<Models>

S-Former, ST-Former, T-Former는 각각 공간(Spatial), 시간(Temporal), 공간-시간(Spatio-Temporal) 변환기 모델을 의미합니다. 이 모델들은 입력 데이터를 처리하는 방식에서 서로 다른 특성을 갖고 있습니다. 이를 통해 데이터를 다차원적으로 처리하고, 다양한 특징을 추출하는 데 도움을 줍니다.

**S\_Former**

이 코드는 PyTorch와 관련된 라이브러리들을 사용하여 Residual Network(ResNet)과 Transformer 구조를 결합한 모델을 정의. 기본적으로 ResNet 구조를 따르면서 마지막에 공간 정보를 변환하기 위해 Transformer를 사용..

**주요 라이브러리**

* **torch**: PyTorch의 핵심 라이브러리.
* **einops**: 텐서 조작을 간단하게 하기 위한 라이브러리.
* **torch.nn**: 신경망을 구성하는 다양한 레이어를 포함하는 모듈.
* **torch.nn.functional**: 다양한 함수들을 제공하는 모듈.
* **math**: 수학적 계산을 위한 모듈.

class GELU(nn.Module): **GELU** 활성화 함수를 정의합니다. (Gaussian Error Linear Units)

class Residual(nn.Module):주어진 함수 **fn**을 적용한 결과에 입력값을 더하는 잔차 연결(residual connection)을 구현

class PreNorm(nn.Module):입력값에 Layer Normalization을 적용한 후 주어진 함수 **fn**을 적용

class FeedForward(nn.Module):두 개의 선형 레이어와 GELU 활성화 함수를 포함하는 피드포워드 신경망을 정의

class Attention(nn.Module):멀티헤드 셀프 어텐션 메커니즘을 정의

class Transformer(nn.Module):주어진 깊이(depth)에 따라 여러 개의 어텐션 및 피드포워드 레이어를 쌓아 트랜스포머 블록을 구성

#### conv3x3 및 conv1x1 함수

3x3 및 1x1 컨볼루션 레이어를 정의

class BasicBlock(nn.Module):ResNet의 기본 블록을 정의

class ResNet(nn.Module):ResNet 구조를 정의하며, 마지막에 Transformer를 사용하여 공간 정보를 변환

* **spatial\_transformer** 함수

모델을 정의하고, 가짜 이미지 데이터를 생성하여 모델을 테스트

**ST\_Former**

이 코드는 PyTorch를 사용하여 공간 변환기(Spatial Transformer)와 시간 변환기(Temporal Transformer)를 결합한 모델을 정의.

이 코드의 목적은 공간 및 시간 변환기를 결합한 복합 모델을 정의하고, 해당 모델을 테스트하는 것입니다. 이 모델은 입력 이미지 데이터를 순차적으로 공간 변환기와 시간 변환기를 통해 처리하고, 최종적으로 선형 레이어를 통해 5차원 출력 벡터를 생성합니다.

### 주요 라이브러리

* **torch**: PyTorch의 핵심 라이브러리.
* **torch.nn**: 신경망을 구성하는 다양한 레이어를 포함하는 모듈.

class GenerateModel(nn.Module):

* 이 클래스는 **nn.Module**을 상속받아 사용자 정의 신경망 모델을 만듭니다.
* **\_\_init\_\_** 메서드에서 공간 변환기와 시간 변환기를 초기화하고, 최종적으로 512차원의 입력을 5차원 출력으로 변환하는 선형 레이어를 정의합니다.
* **forward** 메서드에서 입력 데이터를 순차적으로 공간 변환기와 시간 변환기에 통과시키고, 최종적으로 선형 레이어를 거쳐 출력을 생성합니다.

if \_\_name\_\_ == '\_\_main\_\_':

* **\_\_name\_\_**가 **'\_\_main\_\_'**일 때, 즉 이 스크립트가 직접 실행될 때만 실행되는 부분입니다.
* 가짜 이미지 데이터를 생성하여(**torch.randn((1, 16, 3, 112, 112))**) 모델의 **forward** 메서드를 호출하여 모델을 테스트합니다.

**T-former**

이 코드는 PyTorch를 사용하여 Temporal Transformer 모델을 정의하고 있습니다. 이 모델은 시간적 특성을 학습하는 데 중점을 둡니다.

입력 데이터에 CLS 토큰과 위치 임베딩을 추가하여 시간적 특성을 학습합니다

### 주요 라이브러리

* **torch**: PyTorch의 핵심 라이브러리.
* **einops**: 텐서 조작을 간단하게 하기 위한 라이브러리.
* **torch.nn**: 신경망을 구성하는 다양한 레이어를 포함하는 모듈.
* **math**: 수학적 계산을 위한 모듈.

class GELU(nn.Module):**GELU** 활성화 함수를 정의 (Gaussian Error Linear Units)

class Residual(nn.Module):주어진 함수 **fn**을 적용한 결과에 입력값을 더하는 잔차 연결(residual connection)을 구현

class PreNorm(nn.Module):입력값에 Layer Normalization을 적용한 후 주어진 함수 **fn**을 적용

class FeedForward(nn.Module):두 개의 선형 레이어와 GELU 활성화 함수를 포함하는 피드포워드 신경망을 정의

class Attention(nn.Module):멀티헤드 셀프 어텐션 메커니즘을 정의

class Transformer(nn.Module):주어진 깊이(depth)에 따라 여러 개의 어텐션 및 피드포워드 레이어를 쌓아 트랜스포머 블록을 구성

class TFormer(nn.Module):Temporal Transformer 모델을 정의합니다. CLS 토큰과 위치 임베딩을 추가하여 입력을 변환하고, Transformer를 통해 처리

#### temporal\_transformer 함수

**TFormer** 클래스의 인스턴스를 반환하는 함수를 정의

#### 모델 테스트

**\_\_name\_\_**가 **'\_\_main\_\_'**일 때, 즉 이 스크립트가 직접 실행될 때만 실행되는 부분입니다.

가짜 이미지 데이터를 생성하여(**torch.randn((1, 16, 3, 112, 112))**) 모델의 **forward** 메서드를 호출하여 모델을 테스트합니다.

<DataLoader>

**Dataset**

이 코드는 PyTorch를 사용하여 비디오 데이터셋을 처리하고 로드하는 데이터 로더를 정의

### 주요 라이브러리

* **os**: 파일 경로 및 디렉토리 작업을 위한 라이브러리.
* **numpy.random**: 난수 생성을 위한 라이브러리.
* **torch.utils.data**: PyTorch 데이터셋과 데이터 로더를 위한 모듈.
* **glob**: 파일 경로 패턴 매칭을 위한 라이브러리.
* **numpy**: 수학 및 배열 계산을 위한 라이브러리.
* **dataloader.video\_transform**: 비디오 변환 관련 모듈 (여기서는 상세히 설명되지 않음).

class VideoRecord(object):비디오의 메타데이터(경로, 프레임 수, 레이블)를 저장하는 클래스

class VideoDataset(data.Dataset):PyTorch 데이터셋 클래스를 상속받아 비디오 데이터를 처리하고 로드하는 역할. (**비디오 데이터셋을 처리하고 PyTorch 데이터셋 형식으로 사용하기 위한 클래스**) 비디오 목록 파일을 읽어 유효한 비디오 리스트를 생성하고, 각 비디오를 세그먼트로 나누어 프레임을 선택. 선택된 프레임은 지정된 이미지 변환을 거쳐 최종적으로 모델에 입력될 형태로 반환.

* **train\_data\_loader**: 학습 데이터를 로드하는 함수.
* **test\_data\_loader**: 테스트 데이터를 로드하는 함수.
* **custom\_data\_loader**: 사용자 지정 데이터를 로드하는 함수.

**Video\_Transform**

이 코드는 PyTorch를 사용하여 비디오 데이터셋을 처리하는 데 필요한 다양한 이미지 변환(transform) 클래스를 정의. 각 클래스는 이미지 그룹(여러 이미지 또는 프레임 그룹)에 대해 특정 변환 작업을 수행. 이러한 변환은 데이터 전처리와 데이터 증강(data augmentation)을 통해 모델 학습의 성능을 높이는 데 사용됩니다.

class GroupRandomCrop(object): 이미지 그룹에서 랜덤하게 크롭을 수행

class GroupCenterCrop(object):이미지 그룹에서 중앙 크롭을 수행하는 클래스

class GroupRandomHorizontalFlip(object): 이미지 그룹을 랜덤으로 좌우 반전하는 클래스

class GroupNormalize(object): 이미지 그룹을 정규화하는 클래스

class GroupScale(object):이미지 그룹을 주어진 크기로 리사이즈

class GroupOverSample(object): 이미지 그룹에 대해 오버샘플링을 수행하는 클래스

class GroupMultiScaleCrop(object):여러 크기의 이미지를 랜덤으로 크롭하는 클래스

class GroupResize(object):이미지 그룹을 주어진 크기로 리사이즈하는 클래스

class GroupRandomSizedCrop(object):이미지 그룹을 랜덤 크기와 비율로 크롭하는 클래스

class Stack(object): 이미지 그룹을 하나의 텐서로 쌓는 클래스

class ToTorchFormatTensor(object):이미지 또는 numpy 배열을 PyTorch 텐서 형식으로 변환하는 클래스

class IdentityTransform(object):입력 데이터를 그대로 반환하는 클래스

<Runner\_helper>-Train & Validate

이 코드는 PyTorch를 사용하여 비디오 데이터셋을 처리하고, 모델을 학습 및 평가하며, 학습 과정에서 발생하는 다양한 정보를 기록하고 관리하는 유틸리티 함수들을 정의

class AverageMeter(object):평균 값을 계산하고 저장하는 클래스

class ProgressMeter(object): 훈련 과정의 진행 상황을 출력하는 클래스

def accuracy(output, target, topk=(1,)):모델의 예측 정확도를 계산하는 함수

class RecorderMeter(object):훈련 과정의 손실과 정확도를 기록하고 저장하는 클래스

def train(train\_loader, model, criterion, optimizer, epoch, log\_txt\_path, args): 훈련 데이터를 사용하여 모델을 학습하는 함수

def validate(val\_loader, model, criterion, log\_txt\_path, args):검증 데이터를 사용하여 모델을 평가하는 함수

def save\_checkpoint(state, checkpoint\_path, best\_checkpoint\_path, is\_best): 모델의 체크포인트를 저장하는 함수

<Utils>-데이터 전처리 (감지, 크롭)

Mediapipe와 OpenCV를 사용하여 비디오 데이터에서 얼굴을 감지하고, 감지된 얼굴 영역을 크롭하여 저장하는 유틸리티 함수들을 정의. Mediapipe는 얼굴 감지를 위한 모델을 제공하고, OpenCV는 비디오 프레임을 읽고 이미지를 처리하는 데 사용됩니다. (Mediapipe는 얼굴 감지와 같은 고수준의 작업을, OpenCV는 이미지 및 비디오 파일을 읽고 쓰는 등의 저수준 작업에 유용)

state\_keys = {

"F": 0,

"S": 1,

"D": 2,

"A": 3,

"N": 4,

}이 딕셔너리는 비디오 파일 이름에서 특정 상태를 추출하여 해당 상태에 대한 키 값을 반환하는 데 사용

#### 2. crop\_face\_area 함수

얼굴 감지 결과를 바탕으로 이미지를 크롭합니다.

Mediapipe의 얼굴 감지 결과 사용

#### get\_state 함수

비디오 파일 이름에서 상태 키를 추출합니다.

#### crop\_frames 함수

비디오에서 프레임을 추출하고 얼굴을 크롭하여 저장합니다.

<Valid>

PyTorch를 사용하여 학습된 모델을 검증하는 스크립트입니다. 이 스크립트는 사전 학습된 모델을 로드하고, 검증 데이터셋을 사용하여 모델의 성능을 평가

#### 1. 환경 설정

CUDA 장치를 설정을 통해 GPU사용을 지정하고 필요한 라이브러리를 임포트

#### 2. 검증 함수 정의

검증 데이터셋을 사용하여 모델을 평가하는 함수입니다.

def test(val\_loader, model, criterion, args):

#### 3. Pseudoarg 클래스 정의

검증에 필요한 인자들을 저장하는 클래스

class Pseudoarg():

#### 4. 초기 설정 및 모델 로드

모델을 초기화하고, 사전 학습된 파라미터를 로드

#### 5. 데이터 로드 및 검증 수행

valid 데이터셋을 로드하고, 모델을 평가합니다. 소요시간 출력

how to run:

1. write ‘python process.py’: util의 전처리 함수 중 crop\_frames 이용하여, 얼굴 감지 및 크롭한 이미지를 저장

2. 데이터셋 준비: dataset.py를 사용. train.txt와 test.txt파일을 생성하여 데이터셋 구분

3. write ‘train.py’

4. use ‘valid.py’