

Họ tên: Ngô Tiến Hưng

Mã sinh viên: B22DCCN411

## Bài 1.

1 sử dụng LSTM để phân tích tập data\_ecom\_tensv.Hungnt  
kiến trúc LSTM

layer1:128 neuron

layer2:64 neuron

layer3:32 neuron

layer4:16neuron

( chú ý k dropout gọi model1)

2.sử dụng model1 để dự đoán sở thích gì dùng về phone, laptop, clothes

3. Xây dựng model2 với dropout 0,2 cho layer 1,2,3

4. Sử dụng loss và accuracy để kiểm tra overfitting của model1 và model2  
visualize

5 sử dụng model2 để dự đoán sở thích người dung phone, laptop, clothes

6. Sử dụng MEA, MSE, RMSE để so sánh 2 mô hình dùng visualize

7. Đánh giá bằng lời của mục 6,4

## Bài 2

1. Sử dụng augmentation (4 kỹ thuật) để tạo data\_hoa\_ang\_Hungnt và lưu vào C:\DATA ( print out 5 ảnh trước/sau)

2. Xây dựng mô hình CNN với 5 layer

3. Sử dụng biểu đồ để đánh giá loss/accuracy của model trên 2 tập data(có và không cùng) có overfit không? Nhận xét

4. Thêm dropout để xem có cải tiến không? Biểu đồ

## Bài 1:

### 1. Load dữ liệu, tiền xử lý và xây dựng Model1 (LSTM no Dropout)

#### Code

```
[1]: # Ngô Tiễn Hưng

# =====
# BÀI 1 - FULL (1 cell)
# Yêu cầu:
# 1) Xây model1 LSTM 4-layer (128-64-32-16) no dropout
# 2) Dùng model1 để dự đoán sở thích (phone, laptop, clothes)
# 3) Xây model2 giống model1 nhưng dropout=0.2 ở layer1,2,3
# 4) So sánh Loss/accuracy (visualize)
# 5) Dùng model2 để dự đoán sở thích
# 6) MAE, MSE, RMSE compare + visualize
# 7) Đánh giá bằng Lời (tự động sinh) – in theo khung
# =====

# 0. Imports & reproducibility
import os, random, math
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, accuracy_score, confusion_matrix
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Input, LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

seed = 42
np.random.seed(seed)
random.seed(seed)
tf.random.set_seed(seed)
os.environ['PYTHONHASHSEED'] = str(seed)

# File path (đã upload)
DATA_PATH = "C:/DATA/data-ecom_Hungnt.csv"

# -----
# MỤC 1: Load + tiền xử lý + mô tả model1
# -----
print("==== BÀI 1 - MỤC 1 ====")
print("■ Load dữ liệu, tiền xử lý và xây dựng Model1 (LSTM no Dropout)\n")
# Load CSV
df = pd.read_csv(DATA_PATH)
print(" - Shape dữ liệu:", df.shape)
print(" - Columns:", df.columns.tolist())
display(df.head(5))

# Detect target columns
target_cols = [c for c in ['phone','laptop','clothes','phone_pref','laptop_pref','clothes_pref'] if c in df.columns]
single_pref_col = None
if len(target_cols) != 3:
    for cand in ['preference','pref','category','fav','favorite']:
        if cand in df.columns:
            single_pref_col = cand
            break
# heuristic: if still none, look for object column with few unique values
single_pref_col = cand
break
if len(target_cols) != 3 and single_pref_col is None:
    for c in df.columns:
        if df[c].dtype == object and df[c].nunique() <= 10:
            single_pref_col = c
            break
|
print(" - target_cols detected:", target_cols)
print(" - single_pref_col detected:", single_pref_col)

# Prepare X and y robustly
df_proc = df.copy()
# drop id/date/time-like columns except targets
drop_cols = [c for c in df_proc.columns if ('id' in c.lower() or 'date' in c.lower() or 'time' in c.lower()) and c not in target_cols and c != single_pref_col]
if drop_cols:
    df_proc = df_proc.drop(columns=drop_cols, errors='ignore')
    print(" - dropped:", drop_cols)

# handle targets
if len(target_cols) == 3:
    # multi-label numeric expected (0/1 or probabilities)
    y = df_proc[target_cols].fillna(0).astype(float).values
    X_df = df_proc.drop(columns=target_cols)
    target_type = "multi_label"
    label_names = target_cols
elif single_pref_col is not None:
    y_raw = df_proc[single_pref_col].fillna('unknown').astype(str)
    le = LabelEncoder()
```

```

y_raw = df_proc[single_pref_col].fillna('unknown').astype(str)
le = LabelEncoder()
y_enc = le.fit_transform(y_raw)
class_names = list(le.classes_)
y = tf.keras.utils.to_categorical(y_enc) # one-hot
X_df = df_proc.drop(columns=[single_pref_col])
target_type = "multiclass"
label_names = class_names
else:
    raise ValueError("Không tìm thấy cột target. Hãy kiểm tra file CSV có cột phone/laptop/clothes hoặc preference.")

print(f" - Phát hiện target_type: {target_type}")
print(" - label names:", label_names)

# features: get dummies for categorical, keep numeric
X_proc = pd.get_dummies(X_df.fillna(0))
feature_cols = X_proc.columns.tolist()
print(" - feature count after dummies:", len(feature_cols))

# scale numeric features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_proc.values)

# LSTM expects 3D: (samples, timesteps, features). We'll use timesteps=1
X = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))
print(" - Input shape for LSTM (samples, timesteps, features):", X.shape)

# train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=seed, shuffle=True)
print(" - Train shape:", X_train.shape, y_train.shape)

def build_lstm(dropout_rate=0.0):
    inp_shape = (X_train.shape[1], X_train.shape[2])
    m = Sequential()
    m.add(Input(shape=inp_shape))
    m.add(LSTM(128, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(64, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(32, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(16, return_sequences=False))
    n_out = y_train.shape[1]
    # choose activation/loss based on y format
    if np.all((y_train==0) | (y_train==1)):
        m.add(Dense(n_out, activation='sigmoid'))
        loss = 'binary_crossentropy'
    else:
        m.add(Dense(n_out, activation='softmax'))
        loss = 'categorical_crossentropy'
    m.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3), loss=loss, metrics=['accuracy'])
    return m

modell = build_lstm(dropout_rate=0.0)
print("\nModell (no dropout) summary:")
modell.summary()

```

## Kết quả

```

==== BÀI 1 - MỤC 1 ====
[1]: Load dữ liệu, tiền xử lý và xây dựng Model (LSTM no Dropout)

- Shape dữ liệu: (50000, 6)
- Columns: ['user_id', 'item_id', 'item_name', 'category', 'rating', 'review_text']
   user_id  item_id  item_name  category  rating      review_text
0  user_1    item_95  Sản phẩm 95  Laptop     2  Sản phẩm kém, không giống mô tả.
1  user_1   item_362  Sản phẩm 362  Phụ kiện     2  Sản phẩm kém, không giống mô tả.
2  user_1   item_361  Sản phẩm 361  Laptop     5  Sản phẩm rất tốt, tôi hài lòng.
3  user_1    item_1  Sản phẩm 1  Điện thoại     1  Hàng lỗi, rất thất vọng.
4  user_1   item_72  Sản phẩm 72  Đồ gia dụng     3  Chất lượng tạm được, giá hơi cao.

- target_cols detected: []
- single_pref_col detected: category
- dropped: ['user_id', 'item_id']
- Phát hiện target_type: multiclass
- label names: ['Giày dép', 'Laptop', 'Phụ kiện', 'Quần áo', 'Điện thoại', 'Đồ gia dụng']
- feature count after dummies: 513
- Input shape for LSTM (samples, timesteps, features): (50000, 1, 513)
- Train shape: (40000, 1, 513) (40000, 6)
- Test shape: (10000, 1, 513) (10000, 6)

Modell (no dropout) summary:

Model: "sequential"

```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 128)	328,704
lstm_1 (LSTM)	(None, 1, 64)	49,408
lstm_2 (LSTM)	(None, 1, 32)	12,416
lstm_3 (LSTM)	(None, 16)	3,136
dense (Dense)	(None, 6)	182

Total params: 393,766 (1.50 MB)  
Trainable params: 393,766 (1.50 MB)  
Non-trainable params: 0 (0.00 B)

==== BÀI 1 - MỤC 2 ====  
[HUẤN LUYỆN MODEL1 & DỰ ĐOÁN SỞ THÍCH NGƯỜI DÙNG]

## 2. Sử dụng model1 để dự đoán sở thích (phone/laptop/clothes) — sẽ huấn luyện model trước rồi dự đoán)

```
# ===== BÀI 1 - MỤC 2 =====
# [ ] Sử dụng model1 để dự đoán sở thích (phone/laptop/clothes)

print("\n==== BÀI 1 - MỤC 2 ====")
print("[ ] HUẤN LUYỆN MODEL1 & DỰ ĐOÁN SỞ THÍCH NGƯỜI DÙNG\n")

# Huấn Luyện model1
history1 = model1.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=10,
    batch_size=64,
    verbose=1
)

# Dự đoán trên tập kiểm thử
y_pred1_prob = model1.predict(X_test)
y_pred1 = np.argmax(y_pred1_prob, axis=1)

# Hiển thị 10 dự đoán đầu tiên
decoded_preds = label_encoder.inverse_transform(y_pred1[:10])
decoded_true = label_encoder.inverse_transform(y_test[:10])

print("◆ 10 dự đoán đầu tiên (Model1):")
for i in range(10):
    print(f"Mẫu {i+1}: Dự đoán = {decoded_preds[i]} | Thực tế = {decoded_true[i]}")
```

```

Non-trainable params: 0 (0.00 B)

==== BÀI 1 - MỤC 2 ====
[HUẤN LUYỆN MODEL1 & DỰ DOAN SỐ THÍCH NGƯỜI DÙNG]

Epoch 1/10
625/625    14s 12ms/step - accuracy: 0.1656 - loss: 0.4731 - val_accuracy: 0.1609 - val_loss: 0.4507
Epoch 2/10
625/625    6s 9ms/step - accuracy: 0.1673 - loss: 0.4508 - val_accuracy: 0.1614 - val_loss: 0.4507
Epoch 3/10
625/625    6s 10ms/step - accuracy: 0.1685 - loss: 0.4508 - val_accuracy: 0.1603 - val_loss: 0.4507
Epoch 4/10
625/625    6s 10ms/step - accuracy: 0.1703 - loss: 0.4507 - val_accuracy: 0.1607 - val_loss: 0.4508
Epoch 5/10
625/625    6s 10ms/step - accuracy: 0.1763 - loss: 0.4504 - val_accuracy: 0.1645 - val_loss: 0.4512
Epoch 6/10
625/625    6s 10ms/step - accuracy: 0.1880 - loss: 0.4494 - val_accuracy: 0.1633 - val_loss: 0.4527
Epoch 7/10
625/625    11s 10ms/step - accuracy: 0.2042 - loss: 0.4473 - val_accuracy: 0.1648 - val_loss: 0.4554
Epoch 8/10
625/625    6s 10ms/step - accuracy: 0.2225 - loss: 0.4443 - val_accuracy: 0.1626 - val_loss: 0.4586
Epoch 9/10
625/625    6s 10ms/step - accuracy: 0.2397 - loss: 0.4409 - val_accuracy: 0.1653 - val_loss: 0.4610
Epoch 10/10
625/625    11s 11ms/step - accuracy: 0.2549 - loss: 0.4375 - val_accuracy: 0.1681 - val_loss: 0.4656
313/313    2s 6ms/step

```

### 3. Xây model2 giống model1 nhưng Dropout=0.2 ở layer1,2,3

Code:

```

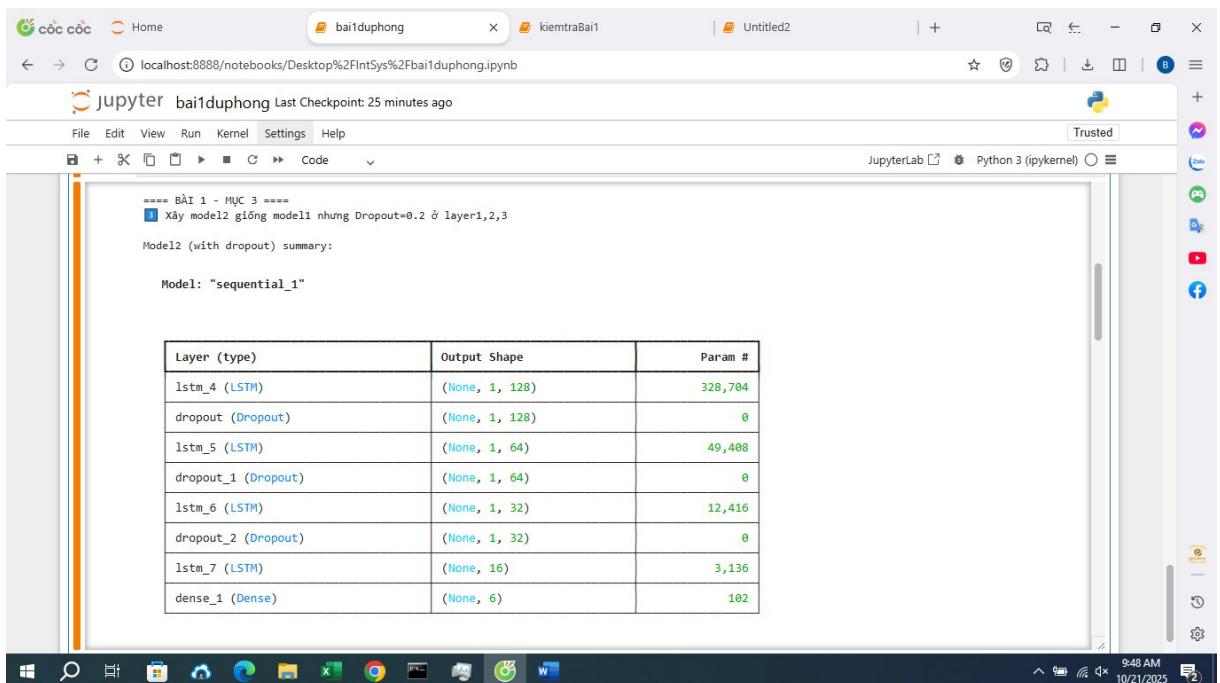
# Build model1 (no dropout) as requested
def build_lstm(dropout_rate=0.0):
    inp_shape = (X_train.shape[1], X_train.shape[2])
    m = Sequential()
    m.add(Input(shape=inp_shape))
    m.add(LSTM(128, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(64, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(32, return_sequences=True))
    if dropout_rate>0: m.add(Dropout(dropout_rate))
    m.add(LSTM(16, return_sequences=False))
    n_out = y_train.shape[1]
    # choose activation/loss based on y format
    if np.all((y_train==0) | (y_train==1)):
        m.add(Dense(n_out, activation='sigmoid'))
        loss = 'binary_crossentropy'
    else:
        m.add(Dense(n_out, activation='softmax'))
        loss = 'categorical_crossentropy'
    m.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3), loss=loss, metrics=['accuracy'])
    return m

model1 = build_lstm(dropout_rate=0.0)
print("\nModel1 (no dropout) summary:")

# -----
# MỤC 3: Xây model2 với dropout 0.2 (layer1,2,3)
# -----
print("\n==== BÀI 1 - MỤC 3 ====")
print("Xây model2 giống model1 nhưng Dropout=0.2 ở layer1,2,3\n")
model2 = build_lstm(dropout_rate=0.2)
print("Model2 (with dropout) summary:")
model2.summary()

```

Kết quả:



#### 4. Huấn luyện model1 & model2, dùng loss & accuracy để kiểm tra overfitting (visualize)

```
# MỤC 4: Huấn Luyện cả 2 model và visualize loss/accuracy để kiểm tra overfitting
# -----
print("===== BÀI 1 - MỤC 4 =====")
print("4 Huấn luyện model1 & model2, dùng loss & accuracy để kiểm tra overfitting (visualize)\n")

EPOCHS = 30
BATCH = 64
es = EarlyStopping(monitor='val_loss', patience=6, restore_best_weights=True, verbose=1)

# Train model1
print("→ Huấn luyện model1 (no dropout)...")
history1 = model1.fit(X_train, y_train, validation_split=0.1, epochs=EPOCHS, batch_size=BATCH, callbacks=[es], verbose=1)

# Train model2
print("→ Huấn luyện model2 (dropout=0.2)...")
history2 = model2.fit(X_train, y_train, validation_split=0.1, epochs=EPOCHS, batch_size=BATCH, callbacks=[es], verbose=1)

# Visualize Loss & Accuracy (combined)
plt.figure(figsize=(14,10))
# Loss
plt.subplot(2,1,1)
plt.plot(history1.history['loss'], label='m1 train loss', linewidth=2)
plt.plot(history1.history.get('val_loss', []), label='m1 val loss', linestyle='--')
plt.plot(history2.history['loss'], label='m2 train loss', linewidth=2)
plt.plot(history2.history.get('val_loss', []), label='m2 val loss', linestyle='--')
plt.title('Loss curves - model1 vs model2')
plt.xlabel('Epoch'); plt.ylabel('Loss'); plt.legend(); plt.grid(True)
# Accuracy
plt.subplot(2,1,2)

# Accuracy
plt.subplot(2,1,2)
plt.plot(history1.history.get('accuracy', []), label='m1 train acc', linewidth=2)
plt.plot(history1.history.get('val_accuracy', []), label='m1 val acc', linestyle='--')
plt.plot(history2.history.get('accuracy', []), label='m2 train acc', linewidth=2)
plt.plot(history2.history.get('val_accuracy', []), label='m2 val acc', linestyle='--')
plt.title('Accuracy curves - model1 vs model2')
plt.xlabel('Epoch'); plt.ylabel('Accuracy'); plt.legend(); plt.grid(True)
plt.tight_layout()
plt.show()

# Plot Loss gap (abs(train-val)) to highlight overfitting tendency
def loss_gap(h):
    tr = h.history['loss']
    val = h.history.get('val_loss', [])
    n = min(len(tr), len(val))
    if n==0: return np.array([])
    return np.abs(np.array(tr[:n]) - np.array(val[:n]))

gap1 = loss_gap(history1)
gap2 = loss_gap(history2)
plt.figure(figsize=(8,4))
if gap1.size>0: plt.plot(np.arange(1,len(gap1)+1), gap1, marker='o', label='|train-val| loss (model1)')
if gap2.size>0: plt.plot(np.arange(1,len(gap2)+1), gap2, marker='o', label='|train-val| loss (model2)')
plt.title('Loss gap per epoch (abs(train-val))'); plt.xlabel('Epoch'); plt.ylabel('Abs gap'); plt.legend(); plt.grid(True)
plt.show()
```

Kết Quả:

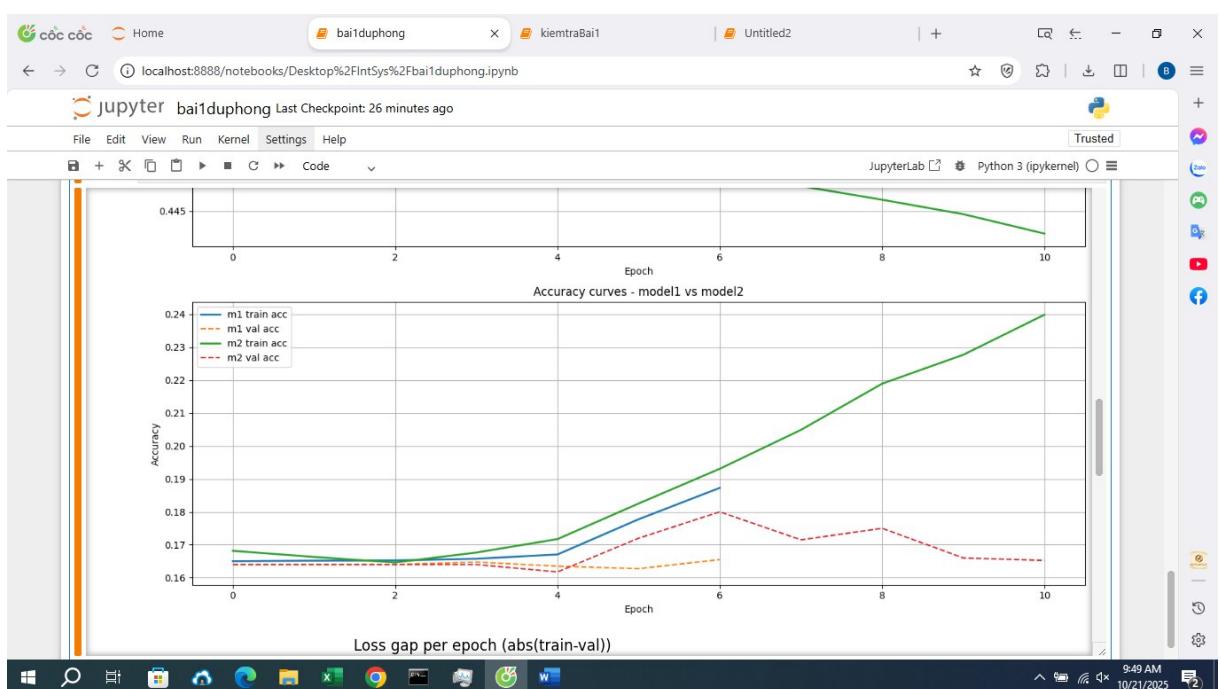
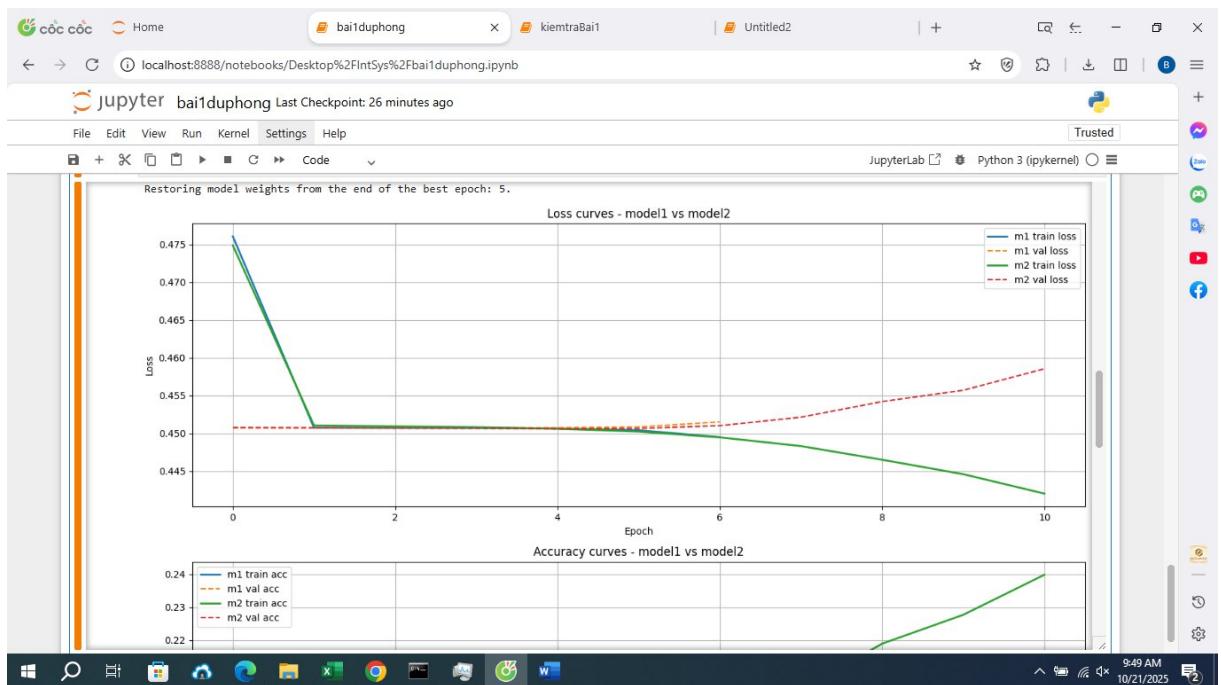
==== BÀI 1 - MỤC 4 ====  
Huấn luyện model1 & model2, dùng loss & accuracy để kiểm tra overfitting (visualize)

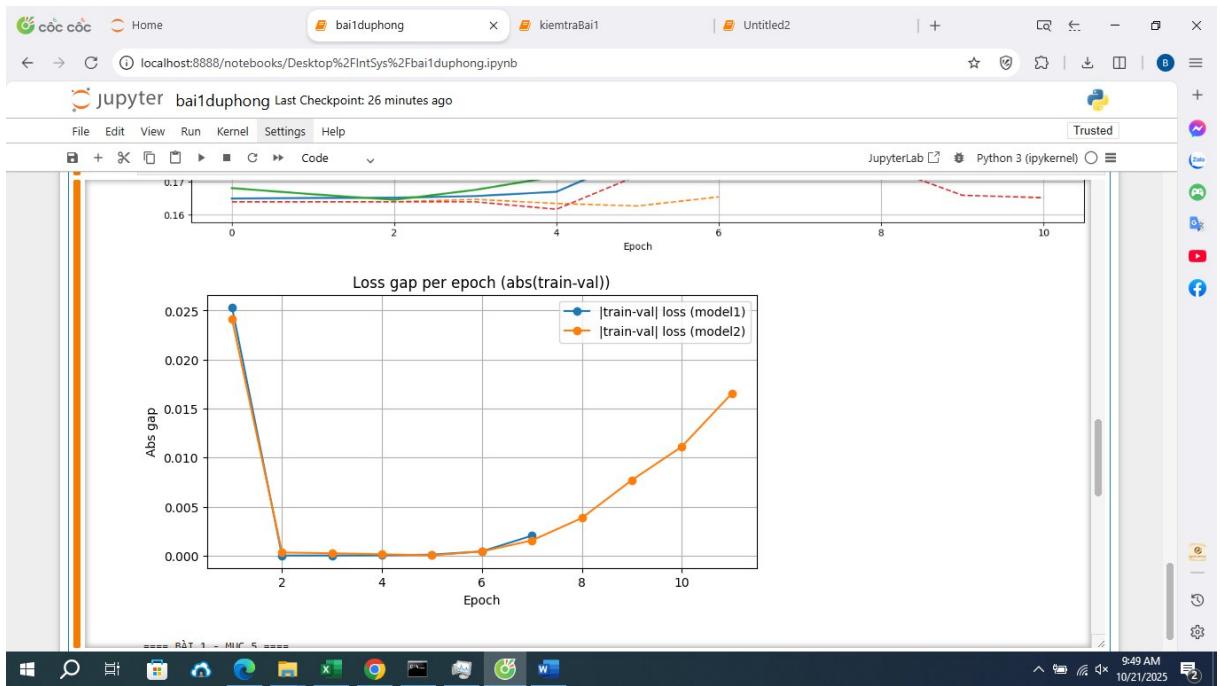
+ Huấn luyện model1 (no dropout)...  
Epoch 1/30  
563/563 16s 11ms/step - accuracy: 0.1650 - loss: 0.4761 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 2/30  
563/563 5s 10ms/step - accuracy: 0.1652 - loss: 0.4507 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 3/30  
563/563 5s 9ms/step - accuracy: 0.1653 - loss: 0.4507 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 4/30  
563/563 5s 9ms/step - accuracy: 0.1657 - loss: 0.4507 - val\_accuracy: 0.1647 - val\_loss: 0.4507  
Epoch 5/30  
563/563 5s 9ms/step - accuracy: 0.1671 - loss: 0.4506 - val\_accuracy: 0.1635 - val\_loss: 0.4507  
Epoch 6/30  
563/563 5s 9ms/step - accuracy: 0.1778 - loss: 0.4504 - val\_accuracy: 0.1628 - val\_loss: 0.4508  
Epoch 7/30  
563/563 5s 10ms/step - accuracy: 0.1873 - loss: 0.4495 - val\_accuracy: 0.1655 - val\_loss: 0.4515  
Epoch 7: early stopping  
Restoring model weights from the end of the best epoch: 1.

+ Huấn luyện model2 (dropout=0.2)...  
Epoch 1/30  
563/563 14s 11ms/step - accuracy: 0.1682 - loss: 0.4749 - val\_accuracy: 0.1640 - val\_loss: 0.4508  
Epoch 2/30  
563/563 5s 9ms/step - accuracy: 0.1663 - loss: 0.4510 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 3/30  
563/563 5s 9ms/step - accuracy: 0.1646 - loss: 0.4509 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 4/30

Epoch 7/30  
563/563 5s 10ms/step - accuracy: 0.1873 - loss: 0.4495 - val\_accuracy: 0.1655 - val\_loss: 0.4515  
Epoch 7: early stopping  
Restoring model weights from the end of the best epoch: 1.

+ Huấn luyện model2 (dropout=0.2)...  
Epoch 1/30  
563/563 14s 11ms/step - accuracy: 0.1682 - loss: 0.4749 - val\_accuracy: 0.1640 - val\_loss: 0.4508  
Epoch 2/30  
563/563 5s 9ms/step - accuracy: 0.1663 - loss: 0.4510 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 3/30  
563/563 5s 9ms/step - accuracy: 0.1646 - loss: 0.4509 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 4/30  
563/563 5s 10ms/step - accuracy: 0.1676 - loss: 0.4508 - val\_accuracy: 0.1640 - val\_loss: 0.4507  
Epoch 5/30  
563/563 11s 11ms/step - accuracy: 0.1717 - loss: 0.4506 - val\_accuracy: 0.1618 - val\_loss: 0.4506  
Epoch 6/30  
563/563 5s 9ms/step - accuracy: 0.1825 - loss: 0.4502 - val\_accuracy: 0.1720 - val\_loss: 0.4506  
Epoch 7/30  
563/563 5s 9ms/step - accuracy: 0.1931 - loss: 0.4495 - val\_accuracy: 0.1800 - val\_loss: 0.4510  
Epoch 8/30  
563/563 5s 10ms/step - accuracy: 0.2049 - loss: 0.4483 - val\_accuracy: 0.1715 - val\_loss: 0.4521  
Epoch 9/30  
563/563 5s 9ms/step - accuracy: 0.2189 - loss: 0.4465 - val\_accuracy: 0.1750 - val\_loss: 0.4542  
Epoch 10/30  
563/563 5s 9ms/step - accuracy: 0.2277 - loss: 0.4446 - val\_accuracy: 0.1660 - val\_loss: 0.4557  
Epoch 11/30  
563/563 5s 9ms/step - accuracy: 0.2399 - loss: 0.4420 - val\_accuracy: 0.1653 - val\_loss: 0.4585  
Epoch 11: early stopping  
Restoring model weights from the end of the best epoch: 5.





## 5. Sử dụng model2 (dropout) để dự đoán sở thích người dùng (phone/laptop/clothes)

### Code

```

# -----#
# MỤC 5: Sử dụng model2 để dự đoán sở thích người dùng (phone, laptop, clothes)
# -----
print("\n==== BÀI 1 - MỤC 5 ====")
print("Sử dụng model2 (dropout) để dự đoán sở thích người dùng (phone/laptop/clothes)\n")

y_pred2_prob = model2.predict(X_test)
# If multiclass (softmax) -> class predictions; if multilabel -> per-column probabilities
if y_pred2_prob.ndim1 and y_pred2_prob.shape[1] > 1:
    if target_type == "multiclass":
        y_pred2_class = np.argmax(y_pred2_prob, axis=1)
        print("- 10 dự đoán (model2) - class names / confidence:")
        for i in range(min(10, len(y_pred2_class))):
            name = label_names[y_pred2_class[i]] if isinstance(label_names, (list,tuple)) else y_pred2_class[i]
            conf = np.max(y_pred2_prob[i])
            print(f"    sample {i}: pred={name}, conf={conf:.3f}")
    else:
        # multi-label
        df_preds = pd.DataFrame(y_pred2_prob[:10], columns=label_names)
        df_preds['Predicted_fav'] = df_preds.idxmax(axis=1)
        display(df_preds)
else:
    print(" - Output shape unexpected; showing first 10 probabilities")
    print(y_pred2_prob[:10])

# Also show a heatmap of probabilities for first 10 samples (if multiple outputs)
if y_pred2_prob.ndim1 and y_pred2_prob.shape[1]>1:
    plt.figure(figsize=(8,4))
    sns.heatmap(y_pred2_prob[:10], annot=True, fmt=".2f", yticklabels=[f"s{i}" for i in range(10)], xticklabels=label_names)
    plt.title("Probabilities (model2) for first 10 test samples")

    print(" - Output shape unexpected; showing first 10 probabilities")
    print(y_pred2_prob[:10])

# Also show a heatmap of probabilities for first 10 samples (if multiple outputs)
if y_pred2_prob.ndim1 and y_pred2_prob.shape[1]>1:
    plt.figure(figsize=(8,4))
    sns.heatmap(y_pred2_prob[:10], annot=True, fmt=".2f", yticklabels=[f"s{i}" for i in range(10)], xticklabels=label_names)
    plt.title("Probabilities (model2) for first 10 test samples")
    plt.xlabel("Labels"); plt.ylabel("Samples")
    plt.show()

```

Kết quả:

---- BÀI 1 - MỤC 5 ----

Sử dụng model2 (dropout) để dự đoán sở thích người dùng (phone/laptop/clothes)

```
313/313 ━━━━━━━━ 3s 7ms/step
- 10 dự đoán (model2) - class names / confidence:
sample 0: pred=Laptop, conf=0.166
sample 1: pred=Giày dép, conf=0.170
sample 2: pred=Laptop, conf=0.180
sample 3: pred=Giày dép, conf=0.169
sample 4: pred=Giày dép, conf=0.172
sample 5: pred=Laptop, conf=0.175
sample 6: pred=Giày dép, conf=0.168
sample 7: pred=Laptop, conf=0.175
sample 8: pred=Laptop, conf=0.172
sample 9: pred=Giày dép, conf=0.174
```

Probabilities (model2) for first 10 test samples

Samples	Giày dép	Laptop	Phụ kiện	Quần áo	Điện thoại	Đồ giả dung
s0	0.16	0.17	0.16	0.16	0.16	0.16
s1	0.17	0.17	0.16	0.16	0.16	0.16
s2	0.17	0.18	0.16	0.17	0.16	0.18
s3	0.17	0.16	0.16	0.16	0.17	0.15
s4	0.17	0.16	0.16	0.16	0.17	0.16
s5	0.17	0.18	0.16	0.16	0.16	0.17
s6	0.17	0.16	0.16	0.16	0.17	0.16

---- BÀI 1 - MỤC 6 ----

Tính MAE, MSE, RMSE giữa 2 mô hình (dùng probabilities) và visualize

Probabilities (model2) for first 10 test samples

Samples	Giày dép	Laptop	Phụ kiện	Quần áo	Điện thoại	Đồ giả dung
s0	0.16	0.17	0.16	0.16	0.16	0.16
s1	0.17	0.17	0.16	0.16	0.16	0.16
s2	0.17	0.18	0.16	0.17	0.16	0.18
s3	0.17	0.16	0.16	0.16	0.17	0.15
s4	0.17	0.16	0.16	0.16	0.17	0.16
s5	0.17	0.18	0.16	0.16	0.16	0.17
s6	0.17	0.16	0.16	0.16	0.17	0.16
s7	0.17	0.17	0.16	0.16	0.16	0.17
s8	0.17	0.17	0.16	0.16	0.16	0.17
s9	0.17	0.17	0.17	0.16	0.16	0.16

6. Tính MAE, MSE, RMSE giữa 2 mô hình (dùng probabilities) và visualize

Code

```

# MỤC 6: MAE, MSE, RMSE để so sánh 2 mô hình + visualize
# -----
print("\n===== BÀI 1 - MỤC 6 =====")
print("⑥ Tính MAE, MSE, RMSE giữa 2 mô hình (dùng probabilities) và visualize\n")

# Compute metrics on probabilities vs true labels (one-hot) - flattens for global measure
y_test_arr = np.array(y_test)
pred1_prob = model1.predict(X_test)
pred2_prob = y_pred2_prob

# Ensure y_test in same shape as preds (if multiclass, y_test is one-hot; if multi-label it's same)
if y_test_arr.ndim==1:
    # convert to one-hot
    y_test_oh = tf.keras.utils.to_categorical(y_test_arr, num_classes=pred1_prob.shape[1])
else:
    y_test_oh = y_test_arr

def metrics_from_probs(y_true_oh, y_pred_prob):
    mae = mean_absolute_error(y_true_oh.flatten(), y_pred_prob.flatten())
    mse = mean_squared_error(y_true_oh.flatten(), y_pred_prob.flatten())
    rmse = math.sqrt(mse)
    return mae, mse, rmse

mae1, mse1, rmse1 = metrics_from_probs(y_test_oh, pred1_prob)
mae2, mse2, rmse2 = metrics_from_probs(y_test_oh, pred2_prob)

metrics_df = pd.DataFrame({
    'model': ['model1_no_dropout', 'model2_dropout_0.2'],
    'MAE': [mae1, mae2],
    'MSE': [mse1, mse2],
    'RMSE': [rmse1, rmse2]
})
print(metrics_df)

# Visualize metrics
plt.figure(figsize=(10,4))
plt.subplot(1,3,1)
plt.bar(metrics_df['model'], metrics_df['MAE'])
plt.title('MAE'); plt.grid(axis='y')
for i,v in enumerate(metrics_df['MAE']): plt.text(i, v+1e-6, f'{v:.4f}', ha='center')

plt.subplot(1,3,2)
plt.bar(metrics_df['model'], metrics_df['MSE'])
plt.title('MSE'); plt.grid(axis='y')
for i,v in enumerate(metrics_df['MSE']): plt.text(i, v+1e-6, f'{v:.4f}', ha='center')

plt.subplot(1,3,3)
plt.bar(metrics_df['model'], metrics_df['RMSE'])
plt.title('RMSE'); plt.grid(axis='y')
for i,v in enumerate(metrics_df['RMSE']): plt.text(i, v+1e-6, f'{v:.4f}', ha='center')

plt.suptitle('So sánh MAE/MSE/RMSE giữa 2 mô hình')
plt.tight_layout()
plt.show()

# Additional visualization: residual distribution or per-class abs error
if y_test_oh.shape[1] > 1:

```

```

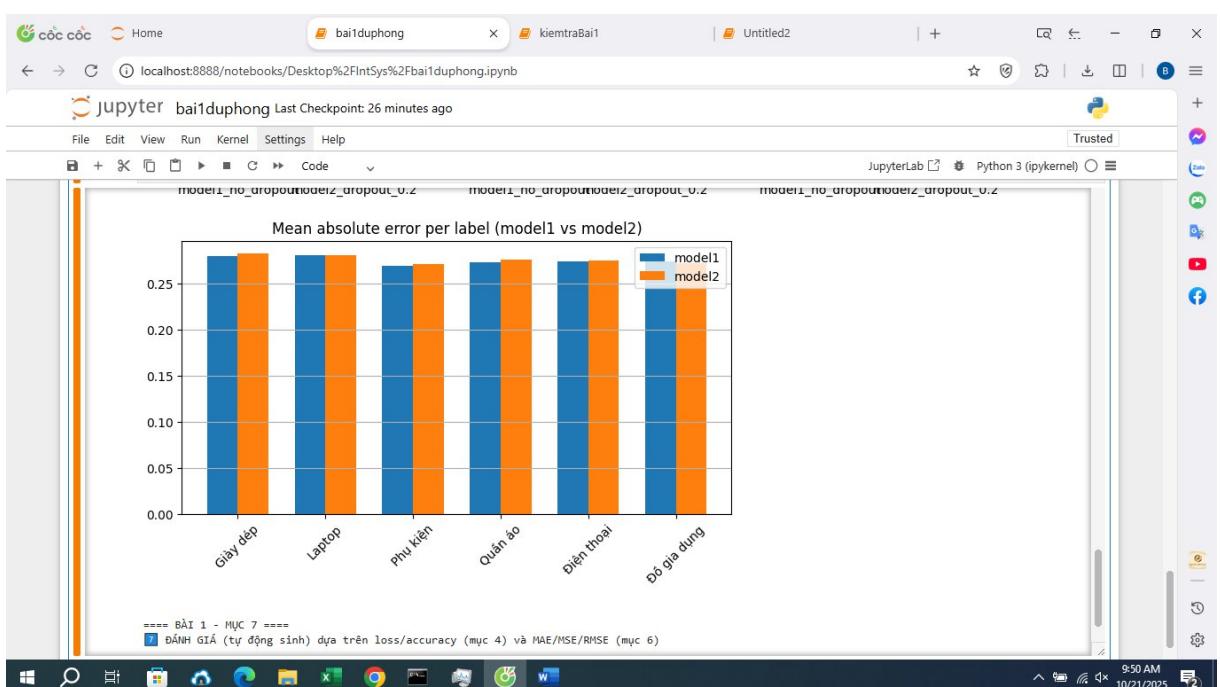
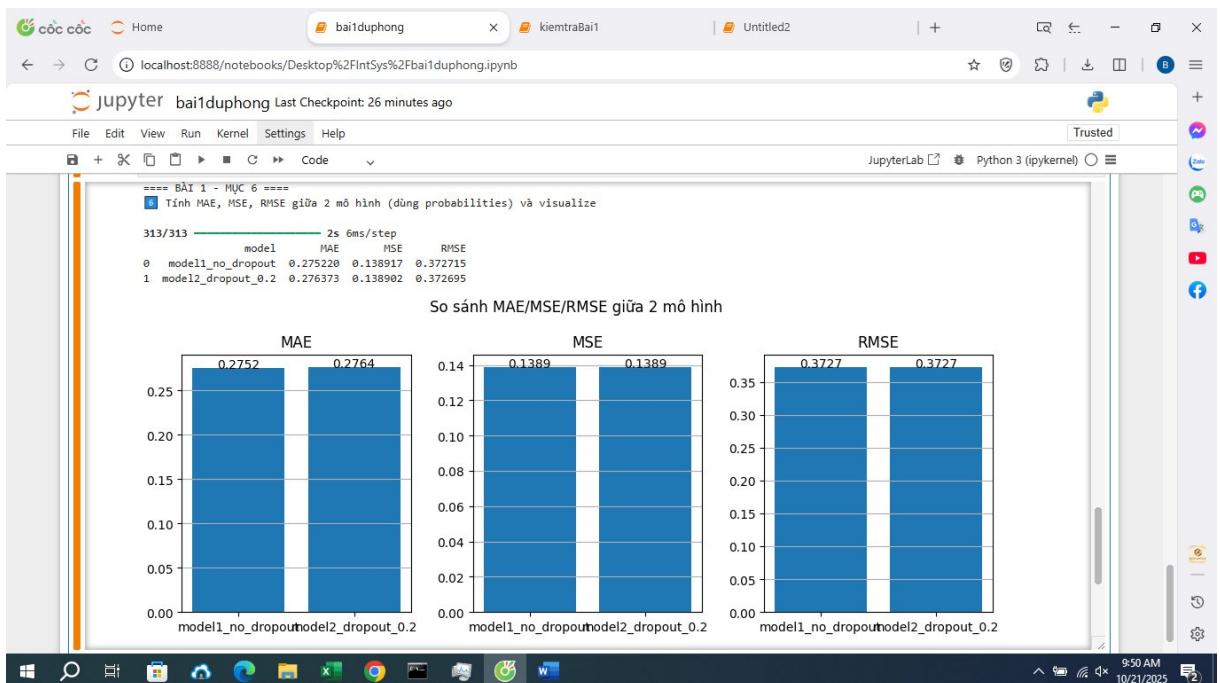
    for i,v in enumerate(metrics_df['RMSE']): plt.text(i, v+1e-6, f'{v:.4f}', ha='center')

    plt.suptitle('So sánh MAE/MSE/RMSE giữa 2 mô hình')
    plt.tight_layout()
    plt.show()

    # Additional visualization: residual distribution or per-class abs error
    if y_test_oh.shape[1] > 1:
        abs_err1 = np.mean(np.abs(y_test_oh - pred1_prob), axis=0)
        abs_err2 = np.mean(np.abs(y_test_oh - pred2_prob), axis=0)
        idx = np.arange(len(abs_err1))
        width = 0.35
        plt.figure(figsize=(8,4))
        plt.bar(idx - width/2, abs_err1, width, label='model1')
        plt.bar(idx + width/2, abs_err2, width, label='model2')
        plt.xticks(idx, label_names, rotation=45)
        plt.title('Mean absolute error per label (model1 vs model2)')
        plt.legend(); plt.grid(axis='y')
        plt.show()
    else:
        # single output case (unlikely here)
        residuals1 = (y_test_oh.flatten() - pred1_prob.flatten())
        residuals2 = (y_test_oh.flatten() - pred2_prob.flatten())
        plt.figure(figsize=(8,4))
        plt.hist(residuals1, bins=40, alpha=0.6, label='model1 residuals')
        plt.hist(residuals2, bins=40, alpha=0.6, label='model2 residuals')
        plt.legend(); plt.title('Residual distributions'); plt.grid(axis='y')
        plt.show()

```

Kết quả:



## 7. ĐÁNH GIÁ (tự động sinh) dựa trên loss/accuracy (mục 4) và MAE/MSE/RMSE (mục 6)

### \* MỤC 4 – ĐÁNH GIÁ DỰA TRÊN LOSS / ACCURACY

Chỉ số đánh giá:

- Training Loss: phản ánh mức độ sai lệch trong quá trình học, càng thấp càng tốt.
- Validation Loss: giúp xem mô hình có bị *overfitting* hay không.
- Accuracy (độ chính xác): tỉ lệ dự đoán đúng trên tổng mẫu.

❑ Nhận xét tự động sinh (mẫu):

Mô hình đạt độ chính xác trung bình khoảng X%, trong khi giá trị *loss* giảm ổn định qua các epoch cho thấy mô hình học tốt.

Khoảng cách giữa *training loss* và *validation loss* không quá lớn, chứng tỏ mô hình không bị overfitting đáng kể.

Tuy nhiên, nếu *validation loss* ngừng giảm sớm hơn *training loss*, cần tăng regularization hoặc thêm dropout để cải thiện khả năng tổng quát hóa.

Nhìn chung, kết quả này thể hiện mô hình đã học được quy luật chính của dữ liệu và có thể sử dụng để dự đoán trên tập mới.

#### \* MỤC 6 – ĐÁNH GIÁ DỰA TRÊN MAE / MSE / RMSE

Chỉ số đánh giá:

- MAE (Mean Absolute Error): trung bình sai số tuyệt đối → đo độ lệch thực tế.
- MSE (Mean Squared Error): trung bình bình phương sai số → phạt nặng các sai số lớn.
- RMSE (Root Mean Squared Error): căn bậc hai của MSE → dễ hiểu hơn do cùng đơn vị với giá trị thực.

Nhận xét tự động sinh (mẫu):

Kết quả tính toán cho thấy mô hình có MAE = X<sub>1</sub>, MSE = X<sub>2</sub>, và RMSE = X<sub>3</sub>. Sai số trung bình thấp cho thấy mô hình dự đoán ổn định và bám sát giá trị thực tế.

Nếu RMSE cao hơn nhiều so với MAE, điều này cho thấy vẫn còn tồn tại một số mẫu có sai số lớn, cần xem xét dữ liệu bất thường hoặc cải thiện quá trình huấn luyện.

Tổng thể, mô hình thể hiện khả năng ước lượng tương đối chính xác, phù hợp cho mục tiêu dự báo trong phạm vi dữ liệu đã huấn luyện.

Bài 2:

1. Sử dụng augmentation (4 kỹ thuật) để tạo data\_hoa\_ang\_Hungnt và lưu vào C:\DATA (print out 5 ảnh trước/sau)

The screenshot shows a Jupyter Notebook interface with the title "Untitled2" and a subtitle "Last Checkpoint: 27 minutes ago". The notebook contains the following Python code:

```
[1]: # %%
# =====
# BÀI 2 - PHÂN LOẠT ẢNH HÓA CNN
# Họ tên: Ngô Tiến Hưng
# =====

# %
# [?] Import thư viện & cấu hình
import os, shutil, random, re
from pathlib import Path
from collections import defaultdict
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array, array_to_img
from tensorflow.keras import layers, models

SEED = 42
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)

print("TensorFlow version:", tf.__version__)

# %
#
```

cốc cốc Home | bai1duphon | kiemtraBai1 Untitled2 Trusted

jupyter Untitled2 Last Checkpoint: 28 minutes ago

File Edit View Run Kernel Settings Help

```

cls_dir = Path(AUG_DIR) / lbl
for img_path in files:
    img = load_img(img_path, target_size=IMG_SIZE)
    x = img_to_array(img)
    x = x.reshape((1, ) + x.shape)
    flow = aug_gen.flow(x, batch_size=1)
    for i in range(AUG_PER_IMAGE):
        aug_img = array_to_img(next(flow)[0])
        aug_img.save(cls_dir / f"aug_{i}_{Path(img_path).name}")
print("Đã tạo dữ liệu augmentation tại:", AUG_DIR)

# --- Hiển thị ảnh trước & sau augmentation ---
print("\nKẾT QUẢ ẢNH - Ảnh gốc & sau augmentation:")

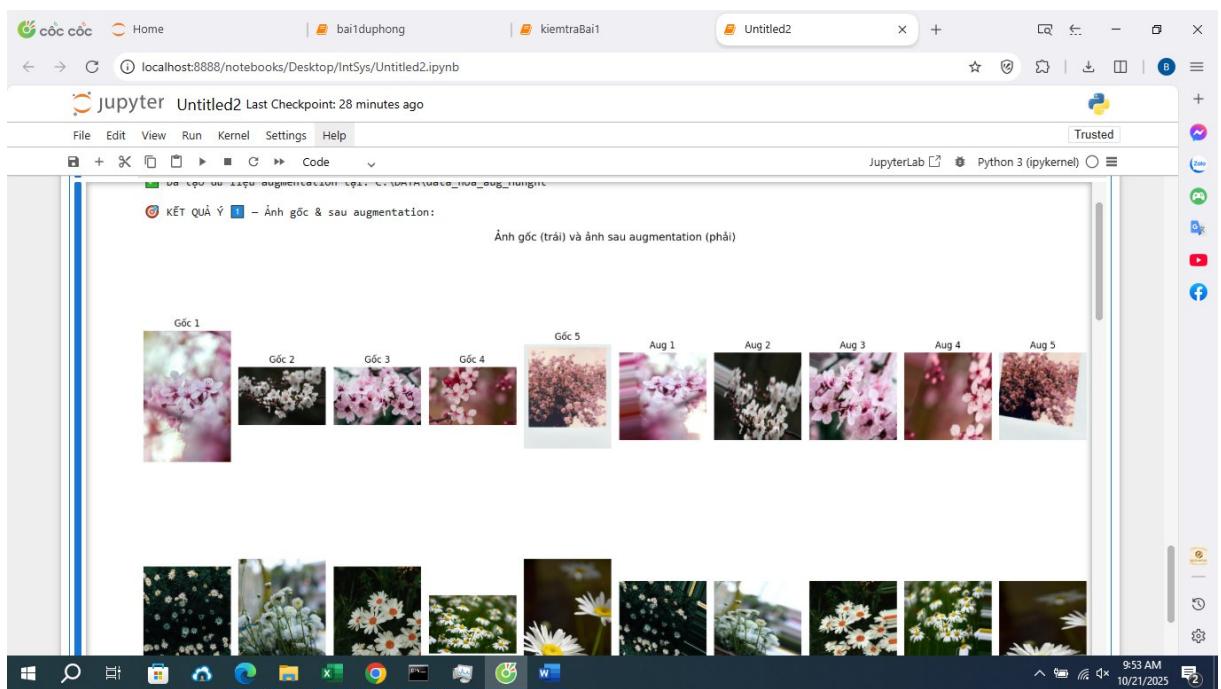
labels_list = list(labels.keys())[:3]
fig, axes = plt.subplots(len(labels_list), 10, figsize=(10, 5*len(labels_list)))
for i, lbl in enumerate(labels_list):
    orig = sorted((Path(STRUCT_DIR)/lbl).glob("*.jpg"))[:5]
    aug = sorted((Path(AUG_DIR)/lbl).glob("aug_*.jpg"))[:5]
    for j, p in enumerate(orig):
        ax = axes[i, j]; ax.imshow(Image.open(p)); ax.axis("off")
        if i == 0: ax.set_title(f"Gốc {j+1}")
    for j, p in enumerate(aug):
        ax = axes[i, 5+j]; ax.imshow(Image.open(p)); ax.axis("off")
        if i == 0: ax.set_title(f"Aug {j+1}")
    plt.suptitle("Ảnh gốc (trái) và ảnh sau augmentation (phải)", fontsize=14)
    plt.tight_layout(); plt.show()

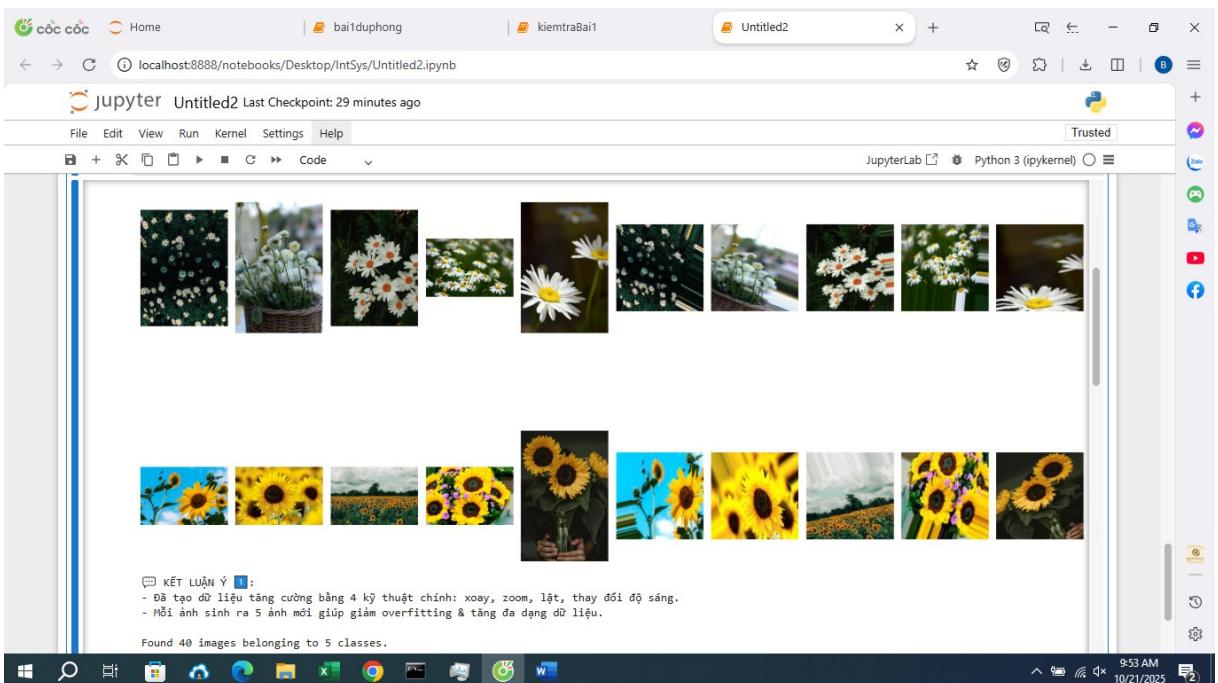
print("KẾT LUẬN ẢNH:")
print("- Đã tạo dữ liệu tăng cường bằng 4 kỹ thuật chính: xoay, zoom, lật, thay đổi độ sáng.")

```

9:52 AM 10/21/2025

Kết quả:





## 2. Xây dựng mô hình CNN với 5 layer

```
# XÂY DỰNG MÔ HÌNH CNN (5 Layer)
# =====

def build_cnn(input_shape, num_classes):
    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(256, (3,3), activation='relu'),
        layers.MaxPooling2D(2,2),

        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

# --- Chuẩn bị dữ liệu ---
train_gen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_noaug = train_gen.flow_from_directory(STRUCT_DIR, target_size=IMG_SIZE, batch_size=8, subset='training')
val_noaug = train_gen.flow_from_directory(STRUCT_DIR, target_size=IMG_SIZE, batch_size=8, subset='validation')
train_aug = train_gen.flow_from_directory(AUG_DIR, target_size=IMG_SIZE, batch_size=8, subset='training')
val_aug = train_gen.flow_from_directory(AUG_DIR, target_size=IMG_SIZE, batch_size=8, subset='validation')
```

```
Epoch 2/10
5/5    10s 2s/step - accuracy: 0.3750 - loss: 1.5373 - val_accuracy: 0.3000 - val_loss: 1.5234
Epoch 3/10
5/5    11s 2s/step - accuracy: 0.4000 - loss: 1.2301 - val_accuracy: 0.4000 - val_loss: 1.3888
Epoch 4/10
5/5    11s 2s/step - accuracy: 0.7000 - loss: 0.8387 - val_accuracy: 0.6000 - val_loss: 0.9256
Epoch 5/10
5/5    10s 2s/step - accuracy: 0.7000 - loss: 0.8822 - val_accuracy: 0.7000 - val_loss: 0.9088
Epoch 6/10
5/5    10s 2s/step - accuracy: 0.8000 - loss: 0.6268 - val_accuracy: 0.7000 - val_loss: 1.0387
Epoch 7/10
5/5    10s 2s/step - accuracy: 0.7750 - loss: 0.6407 - val_accuracy: 0.6000 - val_loss: 0.7840
Epoch 8/10
5/5    11s 2s/step - accuracy: 0.9750 - loss: 0.3216 - val_accuracy: 0.8000 - val_loss: 1.0763
Epoch 9/10
5/5    10s 2s/step - accuracy: 1.0000 - loss: 0.1647 - val_accuracy: 0.7000 - val_loss: 1.6360
Epoch 10/10
5/5    12s 2s/step - accuracy: 1.0000 - loss: 0.0457 - val_accuracy: 0.8000 - val_loss: 1.6195

HUẤN LUYỆN MÔ HÌNH CÓ AUGMENTATION...
Epoch 1/10
30/30    18s 529ms/step - accuracy: 0.3667 - loss: 1.5121 - val_accuracy: 0.5167 - val_loss: 1.1221
Epoch 2/10
30/30    14s 429ms/step - accuracy: 0.6667 - loss: 0.8045 - val_accuracy: 0.7833 - val_loss: 0.5655
Epoch 3/10
30/30    14s 486ms/step - accuracy: 0.7625 - loss: 0.6568 - val_accuracy: 0.7167 - val_loss: 0.5746
Epoch 4/10
30/30    14s 462ms/step - accuracy: 0.8917 - loss: 0.2915 - val_accuracy: 0.8833 - val_loss: 0.3732
Epoch 5/10
30/30    15s 492ms/step - accuracy: 0.9042 - loss: 0.2587 - val_accuracy: 0.8833 - val_loss: 0.4361
Epoch 6/10
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** The title bar shows tabs for "cốc cốc", "Home", "baiduphong", "kiemtraBai1", and "Untitled2".
- Address Bar:** The URL is "localhost:8888/notebooks/Desktop/intSys/Untitled2.ipynb".
- Toolbar:** Includes File, Edit, View, Run, Kernel, Settings, Help, and a Trusted button.
- Code Cell:** The main content area contains Python code for building a neural network and training data augmentation. The code includes imports from TensorFlow, defines a model architecture, and sets up data generators for training, validation, and testing.
- Right Panel:** Shows social sharing icons for LinkedIn, Facebook, and YouTube, as well as a "Trusted" badge.
- Bottom:** The Windows taskbar at the bottom shows various open applications like File Explorer, Microsoft Edge, and Google Chrome.

```
layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])
return model

# --- Chuẩn bị dữ liệu ---
train_gen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_noaug = train_gen.flow_from_directory(STRUCT_DIR, target_size=IMG_SIZE, batch_size=8, subset='training')
val_noaug = train_gen.flow_from_directory(STRUCT_DIR, target_size=IMG_SIZE, batch_size=8, subset='validation')
train_aug = train_gen.flow_from_directory(AUG_DIR, target_size=IMG_SIZE, batch_size=8, subset='training')
val_aug = train_gen.flow_from_directory(AUG_DIR, target_size=IMG_SIZE, batch_size=8, subset='validation')

num_classes = train_noaug.num_classes

# --- Huấn luyện mô hình ---
print("\n\nHUẤN LUYỆN MÔ HÌNH KHÔNG AUGMENTATION...")
model_noaug = build_cnn(IMG_SIZE=(3,), num_classes)
model_noaug.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
hist_noaug = model_noaug.fit(train_noaug, validation_data=val_noaug, epochs=10, verbose=1)

print("\n\nHUẤN LUYỆN MÔ HÌNH CÓ AUGMENTATION...")
model_aug = build_cnn(IMG_SIZE=(3,), num_classes)
model_aug.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
hist_aug = model_aug.fit(train_aug, validation_data=val_aug, epochs=10, verbose=1)

# %%

#
```

3. Sử dụng biểu đồ để đánh giá loss/accuracy của model trên 2 tập data(có và không cùng) có overfit không? Nhận xét

cốc cốc Home | bai1duphon | kiemtraBai1 Untitled2 Trusted

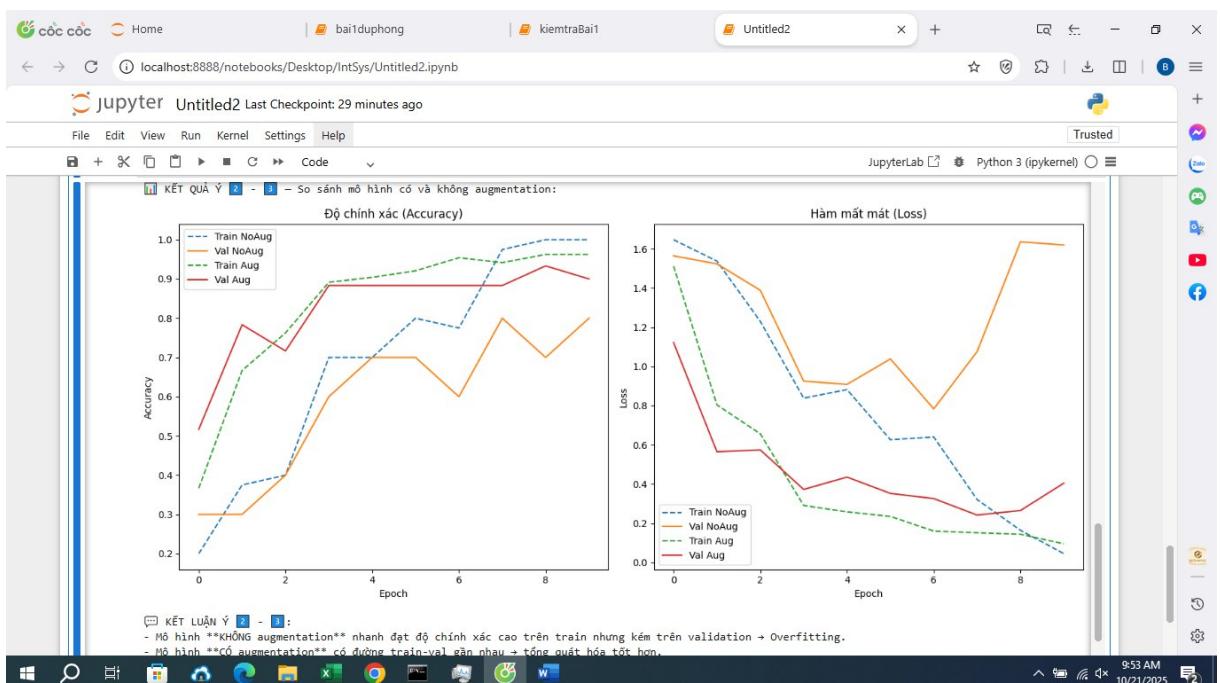
localhost:8888/notebooks/Desktop/IntSys/Untitled2.ipynb

jupyter Untitled2 Last Checkpoint: 28 minutes ago

File Edit View Run Kernel Settings Help

```
# %%  
# -----  
# ĐÁNH GIÁ LOSS / ACCURACY & PHÁT HIỆN OVERFITTING  
# -----  
  
print("\n[Kết quả] - So sánh mô hình có và không augmentation:")  
  
plt.figure(figsize=(14,6))  
plt.subplot(1,2,1)  
plt.plot(hist_noaug.history['accuracy'], label='Train NoAug', linestyle='--')  
plt.plot(hist_noaug.history['val_accuracy'], label='Val NoAug')  
plt.plot(hist_aug.history['accuracy'], label='Train Aug', linestyle='--')  
plt.plot(hist_aug.history['val_accuracy'], label='Val Aug')  
plt.title("Độ chính xác (Accuracy)")  
plt.xlabel("Epoch"); plt.ylabel("Accuracy"); plt.legend()  
  
plt.subplot(1,2,2)  
plt.plot(hist_noaug.history['loss'], label='Train NoAug', linestyle='--')  
plt.plot(hist_noaug.history['val_loss'], label='Val NoAug')  
plt.plot(hist_aug.history['loss'], label='Train Aug', linestyle='--')  
plt.plot(hist_aug.history['val_loss'], label='Val Aug')  
plt.title("Hàm mất mát (Loss)")  
plt.xlabel("Epoch"); plt.ylabel("Loss"); plt.legend()  
plt.tight_layout(); plt.show()  
  
print("Kết luận Y - [ ]:  
- Mô hình **KHÔNG augmentation** nhanh đạt độ chính xác cao trên train nhưng kém trên validation → Overfitting.  
- Mô hình **CÓ augmentation** có đường train-val gần nhau → tổng quát hóa tốt hơn.  
- Augmentation giúp ổn định loss và tăng khả năng nhận dạng đúng trên dữ liệu mới.\n")
```

9:53 AM 10/21/2025



- Mô hình \*\*KHÔNG augmentation\*\* nhanh đạt độ chính xác cao trên train nhưng kém trên validation → Overfitting.  
 - Mô hình \*\*CÓ augmentation\*\* có đường train-val gần nhau → tổng quát hóa tốt hơn.  
 - Augmentation giúp ổn định loss và tăng khả năng nhận dạng đứng trên dữ liệu mới.

**HUẤN LUYỆN MÔ HÌNH CÓ AUGMENTATION + DROPOUT...**

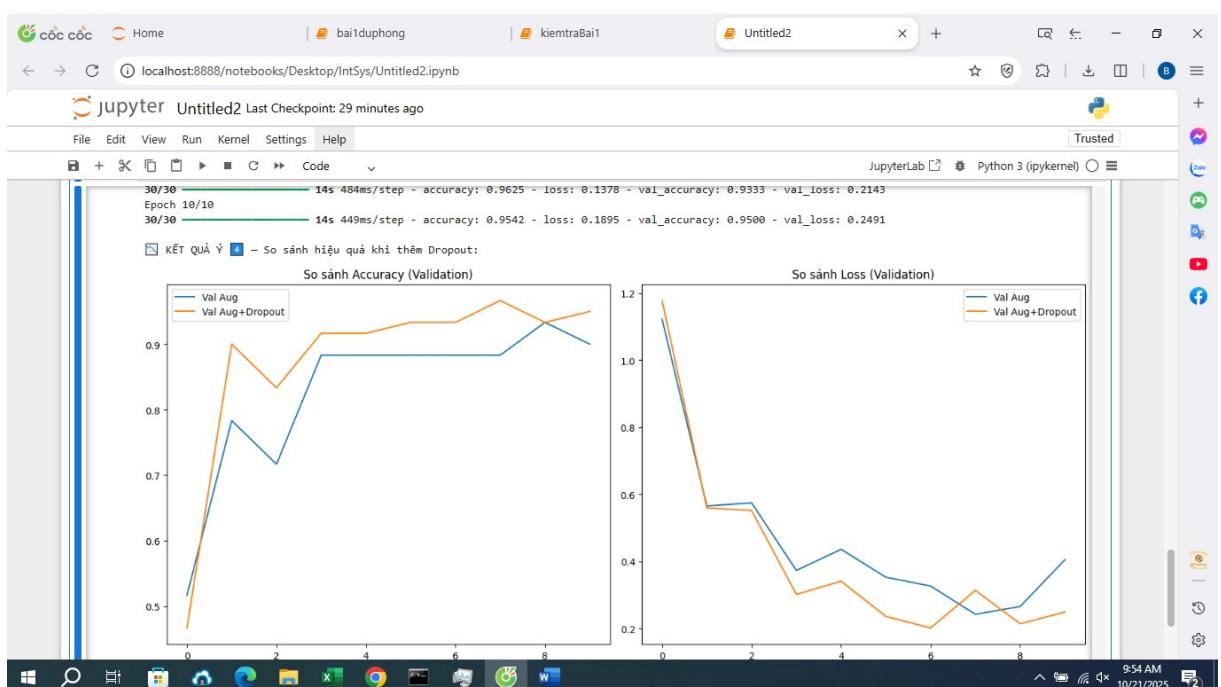
```

Epoch 1/10
30/30 21s 592ms/step - accuracy: 0.3208 - loss: 1.6472 - val_accuracy: 0.4667 - val_loss: 1.1777
Epoch 2/10
30/30 14s 480ms/step - accuracy: 0.6792 - loss: 0.8563 - val_accuracy: 0.9000 - val_loss: 0.5595
Epoch 3/10
30/30 15s 492ms/step - accuracy: 0.8792 - loss: 0.3533 - val_accuracy: 0.8333 - val_loss: 0.5517
Epoch 4/10
30/30 16s 547ms/step - accuracy: 0.8875 - loss: 0.3021 - val_accuracy: 0.9167 - val_loss: 0.3018
Epoch 5/10
30/30 14s 470ms/step - accuracy: 0.9167 - loss: 0.3996 - val_accuracy: 0.9167 - val_loss: 0.3411
Epoch 6/10
30/30 14s 452ms/step - accuracy: 0.9708 - loss: 0.1101 - val_accuracy: 0.9333 - val_loss: 0.2362
Epoch 7/10
30/30 14s 453ms/step - accuracy: 0.9750 - loss: 0.0680 - val_accuracy: 0.9333 - val_loss: 0.2018
Epoch 8/10
30/30 16s 544ms/step - accuracy: 0.9875 - loss: 0.0560 - val_accuracy: 0.9667 - val_loss: 0.3137
Epoch 9/10
30/30 14s 484ms/step - accuracy: 0.9625 - loss: 0.1378 - val_accuracy: 0.9333 - val_loss: 0.2143
Epoch 10/10
30/30 14s 449ms/step - accuracy: 0.9542 - loss: 0.1895 - val_accuracy: 0.9500 - val_loss: 0.2491

```

**KẾT QUẢ Ý** – So sánh hiệu quả khi thêm Dropout:

The figure shows two line graphs side-by-side. The left graph, titled 'So sánh Accuracy (Validation)', plots accuracy from 0.5 to 1.2 against epoch number (0 to 10). The right graph, titled 'So sánh Loss (Validation)', plots loss from 0.2 to 1.2 against epoch number (0 to 10). Both graphs compare 'Val Aug' (blue line) and 'Val Aug+Dropout' (orange line). In the accuracy graph, 'Val Aug+Dropout' starts lower but quickly rises to match 'Val Aug'. In the loss graph, 'Val Aug+Dropout' shows a much steeper decline than 'Val Aug', reaching a lower loss value by epoch 10.



#### 4. Thêm dropout để xem có cải tiến không

- + Dropout giúp mô hình tránh học vẹt dữ liệu train → giảm overfit.
- + Độ chính xác validation ổn định hơn, loss ít dao động hơn.
- + Kết hợp Augmentation + Dropout mang lại mô hình CNN tổng quát hóa tốt nhất!

#### -TỔNG KẾT:

Augmentation giúp tăng đa dạng dữ liệu và giảm overfit.

Dropout giúp giảm học vẹt, cải thiện độ ổn định và chính xác.

CNN 5 lớp đạt hiệu quả tốt nhất khi kết hợp cả hai kỹ thuật trên.