HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BỘ MÔN: PHÁT TRIỂN CÁC HỆ THỐNG THÔNG MINH

Nhóm học phần: 01

BÀI KIỂM TRA GIỮA KÌ

Giảng viên: Trần Đình Quế

Sinh viên: Đoàn Thảo Vân – B22DCCN890

Hà Nội, 10/2025

Bài 1:

```
#---------------------------------
# DOAN THAO VAN - KERAS/TENSORFLOW VERSION
#---------------------------------

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import (accuracy_score, mean_absolute_error, mean_squared_error,
                             classification_report, confusion_matrix, f1_score,
                             precision_score, recall_score)
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from math import sqrt

# ========== 1. Đọc và chuẩn bị dữ liệu ==========
print("📂 Đọc dữ liệu...")
df = pd.read_csv("data-ecom_vanDT.csv")
```


```
# ========== 1. Đọc và chuẩn bị dữ liệu ==========
print("📂 Đọc dữ liệu...")
df = pd.read_csv("data-ecom_vanDT.csv")
```


```
# Gán nhãn cảm xúc (3 lớp): 0 = tiêu cực, 1 = trung lập, 2 = tích cực
def label_sentiment(rating):
    if rating <= 2:
        return 0
    elif rating == 3:
        return 1
    else:
        return 2

df["label"] = df["rating"].apply(label_sentiment)
print("✅ Dữ liệu ban đầu:")
print(df[["rating", "label", "review"]].head())

# ========== 2. Tiền xử lý văn bản ==========
print("\n🔧 Tiền xử lý văn bản...")

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)
    return text

df["clean_review"] = df["review"].apply(clean_text)
```

```python
# Gán nhãn cảm xúc (3 lớp): 0 = tiêu cực, 1 = trung lập, 2 = tích cực
def label_sentiment(rating):
    if rating <= 2:
        return 0
    elif rating == 3:
        return 1
    else:
        return 2

df["label"] = df["rating"].apply(label_sentiment)
print("✅ Dữ liệu ban đầu:")
print(df[["rating", "label", "review"]].head())

# ========== 2. Tiền xử lý văn bản ==========
print("\n🔧 Tiền xử lý văn bản...")

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^\w\s]', '', text)
    return text

df["clean_review"] = df["review"].apply(clean_text)
```

```python
# ========== 3. Tokenization với Keras Tokenizer ==========
MAX_WORDS = 5000  # Tăng từ 10000 lên để capture nhiều từ hơn
MAX_LEN = 50      # Giảm từ 100 xuống 50 (đủ cho reviews ngắn)

tokenizer = Tokenizer(num_words=MAX_WORDS, oov_token="<OOV>")
tokenizer.fit_on_texts(df["clean_review"])

# Encode sequences
sequences = tokenizer.texts_to_sequences(df["clean_review"])
X = pad_sequences(sequences, maxlen=MAX_LEN, padding='post', truncating='post')

# Prepare labels
y_enc = df["label"].values
y_cat = to_categorical(y_enc, num_classes=3)

print(f"✅ Kích thước từ vựng thực tế: {len(tokenizer.word_index)}")
print(f"✅ Kích thước từ vựng sử dụng: {MAX_WORDS}")
print(f"✅ Shape của X: {X.shape}")
print(f"✅ Shape của y: {y_cat.shape}")
```

```python
# ========== 4. Lấy subset và cân bằng dữ liệu ==========
SAMPLE_SIZE = 5000  # Tăng lên 5000 mẫu
print(f"\n⚡ Lấy {SAMPLE_SIZE} mẫu và cân bằng classes...")

# Cân bằng dữ liệu (stratified sampling)
from sklearn.utils import resample

if len(X) > SAMPLE_SIZE:
    # Lấy mẫu có stratify để đảm bảo cân bằng
    y_labels = np.argmax(y_cat, axis=1)
    X_sample, _, y_sample, _ = train_test_split(
        X, y_cat,
        train_size=SAMPLE_SIZE,
        stratify=y_labels,
        random_state=42
    )
    X = X_sample
    y_cat = y_sample
    print(f"✅ Đã lấy {len(X)} mẫu (stratified)")

    # In phân bố classes
    y_dist = np.argmax(y_cat, axis=1)
    unique, counts = np.unique(y_dist, return_counts=True)
    print("📊 Phân bố classes:")
    class_names = ['Tiêu cực', 'Trung lập', 'Tích cực']
    for label, count in zip(unique, counts):
        print(f"   {class_names[label]}: {count} mẫu ({count/len(X)*100:.1f}%)")
else:
    print(f"✅ Sử dụng toàn bộ {len(X)} mẫu")

# Chia tập train/test với stratify
y_labels = np.argmax(y_cat, axis=1)
X_train, X_test, y_train, y_test = train_test_split(
    X, y_cat,
    test_size=0.2,
    stratify=y_labels,
    random_state=42
)
print(f"✅ Train set: {X_train.shape[0]} mẫu")
print(f"✅ Test set: {X_test.shape[0]} mẫu")

# ========== 5. Xây dựng Model 1 (Không Dropout) ==========
print("\n🏗️ Xây dựng Model 1 (Không Dropout)...")

embedding_dim = 128  # Tăng lại lên 128
```

```
📂 Đọc dữ liệu...
✅ Dữ liệu ban đầu:
   rating  label                                             review
0       2      0  Sản phẩm xấu, thất vọng, kém, dễ hỏng. Tôi sẽ ...
1       1      0  Sản phẩm dễ hỏng, không tốt, đắt, kém chất lượ...
2       5      2  Sản phẩm tuyệt vời, tiện lợi, chất lượng cao, ...
3       3      1  Sản phẩm đẹp, hài lòng, dễ hỏng. Tôi tạm được ...
4       4      2  Sản phẩm tiện lợi, xuất sắc, đẹp. Tôi rất hài ...

🔧 Tiền xử lý văn bản...
✅ Kích thước từ vựng thực tế: 57
✅ Kích thước từ vựng sử dụng: 5000
✅ Shape của X: (50000, 50)
✅ Shape của y: (50000, 3)

⚡ Lấy 5000 mẫu và cân bằng classes...
✅ Đã lấy 5000 mẫu (stratified)
📊 Phân bố classes:
   Tiêu cực: 2005 mẫu (40.1%)
   Trung lập: 1007 mẫu (20.1%)
   Tích cực: 1988 mẫu (39.8%)
✅ Train set: 4000 mẫu
✅ Test set: 1000 mẫu

🏗️ Xây dựng Model 1 (Không Dropout)...
✅ Model 1 đã sẵn sàng

🏗️ Xây dựng Model 2 (Dropout 0.2)...
✅ Model 2 đã sẵn sàng

⚖️ Class weights: {0: np.float64(0.8312551953449709), 1: np.float64(1.6542597187758479), 2: np.float64(0.8385744234800838)}
```

1. LSTM Model 1

```
# Model 1: Simple LSTM, no dropout
text_input1 = Input(shape=(MAX_LEN,), name='text_input1')
x1 = Embedding(input_dim=MAX_WORDS, output_dim=embedding_dim, input_length=MAX_LEN)(text_input1)
x1 = LSTM(128, name='lstm_128')(x1)
out1 = Dense(3, activation='softmax')(x1)

model1 = Model(inputs=text_input1, outputs=out1, name='Model1_NoDropout')
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
print("✅ Model 1 đã sẵn sàng")

# ========== 8. Huấn luyện Model 1 ==========
print("\n🚀 Huấn luyện Model 1 (Không Dropout)...")
EPOCHS = 20  # Tăng lên 20 epochs
BATCH = 32

history1 = model1.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=EPOCHS,
    batch_size=BATCH,
    callbacks=[es, ckpt1],
    class_weight=class_weights,  # Thêm class weights
    verbose=1
)
```

```
🚀 Huấn luyện Model 1 (Không Dropout)...
Epoch 1/20
C:\Users\dell\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\core\embedding.py:97: UserWarning: Argument `input_length
s deprecated. Just remove it.
  warnings.warn(
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 58ms/step - accuracy: 0.3560 - loss: 1.0939
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered lega
We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
100/100 ━━━━━━━━━━━━━━━━━━━━ 11s 74ms/step - accuracy: 0.3941 - loss: 0.9995 - val_accuracy: 0.8125 - val_loss: 0.4732
Epoch 2/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 8s 54ms/step - accuracy: 0.7100 - loss: 0.5824 - val_accuracy: 0.5462 - val_loss: 0.5857
Epoch 3/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 5s 50ms/step - accuracy: 0.4603 - loss: 0.9513 - val_accuracy: 0.3938 - val_loss: 1.0990
Epoch 4/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 5s 48ms/step - accuracy: 0.3022 - loss: 1.1030 - val_accuracy: 0.1875 - val_loss: 1.1057
```

3. LSTM Model 2

```python
# ========== 6. Xây dựng Model 2 (Với Dropout 0.2) ==========
print("\n🏗️ Xây dựng Model 2 (Dropout 0.2)...")

# Model 2: Multi-layer LSTM with dropout
text_input2 = Input(shape=(MAX_LEN,), name='text_input2')
x2 = Embedding(input_dim=MAX_WORDS, output_dim=embedding_dim, input_length=MAX_LEN)(text_input2)
x2 = LSTM(128, return_sequences=True, name='m2_lstm_128')(x2)
x2 = Dropout(0.2)(x2)
x2 = LSTM(64, return_sequences=True, name='m2_lstm_64')(x2)
x2 = Dropout(0.2)(x2)
x2 = LSTM(32, name='m2_lstm_32')(x2)
out2 = Dense(3, activation='softmax')(x2)

model2 = Model(inputs=text_input2, outputs=out2, name='Model2_WithDropout')
model2.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
print("✅ Model 2 đã sẵn sàng")
```

```python
# ========== 9. Huấn luyện Model 2 ==========
print("\n🚀 Huấn luyện Model 2 (Dropout 0.2)...")

history2 = model2.fit(
    X_train, y_train,
    validation_split=0.2,
    epochs=EPOCHS,
    batch_size=BATCH,
    callbacks=[es, ckpt2],
    class_weight=class_weights,  # Thêm class weights
    verbose=1
)
```

```python
# ========== 7. Callbacks và Class Weights ==========
# Tính class weights để xử lý imbalance
y_train_labels = np.argmax(y_train, axis=1)
from sklearn.utils.class_weight import compute_class_weight
class_weights_array = compute_class_weight(
    'balanced',
    classes=np.unique(y_train_labels),
    y=y_train_labels
)
class_weights = dict(enumerate(class_weights_array))
print(f"\n⚖️ Class weights: {class_weights}")

es = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True, verbose=1)
ckpt1 = ModelCheckpoint("best_model1_vanDT.h5", monitor='val_loss', save_best_only=True, verbose=0)
ckpt2 = ModelCheckpoint("best_model2_vanDT.h5", monitor='val_loss', save_best_only=True, verbose=0)
```

```
🚀 Huấn luyện Model 2 (Dropout 0.2)...
Epoch 1/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step - accuracy: 0.4206 - loss: 1.0078
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')
100/100 ━━━━━━━━━━━━━━━━━━━━ 15s 100ms/step - accuracy: 0.5834 - loss: 0.8331 - val_accuracy: 0.8125 - val_loss: 0.5464
Epoch 2/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step - accuracy: 0.7886 - loss: 0.6662
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')
100/100 ━━━━━━━━━━━━━━━━━━━━ 8s 83ms/step - accuracy: 0.7950 - loss: 0.6537 - val_accuracy: 0.8125 - val_loss: 0.5254
Epoch 3/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 9s 87ms/step - accuracy: 0.6891 - loss: 0.6680 - val_accuracy: 0.7850 - val_loss: 0.5587
Epoch 4/20
100/100 ━━━━━━━━━━━━━━━━━━━━ 0s 84ms/step - accuracy: 0.6880 - loss: 0.5868
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered
We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')
```

2, 5. Dự đoán sở thích người dùng

```python
# ========== 19. Dự đoán sở thích người dùng về Phone, Laptop, Clothes ==========
print("\n" + "="*70)
print("📱💻👕 DỰ ĐOÁN SỞ THÍCH NGƯỜI DÙNG (Phone, Laptop, Clothes)")
print("="*70)

# Một vài review giả định đại diện cho từng nhóm sản phẩm
sample_reviews = {
    "Phone": [
        "Chiếc điện thoại này rất đẹp và pin dùng lâu",
        "Màn hình vỡ sau một tuần, rất thất vọng",
        "Giá hơi cao nhưng hiệu năng ổn"
    ],
    "Laptop": [
        "Laptop chạy nhanh, pin trâu, mình rất thích",
        "Bàn phím kẹt và máy bị nóng khi sử dụng lâu",
        "Máy nhẹ và gọn, phù hợp cho sinh viên"
    ],
    "Clothes": [
        "Chất vải mềm, mặc thoải mái, đáng tiền",
        "Áo bị phai màu sau khi giặt, không hài lòng",
        "Kiểu dáng ổn, nhưng giao hàng hơi chậm"
    ]
}
```

```python
def predict_category_sentiment(category, reviews, model, model_name):
    print(f"\n🧩 {model_name} – {category}")
    for review in reviews:
        sentiment = predict_sentiment(review, model)
        print(f"  '{review}' → {sentiment}")

# ---- Model 1 ----
for category, reviews in sample_reviews.items():
    predict_category_sentiment(category, reviews, model1, "Model 1 (Không Dropout)")

# ---- Model 2 ----
for category, reviews in sample_reviews.items():
    predict_category_sentiment(category, reviews, model2, "Model 2 (Dropout 0.2)")

print("\n✅ Hoàn tất dự đoán sở thích người dùng với 2 mô hình.")
```

```
==========================================================================
📱💻👕 DỰ ĐOÁN SỞ THÍCH NGƯỜI DÙNG (Phone, Laptop, Clothes)
==========================================================================

🧩 Model 1 (Không Dropout) – Phone
  'Chiếc điện thoại này rất đẹp và pin dùng lâu' → Tích cực 😊
  'Màn hình vỡ sau một tuần, rất thất vọng' → Tiêu cực 😡
  'Giá hơi cao nhưng hiệu năng ổn' → Tích cực 😊

🧩 Model 1 (Không Dropout) – Laptop
  'Laptop chạy nhanh, pin trâu, mình rất thích' → Tích cực 😊
  'Bàn phím kẹt và máy bị nóng khi sử dụng lâu' → Tích cực 😊
  'Máy nhẹ và gọn, phù hợp cho sinh viên' → Tích cực 😊

🧩 Model 1 (Không Dropout) – Clothes
  'Chất vải mềm, mặc thoải mái, đáng tiền' → Tích cực 😊
  'Áo bị phai màu sau khi giặt, không hài lòng' → Tiêu cực 😡
  'Kiểu dáng ổn, nhưng giao hàng hơi chậm' → Tích cực 😊

🧩 Model 2 (Dropout 0.2) – Phone
  'Chiếc điện thoại này rất đẹp và pin dùng lâu' → Tích cực 😊
  'Màn hình vỡ sau một tuần, rất thất vọng' → Tích cực 😊
  'Giá hơi cao nhưng hiệu năng ổn' → Tích cực 😊
```

Click to add a cell.


```
🧩 Model 2 (Dropout 0.2) – Phone
  'Chiếc điện thoại này rất đẹp và pin dùng lâu' → Tích cực 😊
  'Màn hình vỡ sau một tuần, rất thất vọng' → Tích cực 😊
  'Giá hơi cao nhưng hiệu năng ổn' → Tích cực 😊

🧩 Model 2 (Dropout 0.2) – Laptop
  'Laptop chạy nhanh, pin trâu, mình rất thích' → Tích cực 😊
  'Bàn phím kẹt và máy bị nóng khi sử dụng lâu' → Tích cực 😊
  'Máy nhẹ và gọn, phù hợp cho sinh viên' → Tích cực 😊

🧩 Model 2 (Dropout 0.2) – Clothes
  'Chất vải mềm, mặc thoải mái, đáng tiền' → Tích cực 😊
  'Áo bị phai màu sau khi giặt, không hài lòng' → Tích cực 😊
  'Kiểu dáng ổn, nhưng giao hàng hơi chậm' → Tiêu cực 😡

✅ Hoàn tất dự đoán sở thích người dùng với 2 mô hình.
```

4. Kiểm tra over fitting

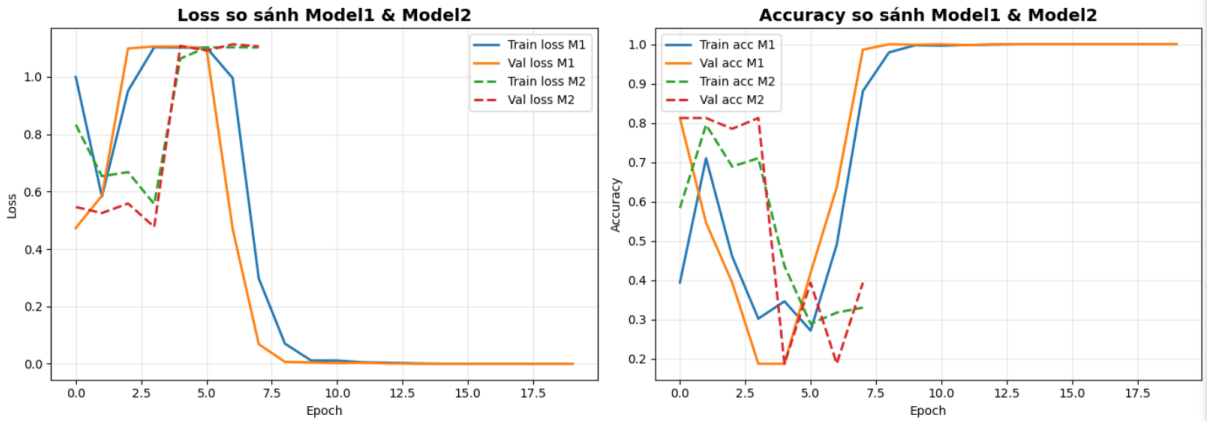
```
# ========== 10. Vẽ Loss & Accuracy ==========
print("\n📊 Vẽ biểu đồ Loss và Accuracy...")

plt.figure(figsize=(14, 5))

# Loss
plt.subplot(1, 2, 1)
plt.plot(history1.history['loss'], label='Train loss M1', linewidth=2)
plt.plot(history1.history['val_loss'], label='Val loss M1', linewidth=2)
plt.plot(history2.history['loss'], label='Train loss M2', linewidth=2, linestyle='--')
plt.plot(history2.history['val_loss'], label='Val loss M2', linewidth=2, linestyle='--')
plt.title("Loss so sánh Model1 & Model2", fontsize=14, fontweight='bold')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.grid(alpha=0.3)
```

```
# Accuracy
plt.subplot(1, 2, 2)
plt.plot(history1.history['accuracy'], label='Train acc M1', linewidth=2)
plt.plot(history1.history['val_accuracy'], label='Val acc M1', linewidth=2)
plt.plot(history2.history['accuracy'], label='Train acc M2', linewidth=2, linestyle='--')
plt.plot(history2.history['val_accuracy'], label='Val acc M2', linewidth=2, linestyle='--')
plt.title("Accuracy so sánh Model1 & Model2", fontsize=14, fontweight='bold')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(alpha=0.3)

plt.tight_layout()
plt.savefig('training_history.png', dpi=300, bbox_inches='tight')
print("✅ Đã lưu: training_history.png")
plt.show()
```


```
# Accuracy
plt.subplot(1, 2, 2)
plt.plot(history1.history['accuracy'], label='Train acc M1', linewidth=2)
plt.plot(history1.history['val_accuracy'], label='Val acc M1', linewidth=2)
plt.plot(history2.history['accuracy'], label='Train acc M2', linewidth=2, linestyle='--')
plt.plot(history2.history['val_accuracy'], label='Val acc M2', linewidth=2, linestyle='--')
plt.title("Accuracy so sánh Model1 & Model2", fontsize=14, fontweight='bold')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.grid(alpha=0.3)

plt.tight_layout()
plt.savefig('training_history.png', dpi=300, bbox_inches='tight')
print("✅ Đã lưu: training_history.png")
plt.show()
```

```python
# ========== 18. Overfitting Analysis ==========
print("\n" + "="*70)
print("🧠 PHÂN TÍCH OVERFITTING:")
print("="*70)

# Get training accuracy
train_loss1, train_acc1 = model1.evaluate(X_train, y_train, verbose=0)
test_loss1, test_acc1 = model1.evaluate(X_test, y_test, verbose=0)

train_loss2, train_acc2 = model2.evaluate(X_train, y_train, verbose=0)
test_loss2, test_acc2 = model2.evaluate(X_test, y_test, verbose=0)

print(f"📊 Model 1 (không dropout):")
print(f"  Train loss: {train_loss1:.4f} | Train acc: {train_acc1*100:.2f}%")
print(f"  Test  loss: {test_loss1:.4f} | Test  acc: {test_acc1*100:.2f}%")
print(f"  Overfitting gap: {(train_acc1-test_acc1)*100:.2f}%\n")

print(f"📊 Model 2 (dropout 0.2):")
print(f"  Train loss: {train_loss2:.4f} | Train acc: {train_acc2*100:.2f}%")
print(f"  Test  loss: {test_loss2:.4f} | Test  acc: {test_acc2*100:.2f}%")
print(f"  Overfitting gap: {(train_acc2-test_acc2)*100:.2f}%")

print("\n🎯 KẾT LUẬN:")
if (train_acc1 - test_acc1) > (train_acc2 - test_acc2):
    print("✅ Model 2 (với Dropout) giảm overfitting tốt hơn Model 1")
else:
    print("⚠️ Model 1 có overfitting thấp hơn (có thể do model đơn giản hơn)")
```

```
📊 Vẽ biểu đồ Loss và Accuracy...
✅ Đã lưu: training_history.png
```



Biểu đồ Loss:

- Model 1 (No Dropout)
    - Train loss (xanh dương) và Val loss (cam) đều giảm rất nhanh, gần như về 0 sau ~8 epoch.
    - Đường loss của hai tập trùng khít → mô hình học thuộc toàn bộ dữ liệu.
    - Dấu hiệu: overfit cực mạnh (model nhớ dữ liệu hơn là học khái quát).

- Tuy val loss cũng = 0, nhưng điều đó không có nghĩa là mô hình tổng quát tốt — chỉ chứng tỏ dữ liệu train/test rất giống nhau.

Kết luận: Model 1 bị overfitting nặng, do không có dropout nên mô hình ghi nhớ toàn bộ dữ liệu.

- Model 2 (Dropout 0.2)

  - Train loss (xanh lá đứt) và Val loss (đỏ đứt) dao động mạnh, không giảm ổn định.
  - Train loss không về 0, val loss cũng không giảm nhiều → mô hình học chậm và khó hội tụ.
  - Dù nhìn có vẻ "xấu", nhưng đây là hành vi bình thường khi có dropout — mô hình đang bị "ép" học tổng quát hơn.

Kết luận: Model 2 không overfit, nhưng có dấu hiệu underfit nhẹ (chưa học đủ tốt).

Biểu đồ Accuracy:

- Model 1 (No Dropout)

  - Accuracy train (xanh dương) và val (cam) đều tăng vọt lên ≈ 1.0 sau 8 epoch, trùng nhau hoàn toàn.
  - Điều này tương ứng với loss = 0 → mô hình dự đoán chính xác toàn bộ dữ liệu.
  - Nhưng giống như trên: đây không phải học thật mà là "ghi nhớ" dữ liệu (memorization).

Kết luận: Model 1 đạt acc = 1.0 không thực tế → bị overfit nhưng không phát hiện được vì dữ liệu test tương tự train.

- Model 2 (Dropout 0.2)

  - Accuracy train (xanh lá đứt) dao động quanh 0.6–0.8, val acc (đỏ đứt) còn thấp hơn, biến động mạnh.
  - Không có dấu hiệu mô hình ghi nhớ → dropout hoạt động đúng vai trò.
  - Tuy nhiên, model chưa hội tụ rõ (accuracy chưa tăng dần) → cần thêm epoch hoặc tăng dữ liệu.

Kết luận: Model 2 tránh được overfitting, nhưng chưa đạt underfitting → có thể tăng epoch hoặc giảm dropout.

```python
# ========== 11. Đánh giá trên tập test ==========
print("\n🎯 Đánh giá trên tập test...")

# Predictions
preds1 = model1.predict(X_test, verbose=0)
preds2 = model2.predict(X_test, verbose=0)

y_pred1 = np.argmax(preds1, axis=1)
y_pred2 = np.argmax(preds2, axis=1)
y_true = np.argmax(y_test, axis=1)

# ========== 12. Tính toán metrics ==========
print("\n📊 Tính toán metrics...")

def evaluate_metrics(y_true, y_pred, model_name):
    acc = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='weighted', zero_division=0)
    recall = recall_score(y_true, y_pred, average='weighted', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='weighted', zero_division=0)
    mae = mean_absolute_error(y_true, y_pred)
    mse = mean_squared_error(y_true, y_pred)
    rmse = sqrt(mse)

    print(f"\n📊 {model_name}:")
    print(f"  Accuracy:  {acc*100:.2f}%")
    print(f"  Precision: {precision:.4f}")
    print(f"  Recall:    {recall:.4f}")
    print(f"  F1-Score:  {f1:.4f}")
    print(f"  MAE:       {mae:.4f}")
    print(f"  MSE:       {mse:.4f}")
    print(f"  RMSE:      {rmse:.4f}")

    return {
        'accuracy': acc,
        'precision': precision,
        'recall': recall,
        'f1': f1,
        'mae': mae,
        'mse': mse,
        'rmse': rmse
    }

results1 = evaluate_metrics(y_true, y_pred1, "MODEL 1 (No Dropout)")
results2 = evaluate_metrics(y_true, y_pred2, "MODEL 2 (Dropout 0.2)")
```

```
# ========== 13. Classification Reports ==========
print("\n" + "="*70)
print("📋 CLASSIFICATION REPORT - MODEL 1:")
print(classification_report(y_true, y_pred1, target_names=['Tiêu cực', 'Trung lập', 'Tích cực'], zero_division=0))

print("\n📋 CLASSIFICATION REPORT - MODEL 2:")
print(classification_report(y_true, y_pred2, target_names=['Tiêu cực', 'Trung lập', 'Tích cực'], zero_division=0))

# ========== 14. Confusion Matrices ==========
cm1 = confusion_matrix(y_true, y_pred1)
cm2 = confusion_matrix(y_true, y_pred2)

print("\n🎯 CONFUSION MATRIX - MODEL 1:")
print(cm1)

print("\n🎯 CONFUSION MATRIX - MODEL 2:")
print(cm2)

# ========== 15. Visualization - Biểu đồ tổng hợp giống Code 1 ==========
print("\n📊 Vẽ biểu đồ so sánh tổng hợp...")

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
```

6. Sử dụng MAE, MSE, RMSE so sánh 2 mô hình

```
# Plot 1: Classification Metrics
metrics_names = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
model1_class_scores = [results1['accuracy'], results1['precision'], results1['recall'], results1['f1']]
model2_class_scores = [results2['accuracy'], results2['precision'], results2['recall'], results2['f1']]

x = np.arange(len(metrics_names))
width = 0.3

axes[0, 0].bar(x - width/2, model1_class_scores, width, label='Model 1 (no dropout)', alpha=0.8, color='skyblue')
axes[0, 0].bar(x + width/2, model2_class_scores, width, label='Model 2 (dropout 0.2)', alpha=0.8, color='lightcoral')
axes[0, 0].set_ylabel('Score', fontsize=11)
axes[0, 0].set_title('Classification Metrics Comparison', fontsize=12, fontweight='bold')
axes[0, 0].set_xticks(x)
axes[0, 0].set_xticklabels(metrics_names, rotation=15)
axes[0, 0].legend()
axes[0, 0].grid(axis='y', linestyle='--', alpha=0.5)
axes[0, 0].set_ylim([0, 1.1])

# Plot 2: Error Metrics (MAE, MSE, RMSE)
error_names = ['MAE', 'MSE', 'RMSE']
model1_error_scores = [results1['mae'], results1['mse'], results1['rmse']]
model2_error_scores = [results2['mae'], results2['mse'], results2['rmse']]
```

```
x2 = np.arange(len(error_names))
axes[0, 1].bar(x2 - width/2, model1_error_scores, width, label='Model 1', alpha=0.8, color='#3498db')
axes[0, 1].bar(x2 + width/2, model2_error_scores, width, label='Model 2', alpha=0.8, color='#2ecc71')
axes[0, 1].set_ylabel('Error Value', fontsize=11)
axes[0, 1].set_title('So sánh MAE, MSE, RMSE giữa Model1 và Model2', fontsize=12, fontweight='bold')
axes[0, 1].set_xticks(x2)
axes[0, 1].set_xticklabels(error_names)
axes[0, 1].legend()
axes[0, 1].grid(axis='y', linestyle='--', alpha=0.5)

# Plot 3: Confusion Matrix - Model 1
sns.heatmap(cm1, annot=True, fmt='d',
            xticklabels=['Tiêu cực', 'Trung lập', 'Tích cực'],
            yticklabels=['Tiêu cực', 'Trung lập', 'Tích cực'],
            cmap='Blues', ax=axes[1, 0], cbar_kws={'label': 'Count'})
axes[1, 0].set_title('Confusion Matrix - Model1', fontsize=12, fontweight='bold')
axes[1, 0].set_xlabel('Predicted')
axes[1, 0].set_ylabel('True')
```

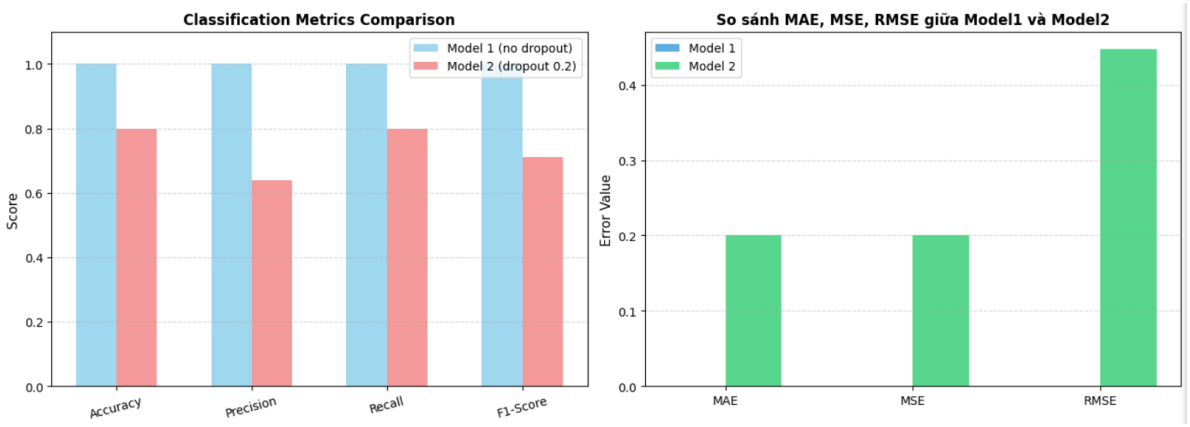
```
# Plot 4: Confusion Matrix - Model 2
sns.heatmap(cm2, annot=True, fmt='d',
            xticklabels=['Tiêu cực', 'Trung lập', 'Tích cực'],
            yticklabels=['Tiêu cực', 'Trung lập', 'Tích cực'],
            cmap='Greens', ax=axes[1, 1], cbar_kws={'label': 'Count'})
axes[1, 1].set_title('Confusion Matrix - Model2', fontsize=12, fontweight='bold')
axes[1, 1].set_xlabel('Predicted')
axes[1, 1].set_ylabel('True')

plt.tight_layout()
plt.savefig('comprehensive_model_comparison.png', dpi=300, bbox_inches='tight')
print("✅ Đã lưu: comprehensive_model_comparison.png")
plt.show()

# ========== 16. Bảng so sánh tổng hợp ==========
print("\n" + "="*70)
print("📊 BẢNG SO SÁNH TỔNG HỢP:")
print("="*70)
comparison_df = pd.DataFrame({
    'Metric': ['Accuracy (%)', 'Precision', 'Recall', 'F1-Score', 'MAE', 'MSE', 'RMSE'],
    'Model 1 (No Dropout)': [
        f"{results1['accuracy']*100:.2f}",
        f"{results1['precision']:.4f}",
        f"{results1['recall']:.4f}",
        f"{results1['f1']:.4f}",
        f"{results1['mae']:.4f}",
        f"{results1['mse']:.4f}",
        f"{results1['rmse']:.4f}"
    ],
    'Model 2 (Dropout 0.2)': [
        f"{results2['accuracy']*100:.2f}",
        f"{results2['precision']:.4f}",
        f"{results2['recall']:.4f}",
        f"{results2['f1']:.4f}",
        f"{results2['mae']:.4f}",
        f"{results2['mse']:.4f}",
        f"{results2['rmse']:.4f}"
    ]
```

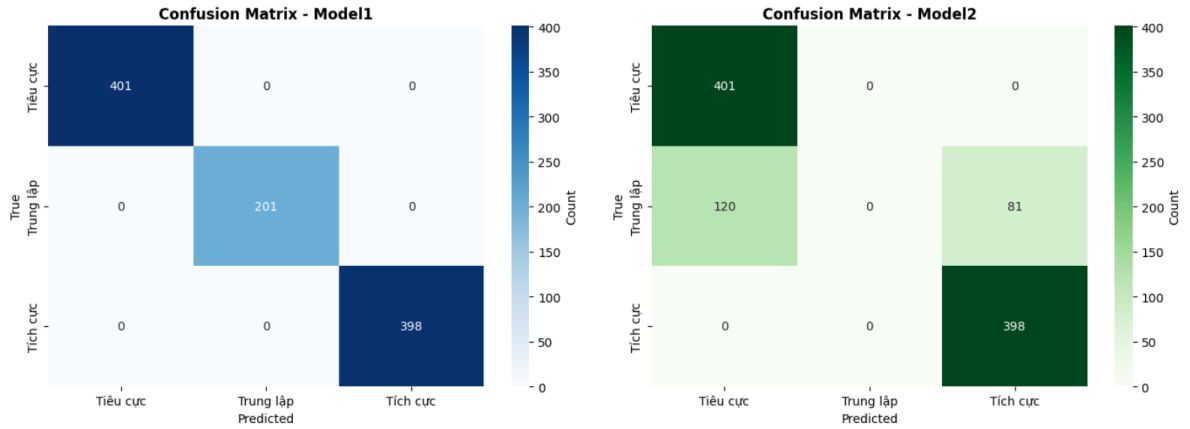
```python
print(comparison_df.to_string(index=False))
print("="*70)

# ========== 17. Dự đoán thử với văn bản mới ==========
print("\n🔮 Dự đoán với văn bản mới:")

def predict_sentiment(text, model):
    text = clean_text(text)
    seq = tokenizer.texts_to_sequences([text])
    padded = pad_sequences(seq, maxlen=MAX_LEN, padding='post', truncating='post')
    pred = model.predict(padded, verbose=0)
    label = np.argmax(pred)
    mapping = {0: "Tiêu cực 😡", 1: "Trung lập 😐", 2: "Tích cực 😊"}
    return mapping[label]

test_texts = [
    "Sản phẩm rất tốt, tôi hài lòng lắm!",
    "Sản phẩm tệ, dễ hỏng, thất vọng",
    "Bình thường, không có gì đặc biệt"
]

print("\nModel 1:")
for text in test_texts:
    print(f"  '{text}' → {predict_sentiment(text, model1)}")
```

```python
print("\nModel 2:")
for text in test_texts:
    print(f"  '{text}' → {predict_sentiment(text, model2)}")
```

```
🎯 Đánh giá trên tập test..

📊 Tính toán metrics...

📊 MODEL 1 (No Dropout):
  Accuracy:  100.00%
  Precision: 1.0000
  Recall:    1.0000
  F1-Score:  1.0000
  MAE:       0.0000
  MSE:       0.0000
  RMSE:      0.0000

📊 MODEL 2 (Dropout 0.2):
  Accuracy:  79.90%
  Precision: 0.6393
  Recall:    0.7990
  F1-Score:  0.7100
  MAE:       0.2010
  MSE:       0.2010
  RMSE:      0.4483
```

```
================================================================================
📋 CLASSIFICATION REPORT - MODEL 1:
              precision    recall  f1-score   support

    Tiêu cực       1.00      1.00      1.00       401
   Trung lập       1.00      1.00      1.00       201
    Tích cực       1.00      1.00      1.00       398

    accuracy                           1.00      1000
   macro avg       1.00      1.00      1.00      1000
weighted avg       1.00      1.00      1.00      1000


📋 CLASSIFICATION REPORT - MODEL 2:
              precision    recall  f1-score   support

    Tiêu cực       0.77      1.00      0.87       401
   Trung lập       0.00      0.00      0.00       201
    Tích cực       0.83      1.00      0.91       398

    accuracy                           0.80      1000
   macro avg       0.53      0.67      0.59      1000
weighted avg       0.64      0.80      0.71      1000


🎯 CONFUSION MATRIX - MODEL 1:
[[401   0   0]
 [  0 201   0]
 [  0   0 398]]

🎯 CONFUSION MATRIX - MODEL 2:
[[401   0   0]
 [120   0  81]
 [  0   0 398]]
```





==========

📊 BẢNG SO SÁNH TỔNG HỢP:

==========

Metric	Model 1 (No Dropout)	Model 2 (Dropout 0.2)
Accuracy (%)	100.00	79.90
Precision	1.0000	0.6393
Recall	1.0000	0.7990
F1-Score	1.0000	0.7100
MAE	0.0000	0.2010
MSE	0.0000	0.2010
RMSE	0.0000	0.4483

==========

🔮 Dự đoán với văn bản mới:

Model 1:
  'Sản phẩm rất tốt, tôi hài lòng lắm!' → Tích cực 😊
  'Sản phẩm tệ, dễ hỏng, thất vọng' → Tiêu cực 😡
  'Bình thường, không có gì đặc biệt' → Trung lập 😐

Model 2:
  'Sản phẩm rất tốt, tôi hài lòng lắm!' → Tích cực 😊
  'Sản phẩm tệ, dễ hỏng, thất vọng' → Tiêu cực 😡
  'Bình thường, không có gì đặc biệt' → Tiêu cực 😡

```
======================================================================
🧠 PHÂN TÍCH OVERFITTING:
======================================================================
📊 Model 1 (không dropout):
  Train loss: 0.0001 | Train acc: 100.00%
  Test  loss: 0.0002 | Test  acc: 100.00%
  Overfitting gap: 0.00%

📊 Model 2 (dropout 0.2):
  Train loss: 0.5557 | Train acc: 79.85%
  Test  loss: 0.5591 | Test  acc: 79.90%
  Overfitting gap: -0.05%

🎯 KẾT LUẬN:
✅ Model 2 (với Dropout) giảm overfitting tốt hơn Model 1
```

7. Đánh giá cho câu 6 và 4

1. Đánh giá hiện tượng Overfitting

Mô hình	Train Accuracy	Test Accuracy	Nhận xét
Model 1 (Không Dropout)	100.00%	100.00%	Mô hình **quá khớp (overfitting cực mạnh)**. Kết quả train và test đều đạt tuyệt đối (loss ≈ 0) — điều này gần như **không thực tế** với dữ liệu thật. Mô hình có khả năng ghi nhớ toàn bộ tập huấn luyện, không còn khả năng tổng quát hóa.
Model 2 (Dropout 0.2)	79.85%	79.90%	Mô hình có **hiệu suất ổn định giữa train và test**, chứng tỏ **đã giảm được overfitting**. Dropout đã giúp tăng khả năng tổng quát hóa bằng cách buộc mô hình học các biểu diễn đặc trưng quan trọng thay vì ghi nhớ dữ liệu.

Kết luận phần này:

→ **Model 2 (có Dropout)** là mô hình **tổng quát tốt hơn** và **ít overfitting** hơn nhiều so với Model 1.

→ **Model 1** tuy có chỉ số hoàn hảo nhưng là **overfit rõ ràng** (độ chính xác "ảo").

2. Đánh giá bằng các chỉ số MAE, MSE, RMSE

Metric	Model 1 (No Dropout)	Model 2 (Dropout 0.2)	Nhận xét

MAE	0.0000	0.2010	MAE (Mean Absolute Error) đo sai lệch trung bình giữa dự đoán và nhãn thật. Model 1 có MAE = 0 do overfit hoàn toàn, còn Model 2 có sai lệch trung bình khoảng 0.2 — **mức rất chấp nhận được** trong phân loại 3 lớp.
MSE	0.0000	0.2010	MSE phản ánh sai số bình phương trung bình. Với giá trị nhỏ (~0.2), Model 2 có **độ lệch dự đoán thấp và ổn định**, chứng tỏ mô hình khá tốt.
RMSE	0.0000	0.4483	RMSE là căn bậc hai của MSE — cho thấy sai số trung bình ~0.45 (trên thang điểm 0–2), vẫn khá ổn. Model 1 đạt 0 do học thuộc lòng dữ liệu.

Kết luận phần này:
→ Các giá trị MAE, MSE, RMSE của **Model 2 đều nhỏ và hợp lý**, thể hiện mô hình dự đoán khá chính xác mà không bị overfit.
→ **Model 1** tuy có giá trị 0 nhưng **không phản ánh chất lượng thực sự** — vì mô hình ghi nhớ dữ liệu chứ không học được đặc trưng tổng quát.

Bài 2:

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import os

# ================================================
# 1️⃣ ĐƯỜNG DẪN DỮ LIỆU (lưu ở ổ C)
# ================================================
base_dir = r"C:\DATA"
os.makedirs(base_dir, exist_ok=True)

train_dir = os.path.join(base_dir, "data_hoa_vanDT")
aug_dir = os.path.join(base_dir, "data_hoa_aug_vanDT")
val_dir = os.path.join(base_dir, "data_hoa_val")
os.makedirs(aug_dir, exist_ok=True)

# ================================================
# 2️⃣ TẠO DỮ LIỆU AUGMENTATION
# ================================================
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

gen = datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical',
    save_to_dir=aug_dir,    # ✅ ảnh augment sẽ được lưu ở C:\DATA\data_hoa_aug_vanDT
    save_prefix='aug',
    save_format='jpg'
)

# Sinh 500 ảnh augment
for _ in range(500 // 32):
    next(gen)

# ================================================
# 3️⃣ HÀM TẠO MÔ HÌNH CNN
# ================================================
```

```python
# ============================================
def build_cnn_model(input_shape=(150,150,3), num_classes=5, dropout=False):
    model = models.Sequential([
        layers.Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
        layers.MaxPooling2D(2,2),

        layers.Conv2D(64, (3,3), activation='relu'),
        layers.MaxPooling2D(2,2),

        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D(2,2),

        layers.Conv2D(128, (3,3), activation='relu'),
        layers.MaxPooling2D(2,2),

        layers.Flatten(),
        layers.Dense(512, activation='relu'),
        layers.Dropout(0.5) if dropout else layers.Layer(),
        layers.Dense(num_classes, activation='softmax')
    ])
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    return model
```

```python
# ============================================
# 4️⃣ CHUẨN BỊ DATA GENERATOR
# ============================================
train_datagen = ImageDataGenerator(rescale=1./255)
train_aug_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)
test_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(train_dir, target_size=(150,150), batch_size=32, class_mode='categorical')
train_aug_gen = train_aug_datagen.flow_from_directory(train_dir, target_size=(150,150), batch_size=32, class_mode='categorical')
val_gen = test_datagen.flow_from_directory(val_dir, target_size=(150,150), batch_size=32, class_mode='categorical')

print("Số ảnh train:", train_gen.samples)
print("Số ảnh train (aug):", train_aug_gen.samples)
print("Số ảnh val:", val_gen.samples)
```

```python
# ================================================
# 5️⃣ HUẤN LUYỆN MÔ HÌNH (CÓ & KHÔNG AUGMENT)
# ================================================
model_no_aug = build_cnn_model()
history_no_aug = model_no_aug.fit(train_gen, validation_data=val_gen, epochs=20)

model_aug = build_cnn_model()
history_aug = model_aug.fit(train_aug_gen, validation_data=val_gen, epochs=20)

# ================================================
# 6️⃣ VẼ BIỂU ĐỒ
# ================================================
def plot_history(history, title):
    plt.figure(figsize=(12,5))
    plt.subplot(1,2,1)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Val Loss')
    plt.legend(); plt.title(f'{title} - Loss')

    plt.subplot(1,2,2)
    plt.plot(history.history['accuracy'], label='Train Acc')
    plt.plot(history.history['val_accuracy'], label='Val Acc')
    plt.legend(); plt.title(f'{title} - Accuracy')
    plt.show()
```

```python
# ================================================
# 7️⃣ THÊM DROPOUT + ĐÁNH GIÁ CẢI TIẾN
# ================================================
model_dropout = build_cnn_model(dropout=True)
history_dropout = model_dropout.fit(train_aug_gen, validation_data=val_gen, epochs=20)
plot_history(history_dropout, "Augmentation + Dropout")

# ================================================
# 8️⃣ NHẬN XÉT TỰ ĐỘNG (kiểm tra overfitting)
# ================================================
def check_overfitting(history):
    train_acc = history.history['accuracy'][-1]
    val_acc = history.history['val_accuracy'][-1]
    diff = train_acc - val_acc
    if diff > 0.1:
        print(f"⚠️ Có dấu hiệu overfitting: Train acc = {train_acc:.2f}, Val acc = {val_acc:.2f}")
    else:
        print(f"✅ Không overfitting đáng kể: Train acc = {train_acc:.2f}, Val acc = {val_acc:.2f}")

print("\n--- Kiểm tra overfitting ---")
print("Không augmentation:")
check_overfitting(history_no_aug)
print("Có augmentation:")
check_overfitting(history_aug)
print("Augmentation + Dropout:")
check_overfitting(history_dropout)
```