

```
# Завантаження та підготовка даних
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

# 1. Завантаження та підготовка даних
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/mushroom-epiotea.data"
columns = ('stalk-type', 'cap-shape', 'cap-surface', 'cap-color', 'stipe-color', 'stipe-shape',
           'gill-attachment', 'gill-spacing', 'gill-size', 'gill-color', 'stalk-shape',
           'stalk-root', 'stalk-surface-above-ring', 'stalk-surface-below-ring',
           'stalk-color-above-ring', 'stalk-color-below-ring', 'veil-type',
           'veil-color', 'ring-number', 'ring-type', 'spore-print-color',
           'population', 'habitat')

df = pd.read_csv(url, header=None, names=columns)

# Кодивання категоріальних змінних
le = LabelEncoder()
y = le.fit_transform(df['class'])

# Розподіл даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 2. Побудова моделі
model = Sequential([
    Dense(64, activation='relu', input_shape=X_train.shape[1:]),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid') # Вихідний шар для класифікації
])

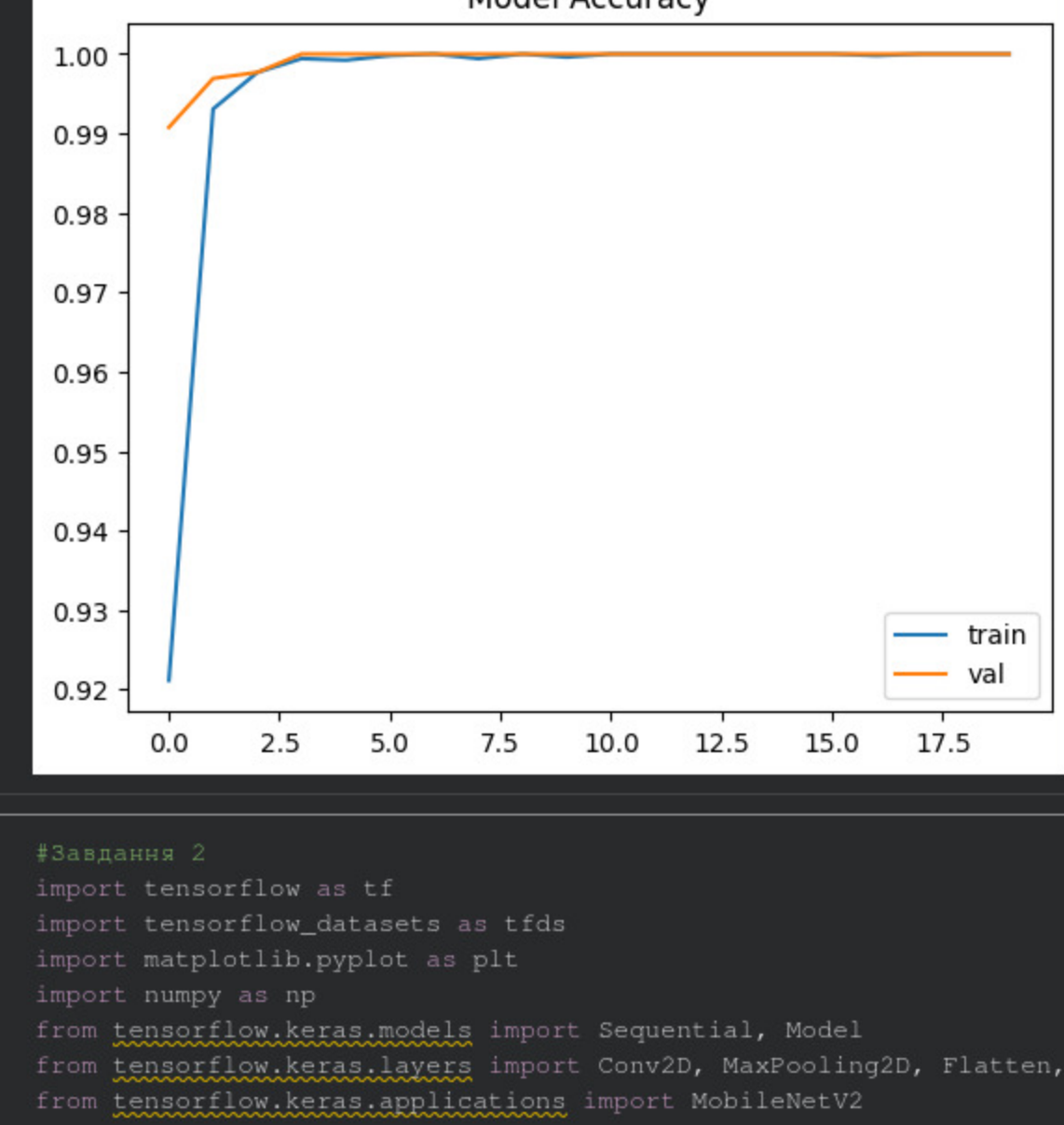
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

# 3. Навчання моделі
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_split=0.2, verbose=1)

# 4. Оцінка та виведення результатів
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

# 5. Виведення результатів
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
rf_acc = accuracy_score(y_test, rf.predict(X_test))
print(f"Random Forest Accuracy: {rf_acc:.4f}")

# 6. Виведення результатів
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='val')
plt.title('Model Accuracy')
plt.legend()
plt.show()
```



```
# Завантаження та підготовка даних
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, GlobalAveragePooling2D, Input
from tensorflow.keras.applications import MobileNetV2

# Завантаження даних
(ds_train, ds_val, ds_test), ds_info = tfds.load(
    'cats_vs_dogs',
    split=['train', 'validation', 'test'],
    shuffle_files=True,
    as_supervised=True, # Розподіл даних за класами (image, label)
    with_info=True
)

# Параметри
IMG_SIZE = 224 # Розмір для MobileNet
BATCH_SIZE = 32
NUM_CLASSES = 3

# Функція для обробки зображень
def preprocess(image, label):
    # Змінюємо розмір зображення на стандартний 224x224
    image = tf.image.resize(image, [IMG_SIZE, IMG_SIZE])
    # Нормалізуємо пікселі до діапазону [0, 1]
    image = image / 255.0
    # Розширюємо одиничний вектор до one-hot вектора (наприклад, 2 -> [0, 0, 1])
    label = tf.one_hot(label, depth=NUM_CLASSES)
    return image, label

# Розподіл даних на навчальний та тестовий набори
train_data = ds_train.map(preprocess).cache().shuffle(1000).batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
val_data = ds_val.map(preprocess).cache().batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
test_data = ds_test.map(preprocess).cache().batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)

print(f"Класи: {ds_info.features['label'].names}")

# 1. Простий CNN
model_simple = Sequential([
    Input(shape=(IMG_SIZE, IMG_SIZE, 3)),
    # 1-й шар
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    # 2-й шар
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    # 3-й шар
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    # Розгортання
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5), # Випадкова вибірка
    Dense(NUM_CLASSES, activation='softmax')
])

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

# Навчання моделі
history_simple = model_simple.fit(
    train_data,
    epochs=10,
    validation_data=val_data,
    verbose=1
)

# 2. TRANSFER LEARNING (MobileNetV2)
# Завантаження базової моделі з "frozen" (include_top=False)
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3),
                        include_top=False,
                        weights='imagenet')

# "Заморожуємо" базову частину моделі, щоб не втрачати те, що модель вже навчилася
base_model.trainable = False

# Додаємо нові шари
inputs = Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = base_model(inputs, training=False)
x = GlobalAveragePooling2D()(x) # Заміна Flatten, краще для сучасних мереж
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
outputs = Dense(NUM_CLASSES, activation='softmax')(x)

model_transfer = Model(inputs, outputs)

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

history_transfer = model_transfer.fit(
    train_data,
    epochs=10, # MobileNet вартує дуже швидко
    validation_data=val_data,
    verbose=1
)

# 3. ДОПОВИДНИЙ ТА ДОПОВИДНИЙ
# Оцінка на тестових даних
acc_simple = model_simple.evaluate(test_data)[1]
acc_transfer = model_transfer.evaluate(test_data)[1]

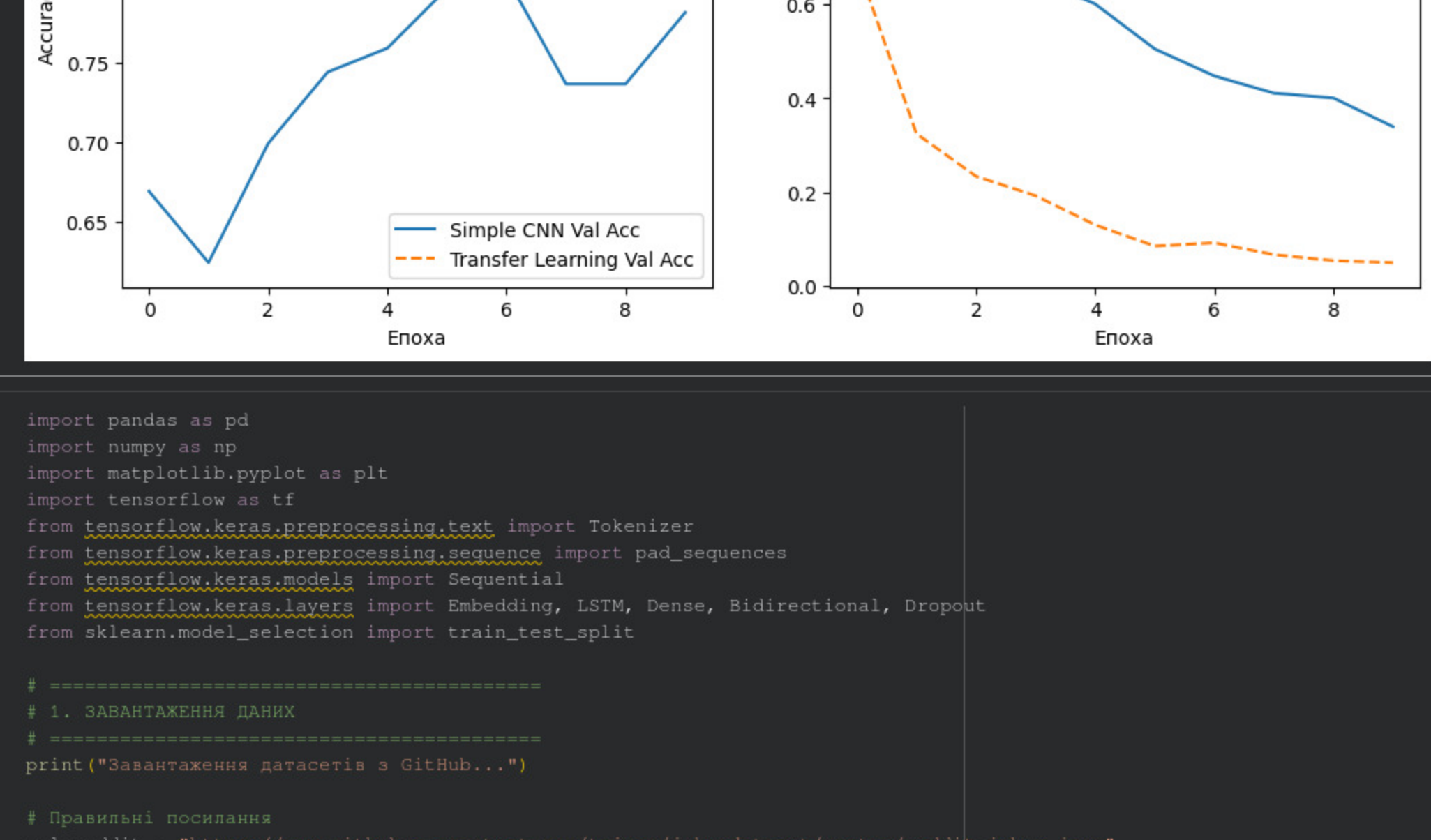
print(f"Точність простого CNN на тесті: {acc_simple:.2%}")
print(f"Точність Transfer Learning на тесті: {acc_transfer:.2%}")

# 4. Порівняння
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history_simple.history['val_accuracy'], label='Simple CNN Val Acc')
plt.plot(history_transfer.history['val_accuracy'], label='Transfer Learning Val Acc', linestyle='--')
plt.title('Порівняння точності (Validation)')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_simple.history['loss'], label='Simple CNN Loss')
plt.plot(history_transfer.history['loss'], label='Transfer Learning Loss', linestyle='--')
plt.title('Порівняння втрат (Loss)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense, Bidirectional, Dropout
from sklearn.model_selection import train_test_split

# 1. ЗАВАНТАЖЕННЯ ДАНИХ
# Завантаження даних з GitHub
url_reddit = "https://raw.githubusercontent.com/taivop/joke-dataset/master/reddit-jokes.json"
url_waka = "https://raw.githubusercontent.com/taivop/joke-dataset/master/waka.json"

df_reddit = pd.read_json(url_reddit)
df_waka = pd.read_json(url_waka)

# Створення мітки (label): 0 - Reddit, 1 - Waka
df_reddit['label'] = 0
df_waka['label'] = 1

# Розподіл даних на навчальний та тестовий набори
sample_size = 10000
df = pd.concat([
    df_reddit.sample(sample_size, random_state=42),
    df_waka.sample(min(len(df_waka), sample_size), random_state=42)
], ignore_index=True)

# Розподіл даних
df = df.sample(frac=1, random_state=42).reset_index(drop=True)

texts = df['body'].astype(str).tolist()
labels = df['label'].values

print(f"Кількість зразків: {len(texts)}")
print(f"Кількість класів: {len(set(labels))}")

# 2. ПЕРЕТВОРЕННЯ (TOKENIZATION)
VOCAB_SIZE = 10000 # Кількість слів у словнику
MAX_LENGTH = 40 # Максимальна довжина рядка
EMBEDDING_DIM = 100 # Розмір вектора емоцій (згідно GloVe 100d)

tokenizer = Tokenizer(num_words=VOCAB_SIZE, oov_token='<OOV>')
tokenizer.fit_on_texts(texts)

sequences = tokenizer.texts_to_sequences(texts)
padded = pad_sequences(sequences, maxlen=MAX_LENGTH, padding='post', truncating='post')

# Розподіл даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(padded, labels, test_size=0.2, random_state=42)

# 3. МОДЕЛЬ A: БАЗОВА МОДЕЛЬ
model_a = Sequential([
    Embedding(VOCAB_SIZE, EMBEDDING_DIM, input_length=MAX_LENGTH),
    Bidirectional(LSTM(64, return_sequences=False)),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model_a.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

history_a = model_a.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test), batch_size=32, verbose=1)

# 4. МОДЕЛЬ B: GLOVE EMBEDDINGS
# Завантаження матриці емоцій
url_glove = "https://nlp.stanford.edu/data/glove.6B.zip"
!wget -q -O glove.6B.zip $url_glove
!unzip -q -o glove.6B.zip

# Створення матриці емоцій
embeddings_index = {}
with open('glove.6B.100d.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs

# Створення матриці емоцій
embedding_matrix = np.zeros((VOCAB_SIZE, EMBEDDING_DIM))
for word, i in tokenizer.word_index.items():
    if i < VOCAB_SIZE:
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector

print(f"Матриця емоцій B (Glove Pre-trained) ---")

model_b = Sequential([
    Embedding(VOCAB_SIZE, EMBEDDING_DIM, input_length=MAX_LENGTH,
              weights=[embedding_matrix, trainable=False]),
    Bidirectional(LSTM(64)),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model_b.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

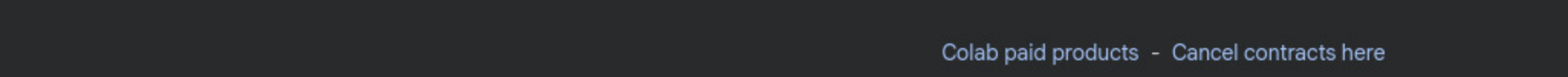
history_b = model_b.fit(X_train, y_train, epochs=5, validation_data=(X_test, y_test), batch_size=32, verbose=1)

# 5. ПОРІВНЯННЯ
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history_a.history['val_accuracy'], label='Model A (Scratch)')
plt.plot(history_b.history['val_accuracy'], label='Model B (Glove)', linestyle='--')
plt.title('Порівняння точності (Validation)')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history_a.history['loss'], label='Model A (Scratch)')
plt.plot(history_b.history['loss'], label='Model B (Glove)', linestyle='--')
plt.title('Порівняння втрат (Loss)')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Точність моделі A (Scratch): 0.8110
Точність моделі B (Glove): 0.8770

Створення матриці емоцій...
Model A (Scratch) ---
Epoch 1/5
500/500 --- 187s 365ms/step - accuracy: 0.7418 - loss: 0.5158 - val_accuracy: 0.8360 - val_loss: 0.3678
Epoch 2/5
500/500 --- 162s 364ms/step - accuracy: 0.8429 - loss: 0.3547 - val_accuracy: 0.8659 - val_loss: 0.3081
Epoch 3/5
500/500 --- 165s 376ms/step - accuracy: 0.8682 - loss: 0.2945 - val_accuracy: 0.8770 - val_loss: 0.2864
Epoch 4/5
500/500 --- 188s 370ms/step - accuracy: 0.8926 - loss: 0.2606 - val_accuracy: 0.8748 - val_loss: 0.2636
Epoch 5/5
500/500 --- 203s 372ms/step - accuracy: 0.8994 - loss: 0.2361 - val_accuracy: 0.8720 - val_loss: 0.2702