```
import os
from PIL import Image
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

import torch
from torch.utils.data import Dataset, DataLoader
import torch.nn as nn
import torch.optim as optim

from torchvision import transforms
from torchvision.models import resnet50  # or vgg

import matplotlib.pyplot as plt


data_path = '/content/drive/MyDrive/FaceRecognition/Dataset'


from google.colab import drive
drive.mount('/content/drive')
os.listdir(data_path)
```

```
Mounted at /content/drive
['gender_classification.csv',
 'gender_classification.xlsx',
 'class_identity.txt',
 'list_attribute.txt',
 'Images',
 'model_saved']
```

```
if os.path.exists(data_path):
    print(f"Directory found: {data_path}")
    os.listdir(data_path)
else:
    print(f"Directory not found: {data_path}")
```

```
Directory found: /content/drive/MyDrive/FaceRecognition/Dataset
```

```
!find "/content/drive/MyDrive/FaceRecognition" -type d -name Images
```

```
/content/drive/MyDrive/FaceRecognition/Dataset/Images
```

```
import os
import pandas as pd
from PIL import Image
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms

# Path ke dataset utama
data_path = '/content/drive/MyDrive/FaceRecognition/Dataset'
image_folder = os.path.join(data_path, 'Images')

# Load CSV dan TXT
gender_csv = os.path.join(data_path, 'gender_classification.csv')
identity_txt = os.path.join(data_path, 'class_identity.txt')

identity_df = pd.read_csv(identity_txt, sep=' ', header=None, names=['image_filename', 'identity'])
gender_df = pd.read_csv(gender_csv)
gender_df['gender'] = gender_df['Male'].map({1: 1, 0: 0})

# Validasi file gambar yang memang ada
valid_files = set(os.listdir(image_folder))
identity_df = identity_df[identity_df['image_filename'].isin(valid_files)].reset_index(drop=True)

# format filename dibuat konsisten soalnya file class_identity.txt sama image_filename
# dlm folder gak sama, kemungkinan tanpa padding
min_len = min(len(identity_df), len(gender_df))
identity_df['image_filename'] = identity_df['image_filename'].apply(lambda x: f"{int(x.split('.')[0]):06}.jpg")
gender_df = gender_df.iloc[:min_len].reset_index(drop=True)

combined_df = identity_df.copy()
combined_df['gender'] = gender_df['gender']

## --- Apply robust filtering *before* the split ---
#drop baris yg gendernya NaN
combined_df = combined_df.dropna(subset=['gender'])
```

```
##combined_df = pd.merge(identity_df, gender_df[['image_filename', 'gender']], on='image_filename', how='inner')
## kalo pake coding ini tanpa bawahnya akan error karena PyTorch CrossEntropyLoss() butuh angka, bukan string.

#supaya gendernya dalam bentuk angka (0 atau 1)
combined_df = combined_df[combined_df['gender'].isin([0, 1])].reset_index(drop=True)

# Convert ke int
combined_df['gender'] = combined_df['gender'].astype(int)
## --- End of robust filtering ---

print("Cleaned Combined DF:", combined_df.shape)

# Cek hasil
print(identity_df.head())
print(identity_df.shape)
print(combined_df)
## note: buat liat semua combined_df pake print(combined_df)
## buat filter spesifik file pake combined_df.loc[combined_df['image_filename'].isin([...])]
## buat filter dengan condition (misal gender=male) pake combined_df.loc[combined_df['gender'] == 'Male']
```

```
Cleaned Combined DF: (5000, 3)
  image_filename  identity
0     000051.jpg      1446
1     000052.jpg      3896
2     000065.jpg      3046
3     000166.jpg      4328
4     000198.jpg       693
(5000, 2)
     image_filename  identity  gender
0        000051.jpg      1446       1
1        000052.jpg      3896       1
2        000065.jpg      3046       1
3        000166.jpg      4328       0
4        000198.jpg       693       0
...             ...       ...     ...
4995     202320.jpg      9696       0
4996     202340.jpg      8450       1
4997     202347.jpg      7306       1
4998     202357.jpg      9149       0
4999     202566.jpg      7736       0

[5000 rows x 3 columns]
```

```
# buat cek data adalah label biner (1 untuk Male, 0 untuk Female), dan emang cuma ada 1 kolom 'Male'.
# trus ditambahin
# gender_df['image_filename'] = gender_df.index.map(lambda x: f"{x+1:06}.jpg")
# gender_df['gender'] = gender_df['Male'].map({1: 'Male', 0: 'Female'})
# supaya bisa merge dengan identity_df. --> buat sambungin label ke filename yang benar
# buat balikin ke asal bisa pake
# original_gender_df = pd.read_csv(gender_csv)
# print(original_gender_df.head())

print(gender_df.columns)
print(gender_df.head())
```

```
Index(['Male', 'gender'], dtype='object')
   Male  gender
0     1       1
1     1       1
2     1       1
3     0       0
4     0       0
```

```
# split the data into train and test sets with a 80:20 ratio
from sklearn.model_selection import train_test_split
train_df, val_df = train_test_split(combined_df, test_size=0.2, random_state=42)



from torchvision import transforms

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
    ]),
    'val': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
    ])
}
```

```
# Define the custom dataset class
class GenderDataset(Dataset):
    def __init__(self, dataframe, image_folder_path, transform=None):
        self.dataframe = dataframe
        self.image_folder_path = image_folder_path
        self.transform = transform

    def __len__(self):
        return len(self.dataframe)

    def __getitem__(self, idx):
        img_name = os.path.join(self.image_folder_path, self.dataframe.iloc[idx, 0])
        image = Image.open(img_name).convert('RGB')
        label = int(self.dataframe.iloc[idx, 2]) # Assuming 'gender' is the third column (index 2)

        if self.transform:
            image = self.transform(image)

        return image, label

# Use the defined GenderDataset class instead of the undefined FaceDataset
train_dataset = GenderDataset(train_df, image_folder, transform=data_transforms['train'])
val_dataset = GenderDataset(val_df, image_folder, transform=data_transforms['val'])

train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)

dataloaders = {'train': train_loader, 'val': val_loader}
dataset_sizes = {'train': len(train_dataset), 'val': len(val_dataset)}



# buat ganti semua gambar jadi format yg diproses model pytorch
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),           # ubah pil.image (RGB) jadi tensor dengan shape [3,224,224] inget BGR!!!
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
image_folder_path = os.path.join(data_path, 'Images')

# Buat objek Dataset
if not train_df.empty:
    train_dataset = GenderDataset(train_df, image_folder_path=image_folder_path, transform=transform)
else:
    print("Warning: train_df is empty after splitting and filtering. train_dataset will be empty.")
    train_dataset = GenderDataset(pd.DataFrame(columns=train_df.columns), image_folder_path=image_folder_path, transform=transform) # Cr

if not val_df.empty:
    val_dataset = GenderDataset(val_df, image_folder_path=image_folder_path, transform=transform)
else:
     print("Warning: val_df is empty after splitting and filtering. val_dataset will be empty.")
     val_dataset = GenderDataset(pd.DataFrame(columns=val_df.columns), image_folder_path=image_folder_path, transform=transform) # Creat


## note: pakai os.path.join(data_path, "images") bukan data_path+ '/Images' karena lbh robust & lintas platform

# Buat DataLoader
if len(train_dataset) > 0:
    train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=2)
else:
    print("Warning: train_dataset is empty. train_loader will not be created.")
    train_loader = None # Or handle this case appropriately

if len(val_dataset) > 0:
    val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False, num_workers=2)
else:
    print("Warning: val_dataset is empty. val_loader will not be created.")
    val_loader = None # Or handle this case appropriately

## note: numworkers buat mengaktifkan multiprocessing saat loading data, buat loading data lebih cepet, terutama kalo transform file IO
## Boleh disesuaikan: num_workers=0 (debug mode), atau num_workers=os.cpu_count() untuk maksimal

# Loader dictionary
dataloaders = {
    'train': train_loader,
    'val': val_loader
}
# Dataset size dictionary
```

```
dataset_sizes = {
    'train': len(train_dataset),
    'val': len(val_dataset)
}
```

```
transform = transforms.Compose([
    transforms.RandomHorizontalFlip(),  #Melakukan flipping gambar secara horizontal secara acak saat training.
    transforms.Resize(224),
    transforms.ToTensor(), #Mengubah gambar dari PIL.Image ke format tensor PyTorch.
    transforms.Normalize(mean=[0.485, 0.456, 0.406],  # pakai mean & std dari ImageNet
                         std=[0.229, 0.224, 0.225])
])
```

```
import torch

use_gpu = torch.cuda.is_available()
print(f"Using GPU: {use_gpu}")
```

```
Using GPU: True
```

```
print(dataset_sizes)
```

```
{'train': 4000, 'val': 1000}
```

```
# DEFINE MODEL OPTMIZER DAN CRITERION (LOSS FUNCTION)
import torchvision.models as models
import torch.nn as nn
import torch.optim as optim
import torch

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model_ft = models.resnet50(pretrained=True)# load pretrained resnet
num_ftrs = model_ft.fc.in_features
model_ft.fc = nn.Linear(num_ftrs, 2)
model_ft = model_ft.to(device)

#model_ft = model_ft.float()
criterion = nn.CrossEntropyLoss()           # defining loss function

optimizer_ft = optim.Adam(model_ft.parameters(), lr=0.0001)
```

```
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained' is deprecated sinc
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a weight enum or `None`
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.cache/torch/hub/checkpoints/resnet50-0676ba61.pth
100%|██████████| 97.8M/97.8M [00:00<00:00, 188MB/s]
```

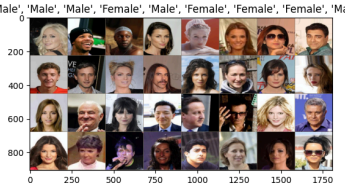
```
import torchvision.utils
import numpy as np
import matplotlib.pyplot as plt
```

```
def imshow(inp, title=None):
    """Imshow for Tensor."""
    inp = inp.numpy().transpose((1, 2, 0))
    mean = np.array([0.485, 0.456, 0.406])
    std = np.array([0.229, 0.224, 0.225])
    inp = std * inp + mean
    inp = np.clip(inp, 0, 1)
    plt.imshow(inp)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)  # pause a bit so that plots are updated
```

```
# Get a batch of training data
# Corrected the variable name from 'dataloders' to 'dataloaders'
inputs, classes = next(iter(dataloaders['train']))

# Make a grid from batch
out = torchvision.utils.make_grid(inputs)

# Define class names
```

```
class_names = ['Female', 'Male'] # Assuming 0 corresponds to Female and 1 to Male
imshow(out, title=[class_names[x] for x in classes])
```

['Male', 'Male', 'Female', 'Male', 'Female', 'Male', 'Female', 'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Male', 'Female', 'Male', 'Female', 'Female', 'Female', 'Male', 'Male', 'Male', 'Female', 'Male', 'Male', 'Female', 'Male', 'Male', 'Male', 'Male', 'Female', 'Female']




```
#BUAT FUNGSI TRAIN_MODEL(...)

import time
from torch.autograd import Variable
import torch

def train_model(model, dataloaders, dataset_sizes, criterion, optimizer, use_gpu=True, num_epochs=10):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    print("Using device:", device)

    model = model.to(device)

    train_acc_list = []
    val_acc_list = []
    #buat full loop training
    for epoch in range(num_epochs):
        print(f"\nEpoch {epoch+1}/{num_epochs}")
        print("-" * 20)

        for phase in ['train', 'val']:
            if dataloaders[phase] is None or dataset_sizes[phase] == 0:
                 print(f"Skipping phase '{phase}' as dataloader is empty or None.")
                 continue

            if phase == 'train':
                model.train()
            else:
                model.eval()  #Loop kedua: bedain fase training dan validasi.

            running_loss = 0.0 #Variabel untuk menyimpan total loss dan jumlah prediksi benar per epoch.
            correct = 0

            for batch_data in dataloaders[phase]:
                inputs, labels = batch_data
                if inputs is None or labels is None:
                    print(f"Warning: Skipping a batch with None values in {phase} phase.")
                    continue # Skip this batch if it contains None values

                inputs, labels = inputs.to(device), labels.to(device)
                optimizer.zero_grad()  #buat reset gradien dari batch sebelumnya

                with torch.set_grad_enabled(phase == 'train'): #nyalain gradien pas training aja, bukan val
                    outputs = model(inputs)   #forward pass --> hasil dari prediksi model
                    loss = criterion(outputs, labels) #hitung selisih prediksi vs label
                    _, preds = torch.max(outputs, 1) #ambil prediksi skor tertinggi untuk klasiifikasi

                    if phase == 'train':
                        loss.backward() #buat hitung gradien
                        optimizer.step() #update bobot model sesuai gradien

                running_loss += loss.item() * inputs.size(0)  #Tambah loss batch ke running_loss.
                correct += torch.sum(preds == labels.data) #hitung brp prediksi benar dan tmbh ke correct

            #hitung loss rata2 dan akurasi buat semua data di fase train/val
            if dataset_sizes[phase] > 0:
              epoch_loss = running_loss / dataset_sizes[phase]
              epoch_acc = correct.double() / dataset_sizes[phase]
            else:
              epoch_loss = 0.0
              epoch_acc = 0.0


            if phase == 'train':
              train_acc_list.append(round(epoch_acc.item() * 100, 1))
            else:
              val_acc_list.append(round(epoch_acc.item() * 100, 1))
```

```
            #tampilkan hasil loss & akurasi di console.
            print(f"{phase.capitalize()} Loss: {epoch_loss:.4f} Acc: {epoch_acc:.4f}")

            if phase == 'train':
              train_acc_list.append(round(epoch_acc.item() * 100, 1))
            else:
              val_acc_list.append(round(epoch_acc.item() * 100, 1))

    print("\ntrain_acc =", train_acc_list)
    print("test_acc  =", val_acc_list)
    return model



# Buang semua baris dengan NaN
combined_df = combined_df.dropna(subset=['gender'])

# Pastikan hanya angka 0 atau 1
combined_df = combined_df[combined_df['gender'].isin([0, 1])]
combined_df['gender'] = combined_df['gender'].astype(int)

# Reset index
combined_df = combined_df.reset_index(drop=True)

print("Cleaned Combined DF:", combined_df.shape)
```

```
Cleaned Combined DF: (5000, 3)
```

```
#DEFINE FUNGSI TRAIN
model_ft = train_model(model_ft, dataloaders, dataset_sizes, criterion, optimizer_ft,num_epochs=10)
```

```
Using device: cuda

Epoch 1/10
--------------------
Train Loss: 0.0960 Acc: 0.9660
Val Loss: 1.7096 Acc: 0.5210

Epoch 2/10
--------------------
Train Loss: 0.1062 Acc: 0.9590
Val Loss: 1.7372 Acc: 0.5110

Epoch 3/10
--------------------
Train Loss: 0.0680 Acc: 0.9743
Val Loss: 1.7889 Acc: 0.5630

Epoch 4/10
--------------------
Train Loss: 0.0395 Acc: 0.9880
Val Loss: 1.8023 Acc: 0.5390

Epoch 5/10
--------------------
Train Loss: 0.0740 Acc: 0.9718
Val Loss: 2.0274 Acc: 0.4930

Epoch 6/10
--------------------
Train Loss: 0.1053 Acc: 0.9573
Val Loss: 1.8651 Acc: 0.5100

Epoch 7/10
--------------------
Train Loss: 0.0687 Acc: 0.9753
Val Loss: 1.6987 Acc: 0.5200

Epoch 8/10
--------------------
Train Loss: 0.0274 Acc: 0.9918
Val Loss: 1.7467 Acc: 0.5280

Epoch 9/10
--------------------
Train Loss: 0.0182 Acc: 0.9950
Val Loss: 1.8220 Acc: 0.5470

Epoch 10/10
--------------------
Train Loss: 0.0079 Acc: 0.9980
Val Loss: 1.8853 Acc: 0.5460

train_acc = [96.6, 96.6, 95.9, 95.9, 97.4, 97.4, 98.8, 98.8, 97.2, 97.2, 95.7, 95.7, 97.5, 97.5, 99.2, 99.2, 99.5, 99.5, 99.8, 99.8
test_acc  = [52.1, 52.1, 51.1, 51.1, 56.3, 56.3, 53.9, 53.9, 49.3, 49.3, 51.0, 51.0, 52.0, 52.0, 52.8, 52.8, 54.7, 54.7, 54.6, 54.6
```

```
matched_files = set(identity_df['image_filename']) & set(os.listdir(image_folder))
print(f"Jumlah file yang match: {len(matched_files)}")
print(list(matched_files)[:10])
```

```
Jumlah file yang match: 5000
['058533.jpg', '125026.jpg', '195102.jpg', '153972.jpg', '154394.jpg', '039594.jpg', '053074.jpg', '062024.jpg', '188299.jpg', '0115
```

EVALUATION

```
import random
import os
from PIL import Image
import matplotlib.pyplot as plt
import torch
from torchvision import transforms

# Set path ke folder gambar
image_folder = os.path.join(data_path, 'Images')

# Ambil 1 file random dari folder
random_image_name = random.choice(os.listdir(image_folder))
image_path = os.path.join(image_folder, random_image_name)

# Transformasi harus sama dengan yang dipakai waktu training
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# Load gambar dan apply transform
image = Image.open(image_path).convert('RGB')
input_tensor = transform(image).unsqueeze(0).to(device)  # tambah batch dimensi dan kirim ke device

# Prediksi
model_ft.eval()  # pastikan model dalam mode eval
with torch.no_grad():
    output = model_ft(input_tensor)
    _, pred = torch.max(output, 1)

# Tampilkan hasil
label_map = {0: 'Female', 1: 'Male'}
predicted_label = label_map[pred.item()]

# Show
plt.imshow(image)
plt.title(f'Predicted: {predicted_label}')
plt.axis('off')
plt.show()
```

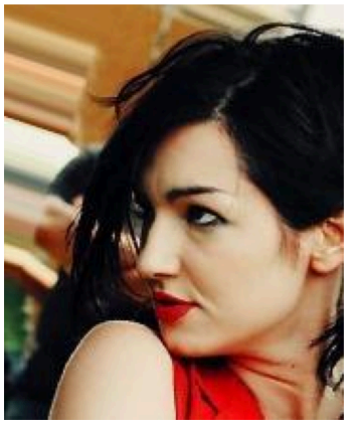
```
fig, axes = plt.subplots(2, 3, figsize=(12, 8))

for i, ax in enumerate(axes.flat):
    if i < len(combined_df):
      row = combined_df.iloc[i]
      img_path = os.path.join(image_folder, row['image_filename'])
      image = Image.open(img_path)

      gender_label = 'Male' if row['gender'] == 1 else 'Female'
      ax.imshow(image)
      ax.set_title(f"Gender: {gender_label}")
      ax.axis('off')
    else:
      break

plt.tight_layout()
plt.show()
```

TOP 5

```
import matplotlib.pyplot as plt
from torchvision import transforms
from PIL import Image
import torch
import os
import random

# Define transform (sama seperti saat training)
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

# Ambil 6 gambar acak dari folder
sample_images = random.sample(os.listdir(image_folder_path), 6)
```

```
# Siapkan gambar & tensor batch
images = []
original_images = []  # For displaying
for filename in sample_images:
    img_path = os.path.join(image_folder_path, filename)
    image = Image.open(img_path).convert('RGB')
    original_images.append(image)
    images.append(transform(image))

# Convert jadi batch tensor
batch_tensor = torch.stack(images).to(device)

# Set model ke eval
model_ft.eval()

# Inference (no grad)
with torch.no_grad():
    outputs = model_ft(batch_tensor)
    probs = torch.softmax(outputs, dim=1)
    top_probs, top_classes = probs.topk(2, dim=1)

# Tampilkan gambar dan top-5 prediksi
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
for i, ax in enumerate(axes.flat):
    if i < len(original_images):
        ax.imshow(original_images[i])
        top_preds = top_classes[i].cpu().numpy()
        top_scores = top_probs[i].cpu().numpy()

        label_text = '\n'.join([f'{int(cls)}: {score:.2f}' for cls, score in zip(top_preds, top_scores)])
        ax.set_title(f"Top-5:\n{label_text}")
        ax.axis('off')
plt.tight_layout()
plt.show()
```

Top-5:
0: 1.00
1: 0.00



Top-5:
1: 1.00
0: 0.00

Top-5:
1: 1.00
0: 0.00

Top-5:
1: 1.00
0: 0.00

Top-5:
0: 1.00
1: 0.00

Top-5:
0: 0.91
1: 0.09