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

```
Mounted at /content/drive
```

```
import torch
from torchvision import transforms
from torch.utils.data import DataLoader, Dataset
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from PIL import Image
import os
import numpy as np
import pandas as pd
from torch.autograd import Variable
import matplotlib.pyplot as plt
```

```
is_cuda = torch.cuda.is_available()
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
# colab 환경에서 학습 시킴
print('Current cuda device is', device)
```

```
Current cuda device is cuda
```

```
data_dir = '/content/drive/MyDrive/Deaplearning/d2l_notebooks/d2l-en/pytorch/D2L_assignment/tr'
file_list = os.listdir(data_dir)  # 이미지 파일 리스트를 가져옵니다.
# 데이터 전처리 및 DataLoader 설정
transform = transforms.Compose([
    transforms.Resize((200, 200)),  # imgae 크기 조정
    transforms.ToTensor(),      # image를 tensor로 변환
    # image 정규화 (평균 0.5, 표준 편차 0.5)
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])
])

class InitDataset(Dataset):
    def __init__(self, file_list, transform=None):
        self.file_list = file_list  # 이미지 파일 리스트 생성자
        self.transform = transform  # transform 된 이미지 파일 리스트 생성자

    def __len__(self):    # 이미지 리스트 총 합계 초기화
        return len(self.file_list)

    def __getitem__(self, idx):   # 이미지 파일 이름에서 나이만 split해서 추출
        img_path = os.path.join(data_dir, self.file_list[idx])
        image = Image.open(img_path)
        # 파일명에서 나이 정보만 split 추출
        age = int(self.file_list[idx].split("_")[0])

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

        return image, age

# 데이터 분할
train_files, val_files = train_test_split(file_list, test_size=0.2, random_state=42)

# DataLoader 설정
train_dataset = InitDataset(train_files, transform=transform)
val_dataset = InitDataset(val_files, transform=transform)
batch_size = 10   # batch size
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size)
```

```
import torch.nn as nn

class Image_to_AgePredictionModel(nn.Module):
    def __init__(self):
        super(Image_to_AgePredictionModel, self).__init__()
        # 입력 이미지 크기에 맞는 Fully Connected 레이어
        self.fc = nn.Linear(200 * 200 * 3, 1)
    def forward(self, x):
        x = x.view(x.size(0), -1)  # Flatten 작업
        x = self.fc(x)
        return x

model = Image_to_AgePredictionModel()
model.to(device)
```

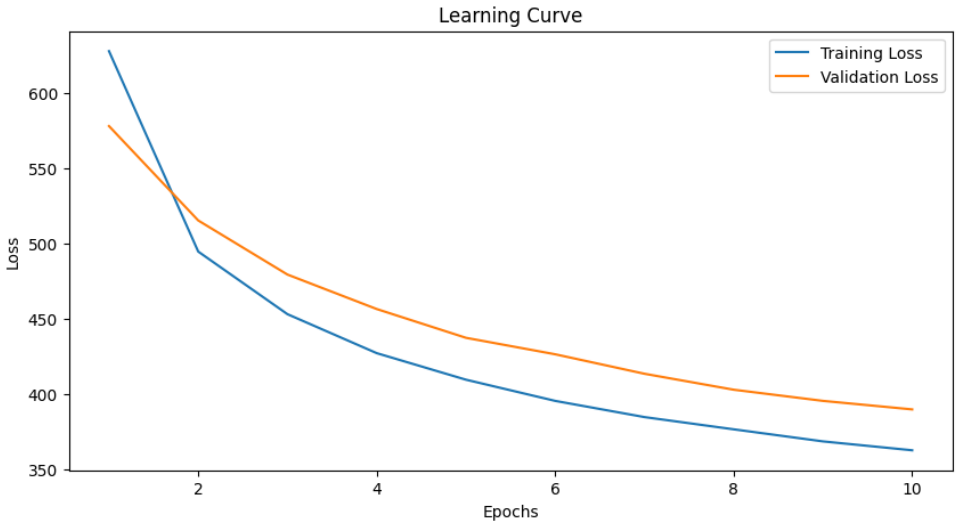
```
Image_to_AgePredictionModel(
  (fc): Linear(in_features=120000, out_features=1, bias=True)
)
```

```
"""
에폭(epoch) 수 - 데이터셋을 반복하는 횟수

배치 크기(batch size) - 매개변수가 갱신되기 전 신경망을 통해 전파된 데이터 샘플의 수

학습률(learning rate) - 각 배치/에폭에서 모델의 매개변수를 조절하는 비율.
값이 작을수록 학습 속도가 느려지고, 값이 크면 학습 중 예측할 수 없는 동작이 발생할 수 있습니다.
"""
criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.000001)


# Hyperparameter 설정
num_epochs = 10

# Learning Curve를 그리기 위한 리스트 초기화
train_loss_mat = []
val_losses_mat = []
train_accuracy_values = []
val_accuracy_values = []


for epoch in range(num_epochs):
    model.train()
    running_loss = 0.0
    train_correct = 0
    train_total = 0
    for images, labels in train_loader:
        images , labels= images.to(device), labels.to(device)  # cuda로 모델 학습
        #  초기설정은 매번 gradient를 더해주는 것으로 설정되어있습니다.
        # 그렇기 때문에 학습 loop를 돌때 이상적으로 학습이 이루어지기 위해선
        # 한번의 학습이 완료되어지면(즉, Iteration이 한번 끝나면)
        # gradients를 항상 0으로 만들어 주어야 합니다.
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs.view(-1), labels.float())   # view 메소드 수정
        # 오차 역전파 과정은 컴퓨터가 예측값의 정확도를 높이기 위해
        # 출력값과 실제 예측하고자 하는 값을 비교하여 가중치를 변경하는 작업
        loss.backward()
        optimizer.step()
        running_loss += loss.item()

        predicted_ages = outputs.round().squeeze()
        train_correct += (predicted_ages == labels).sum().item()
        train_total += labels.size(0)
    # train set accuracy
    train_accuracy = train_correct / train_total
    train_accuracy_values.append(train_accuracy)

    # 훈련 손실 출력
    print(f"Epoch {epoch+1}/{num_epochs}, Training Loss: {running_loss/len(train_loader)}")
    # 훈련 손실 기록
    train_loss = running_loss / len(train_loader)
    train_loss_mat.append(train_loss)
    # 검증 데이터셋에 대한 손실 계산
    model.eval()
    val_loss = 0.0
    val_correct = 0
    val_total = 0
    with torch.no_grad():
        for images, labels in val_loader:
            images , labels= images.to(device), labels.to(device)
            outputs = model(images)
            loss = criterion(outputs.view(-1), labels.float())# view 메소드 수정
            val_loss += loss.item()

            predicted_ages = outputs.round().squeeze()
            val_correct += (predicted_ages == labels).sum().item()
            val_total += labels.size(0)

     # 검증 손실 기록
    val_loss = val_loss / len(val_loader)
    val_losses_mat.append(val_loss)
    # validation set accuracy
    val_accuracy = val_correct / val_total
    val_accuracy_values.append(val_accuracy)

    # 에폭별로 손실 출력
    print(f"Epoch {epoch+1}/{num_epochs}, Validation Loss: {val_loss/len(val_loader)}")
```

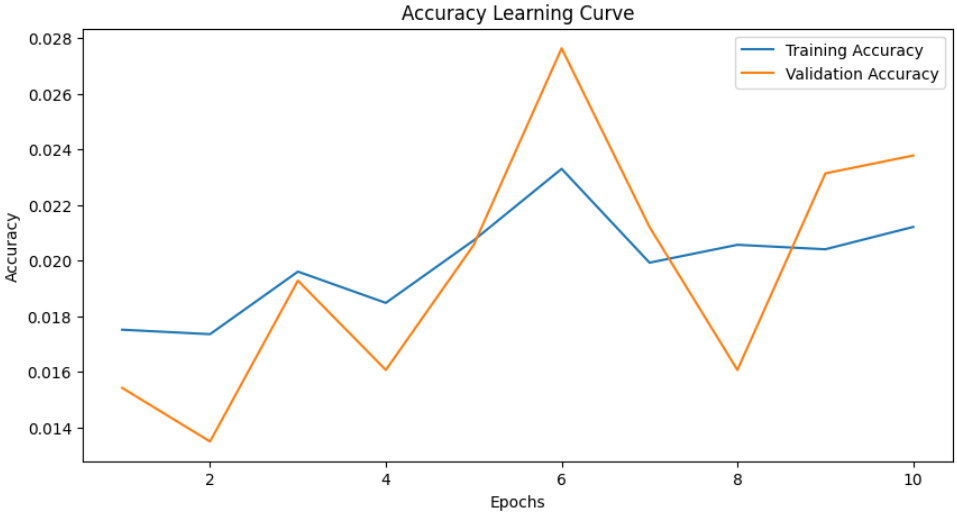
```
Epoch 1/10, Training Loss: 627.650149056081
Epoch 1/10, Validation Loss: 3.7041123554473017
Epoch 2/10, Training Loss: 494.43118294705164
Epoch 2/10, Validation Loss: 3.3019823491926026
Epoch 3/10, Training Loss: 452.82359383768295
Epoch 3/10, Validation Loss: 3.071464480262771
Epoch 4/10, Training Loss: 426.9268256565541
Epoch 4/10, Validation Loss: 2.9246883586705477
Epoch 5/10, Training Loss: 409.28596132869325
```

```
Epoch 5/10, Validation Loss: 2.8021028121782714
Epoch 6/10, Training Loss: 395.2051257689156
Epoch 6/10, Validation Loss: 2.731906110248152
Epoch 7/10, Training Loss: 384.43821090526797
Epoch 7/10, Validation Loss: 2.6489740982406786
Epoch 8/10, Training Loss: 376.3090027033039
Epoch 8/10, Validation Loss: 2.580980048847387
Epoch 9/10, Training Loss: 368.28375545397614
Epoch 9/10, Validation Loss: 2.5333963515803974
Epoch 10/10, Training Loss: 362.37728581803566
Epoch 10/10, Validation Loss: 2.4970885492484087
```

```
# Learning Curve 그리기
plt.figure(figsize=(10, 5))
plt.plot(range(1, num_epochs + 1), train_loss_mat, label='Training Loss')
plt.plot(range(1, num_epochs + 1), val_losses_mat, label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Learning Curve')
plt.show()
```

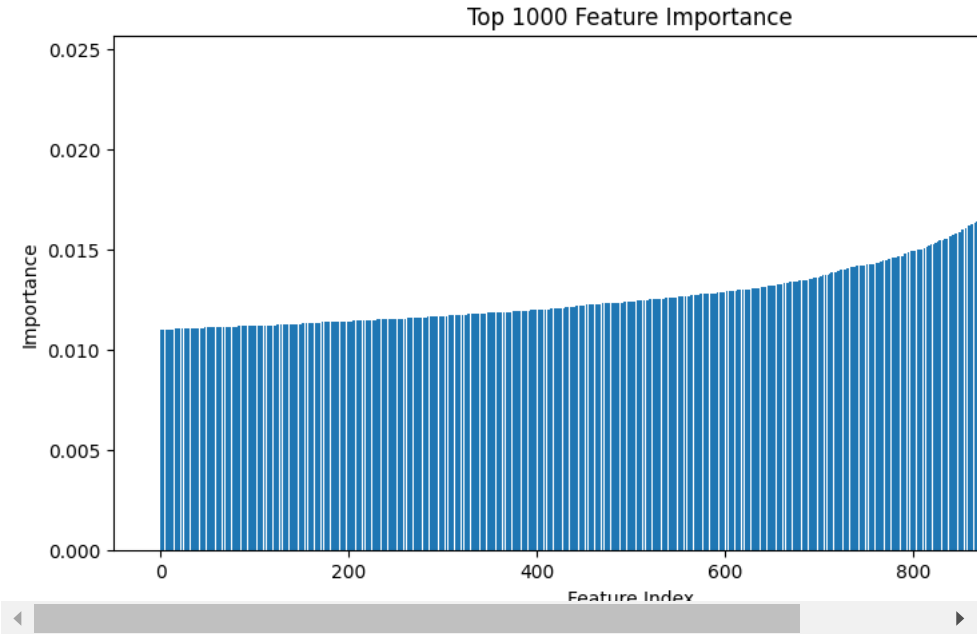



```
plt.figure(figsize=(10, 5))
plt.plot(range(1, num_epochs + 1), train_accuracy_values, label='Training Accuracy')
plt.plot(range(1, num_epochs + 1), val_accuracy_values, label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy Learning Curve')
plt.show()
```

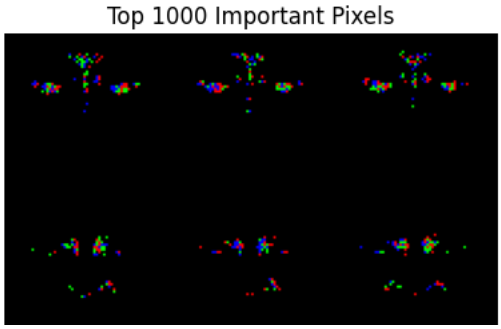



```
# 훈련 데이터셋에서 랜덤하게 이미지 선택
sample_image, _ = train_dataset[np.random.randint(len(train_dataset))]

# 모델의 forward pass 수행
model.eval()
```

```
with torch.no_grad():
    sample_image = sample_image.unsqueeze(0).to(device)
    outputs = model(sample_image)

# Feature Importance weight 적용하여 계산 및 CUDA 이동
ft_importance_map = torch.abs(model.fc.weight.squeeze(0)).detach().cpu().numpy()
ft_importance_map = torch.from_numpy(ft_importance_map).to(device)

# 상위 1000개 feature의 중요도 순으로 정렬
top_1000_features = np.argsort(ft_importance_map.cpu().numpy())[-1000:]

# 상위 1000개 feature 시각화
plt.figure(figsize=(10, 5))
plt.bar(range(len(top_1000_features)), ft_importance_map[top_1000_features].cpu())
plt.xlabel('Feature Index')
plt.ylabel('Importance')
plt.title('Top 1000 Feature Importance')
plt.show()
```




```
# 위의 ft_importance_map에소 1000개 나열을 위한 생성
thousand_pix_of_top = np.argsort(ft_importance_map.cpu().numpy())[-1000:]

# 200*200 빈 이미지 생성
empty_image = np.zeros((200, 200, 3), dtype=np.uint8)

# 채널에 대해 색 매핑
channel_colors = {
    0: (255, 0, 0),  # Red (R channel)
    1: (0, 255, 0),  # Green (G channel)
    2: (0, 0, 255),  # Blue (B channel)
}

#상위 1000픽셀을 반복하고 빈 이미지에 색상을 지정
for pixel_index in thousand_pix_of_top:
    # 픽셀 인덱스를 좌표와 채널로 변환
    channel = pixel_index % 3  # 0 -> R, 1 -> G, 2 -> B
    pixel_index //= 3
    row, col = divmod(pixel_index, 200)

    # 채널의 색상 대입
    color = channel_colors[channel]

   # 빈 이미지의 픽셀에 색상을 지정합니다.
    empty_image[row, col] = color

# display
plt.imshow(empty_image)
plt.title('Top 1000 Important Pixels')
plt.axis('off')
plt.show()
```




```
from torchsummary import summary
summary(model, input_size=(3,200,200), device='cuda')
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                     [-1, 1]         120,001
================================================================
Total params: 120,001
Trainable params: 120,001
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.46
Forward/backward pass size (MB): 0.00
Params size (MB): 0.46
Estimated Total Size (MB): 0.92
----------------------------------------------------------------
```