датасетів. Створення діалогових систем: Розробка чат-ботів, які можуть вести бесіду на певні теми. Автоматична генерація музики або сценаріїв: Створення музичних треків чи сценаріїв для фільмів на основі попереднього навчання. Інші застосування: Відео аналіз: Розпізнавання дій у відео, прогнозування наступних кадрів. Обробка мовлення: Розпізнавання мовлення, конвертація мовлення в текст. Біоінформатика: Прогнозування структури білків, геномний аналіз. Застосування RNN дуже різноманітні завдяки їх здатності обробляти послідовності даних. Проте, вони не є панацеєю, і іноді інші моделі, такі як трансформатори, можуть бути ефективнішими в певних задачах.

**5. Які існують альтернативи RNN для обробки послідовностей даних?**

* Long Short-Term Memory (LSTM):

Вже згадана, але залишається однією з основних архітектур для обробки послідовностей, завдяки своїм вентилям, які дозволяють зберігати та забувати інформацію.

* Gated Recurrent Unit (GRU):

Також була згадана раніше, GRU - спрощена версія LSTM з меншою кількістю вентилів, що може бути менш вимогливою до обчислювальних ресурсів.

* Bidirectional RNN:

Використовує два напрямки (вперед і назад) для обробки послідовностей. Вона може отримати інформацію з обох напрямків та бути корисною для завдань, де контекст із обох кінців важливий.

* Attention Mechanisms:

Замість того, щоб враховувати всю послідовність в кожному кроці, механізми уваги дозволяють моделі фокусуватися на певні частини послідовності.

* Transformer:

Архітектура, що вперше використовується в машинному перекладі. Вона базується на механізмах уваги та може ефективно обробляти послідовності паралельно, що полегшує шкалювання.

* Convolutional Neural Networks (CNN):

Зазвичай використовуються для обробки зображень, але можуть бути адаптовані для обробки послідовностей з використанням одновимірних згорток.

* Echo State Networks (ESN):

Неітеративні рекурентні мережі, які можуть обробляти послідовності зі сталими вагами.

* Neural Turing Machines (NTM), Differentiable Neural Computers (DNC):

Архітектури, які можуть вивчати імперативні або алгоритмічні завдання, що потребують роботи з пам'яттю та обчисленнями.