Обучение своей модели для сравнения изображений — это многоэтапный процесс, но я разберу его подробно и с примерами кода. Мы рассмотрим два подхода:   
1. Использование предобученной модели (Transfer Learning).   
2. Обучение Siamese Network (для точного сравнения).   
---  
# 1. Подготовка данных    
### Что нужно:    
- Папка с изображениями экспонатов (например, museum\_dataset/).   
- Разметка (если модель классифицирует экспонаты, а не сравнивает).   
  
### Структура датасета:

museum\_dataset/

├── vase\_1/ # Класс 1 (ваза)

│ ├── img1.jpg

│ └── img2.jpg

├── painting\_2/ # Класс 2 (картина)

│ ├── img1.jpg

└── ...

Совет: Для сравнения изображений (не классификации) достаточно пар:   
- Положительные пары: Два фото одного экспоната.   
- Отрицательные пары: Фото разных экспонатов.   
---  
# 2. Обучение модели    
## Вариант 1: Transfer Learning (MobileNetV2)    
Используем предобученную модель для извлечения признаков.   
  
### Шаги:    
1. Установите TensorFlow и Keras:

Bash

pip install tensorflow numpy matplotlib

2. Извлеките фичи из изображений:

Python

import tensorflow as tf

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

import numpy as np

*# Загружаем предобученную модель без верхних слоёв*

base\_model = MobileNetV2(weights='imagenet', include\_top=False, pooling='avg')

def extract\_features(img\_path):

img = image.load\_img(img\_path, target\_size=(224, 224))

x = image.img\_to\_array(img)

x = preprocess\_input(x)

x = np.expand\_dims(x, axis=0)

features = base\_model.predict(x)

return features.flatten()

*# Пример для одного изображения*

features = extract\_features('museum\_dataset/vase\_1/img1.jpg')

print(features.shape) *# (1280,) — вектор признак*  
3. Сравнивайте векторы через косинусное расстояние:

Python

from sklearn.metrics.pairwise import cosine\_similarity

def compare\_images(img\_path1, img\_path2):

features1 = extract\_features(img\_path1).reshape(1, -1)

features2 = extract\_features(img\_path2).reshape(1, -1)

similarity = cosine\_similarity(features1, features2)[0][0]

return similarity > 0.8 *# Порог можно настроить*

---  
## Вариант 2: Siamese Network (для точного сравнения)    
Siamese Network обучается на парах изображений и учится определять, одинаковые ли объекты на них.   
  
### Код обучения:

Python

import tensorflow as tf

from tensorflow.keras.layers import Input, Lambda, Dense

from tensorflow.keras.models import Model

*# Создаём модель-близнец*

def create\_siamese\_model(input\_shape):

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=input\_shape)

x = tf.keras.layers.GlobalAveragePooling2D()(base\_model.output)

x = Dense(128, activation='relu')(x)

return Model(inputs=base\_model.input, outputs=x)

input\_shape = (224, 224, 3)

siamese\_network = create\_siamese\_model(input\_shape)

*# Входы для двух изображений*

input\_a = Input(shape=input\_shape)

input\_b = Input(shape=input\_shape)

*# Выходы Siamese Network*

output\_a = siamese\_network(input\_a)

output\_b = siamese\_network(input\_b)

*# Сравнение через L1-расстояние*

distance = Lambda(lambda x: tf.abs(x[0] - x[1]))([output\_a, output\_b])

prediction = Dense(1, activation='sigmoid')(distance)

model = Model(inputs=[input\_a, input\_b], outputs=prediction)

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

*# Генератор пар изображений*

def generate\_pairs(dataset\_dir, batch\_size=32):

*# Здесь должен быть код, который создаёт пары (1 — одинаковые, 0 — разные)*

pass

*# Обучение*

model.fit(generate\_pairs('museum\_dataset'), epochs=10)

---  
# 3. Конвертация в TFLite    
Чтобы использовать модель в Flutter, конвертируем её в TensorFlow Lite:

Python

converter = tf.lite.TFLiteConverter.from\_keras\_model(model)

tflite\_model = converter.convert()

with open('museum\_model.tflite', 'wb') as f:

f.write(tflite\_model)

---17:15

# 4. Интеграция с Flutter    
1. Добавьте `.tflite`-файл в `assets/models`.   
2. Подключите `tflite\_flutter`:   
   YAML

dependencies:

tflite\_flutter: ^0.9.0

3. Загрузите модель и используйте:

dart

Future<List<double>> extractFeatures(String imagePath) async {

final interpreter = await Interpreter.fromAsset('assets/models/museum\_model.tflite');

final inputImage = \_preprocessImage(imagePath); *// Ресайз + нормализация*

final output = List.filled(128, 0.0).reshape([1, 128]);

interpreter.run(inputImage, output);

return output[0];

}

---  
# 5. Оптимизация под музей    
- Аугментация данных: Поворачивайте, изменяйте яркость фото, чтобы модель была устойчивее.   
- Кэширование векторов: Храните признаки экспонатов локально, чтобы не вычислять их каждый раз.   
---  
# Итог    
- Для быстрого старта: Используйте MobileNetV2 + косинусное расстояние.   
- Для максимальной точности: Обучите Siamese Network.   
- Для оффлайн-работы: Конвертируйте модель в .tflite и встройте в приложение.