rPPG Overview

김 대 열

2022.09.30

시작하기 전에 실습을 위해 UBFC 데이터 일부 다운로드

scp -P 12222 -r an@175.209.36.34 :/media/hdd1/dy/dataset/PhysNet_UBFC_train_9.hdf5 로컬 경로

- 비밀번호 : 1324

- https://github.com/TVS-AI/Pytorch_rppgs star & clone 부탁 드립니다.

- git clone git@github.com:TVS-AI/Pytorch_rppgs.git

PPG(Photoplethysmography)

- 혈류를 관찰 하는 방법
- 혈액이 손가락 끝에 닿기 까지는 1~2초가 걸리지만 맥파(Plethysmogram, PTG)는 0.16초 소모
- 빛으로 맥파에 따라 변하는 미세한 혈류량을 측정하면 혈액량의 변화를 알 수 있음
- 빛으로(Photo) 변화를(Plethysmo) 기록한다(Graphy)

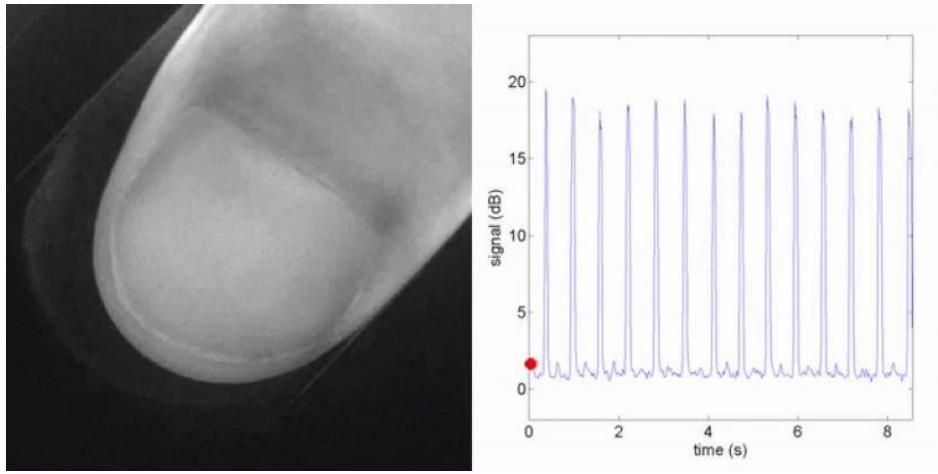


Fig 1. ppg 측정 방안

1. LED로 심박수를 측정한다고? https://news.samsungdisplay.com/30140

PPG(Remote Photoplethysmography)

- 혈류의 성분
  - oxyHb, 45~70%
  - deoxyHb, 0%~5%(동맥) 15%~40%(정맥)
  - MetHb, 0% ~ 1.5%
  - COHb 0~2.5%(비흡연자) 1.5%~10%(흡연자)

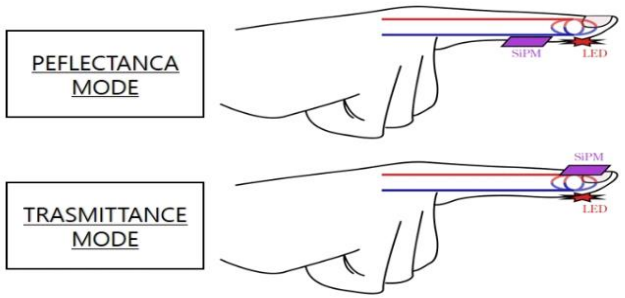


Fig 2. 투과형 및 반사형 Oximeter

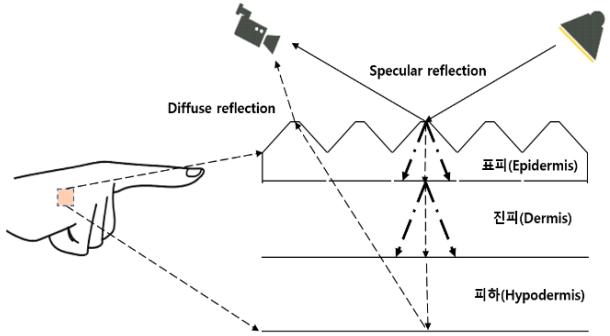


Fig 3. 피부 반사 모델

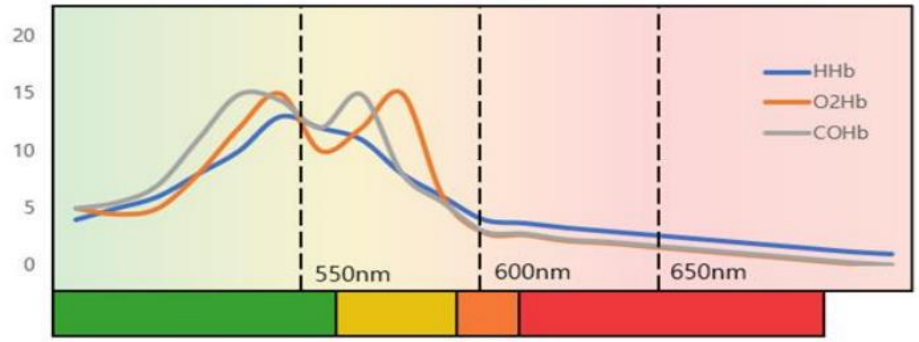


Fig 4. 빛의 파장에 따른 빛 흡수량

2. 모바일 환경에서 안면 영상을 이용한 실시간 생체징후 측정 시스템

rPPG(Remote Photoplethysmography)

- rPPG
  - 측정 부위 : 손 -> 얼굴
  - 측정 기기 : Oximeter -> camera

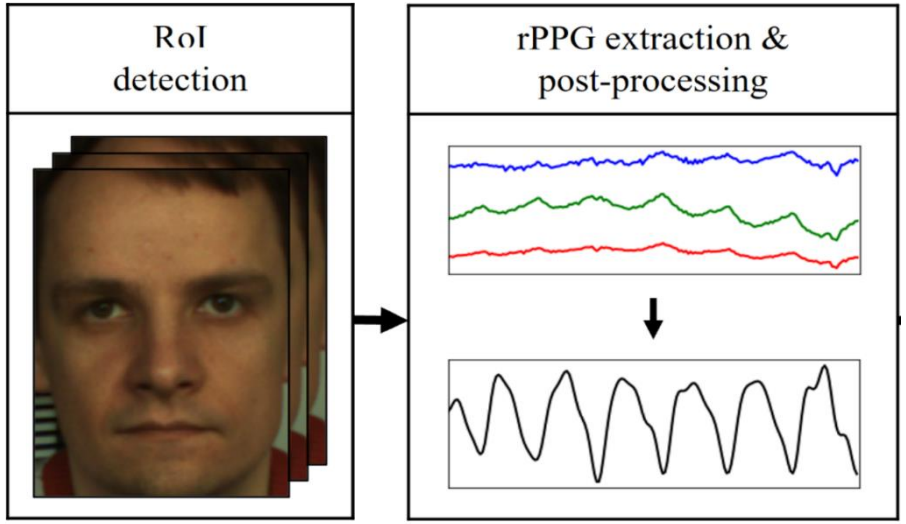


Fig 5. rPPG 모델

rPPG(Remote Photoplethysmography)

- 얼굴의 혈류의 변화 감지를 통해
  다양한 활력징후(Vital Sign)가 측정 가능함
  - 측정 가능한 Vital Sign
    - PPG (Photoplethysmography)
      - HR (Heart Rate)
      - RR (Respiration Rate)
      - BP (Blood Pressure)
      - Stress
    - SPo2 (Saturation of peripheral O2)

rPPG(Remote Photoplethysmography)

- rPPG
  - 측정된 파형을 가공하여 추가 정보를 생성할 수 있음.
  - 일반적으론 주파수 필터를 이용해서 정보를 추가 추출

  - $0.02 - 0.17Hz$: 산소포화도
  - $0.18 - 0.40Hz$: 호흡수
  - $0.80 - 4.00Hz$: 심장박동 수

  - $\leq 0.003\ Hz$: 급성 심장마비 및 부정맥 신호
  - $0.033Hz\ -\ 0.04\ Hz$: 레닌 안티오텐신계 시스템 변수 (혈압, 신장 관련 정보)
  - $0.04Hz\ -\ 0.15\ Hz$: 교감 신경계와 부교감 신경계에 의해 조절
  - $0.15Hz\ -\ 0.4\ Hz$: 호흡과 관련된 심박 변이율

2. 모바일 환경에서 안면 영상을 이용한 실시간 생체징후 측정 시스템

SPo2

- 혈류의 성분
  - oxyHb, 45~70%
  - deoxyHb, 0%~5%(동맥) 15%~40%(정맥)

- 동맥혈과 정맥혈의 색상
  - 철 이온의 전자 상태에 따른 색상 변화
    - OxyHb : 적색
    - deOxyHb : 어두운 붉은 보라색

3. Kienle A, Lilge L, Vitkin AI, et al. Why do veins appear blue? A new look at an old question. Appl Optics 1996; 35:1151-1160. (Maybe more than you wanted to know, but here it is).

rPPG Challenge

- 2가지 rPPG approach
  - V4V(Vision For Vitals)
    - PPG 신호의 형태에 집중
    - 평가 지표 : 신호 correlation의 rmse, mae, correlation factor
  - RePSS(Remote Physiological Signal Sensing)
    - PPG 신호로 뽑아 낼 수 있는 추가적인 생체 징후
    - 혹은 얼굴 영상으로부터 바로 추가적인 생체 징후를 뽑아내고자 함

V4V (Vision for Vital)

- V4V
  - APRL 데이터를 주로 이용하는 Daniel Mcduff 주관.
  - https://competitions.codalab.org/competitions/31978#learn_the_details-evaluation
  - 1st Challenge : ICCV 2021
    - 목표 : 파형의 correlation , 심박, 호흡 측정이 목표

RePSS(Remote Physiological Signal Sensing)

- RePSS
  - OBF 데이터 셋을 만든 필란드 Oulu 대학의 Xiaobai Li, VIPL-HR 데이터 셋을 만든 Hu han이 주최
  - https://competitions.codalab.org/competitions/?q=Repss
  - 1st Challenge : CVPM2020
    - 목표 : 평균 심장박동 수 측정 https://openaccess.thecvf.com/CVPR2020_workshops/CVPR2020_w19
  - 2nd Challenge : ICCV 2021
    - Track 1 : Mean IBI(InterBeat Interval), Track 2 : MAE RR https://openaccess.thecvf.com/ICCV2021_workshops/RePSS
  - 3rd Challenge : ICCV 2023(예정)

대표 연구

- Methods
  - McDuffs Approach
    - DeepPhys[3] : Dyschromatic Reflection Model(DRM)을 이용한 전처리 기법 제안 (빛을 고려한 모델)
    - High order[4] : LVET의 특성 + DRM을 이용한 변곡점 학습 방안을 제안 (심장박동의 주기를 고려한 모델)
  - Zithong Yu Approach
    - PhysNet[5] : 3D Convolution을 이용하여 시공간적인 특징을 학습하는 방법 제안(Deep Learning Module로써 접근)
    - PhysFormer[6] : Tube based Vision Transformer방법 제안(ViT를 사용하는 방법 제안)
  - STMap Approach
    - RhythmNet[7] : STMAP을 이용한 방법 제안(전처리 기법 제안)

4. DeepPhys: Video-Based Physiological Measurement Using Convolutional Attention Networks https://arxiv.org/abs/1805.07888
5. Learning Higher-Order Dynamics in Video-Based Cardiac Measurement https://arxiv.org/pdf/2110.03690.pdf
6. Remote Photoplethysmograph Signal Measurement from Facial Videos Using Spatio-Temporal Networks : https://arxiv.org/abs/1905.02419
7. PhysFormer: Facial Video-based Physiological Measurement with Temporal Difference Transformer : https://arxiv.org/abs/2111.12082
8. RhythmNet: End-to-end Heart Rate Estimation from Face via Spatial-temporal Representation : https://arxiv.org/abs/1910.11515

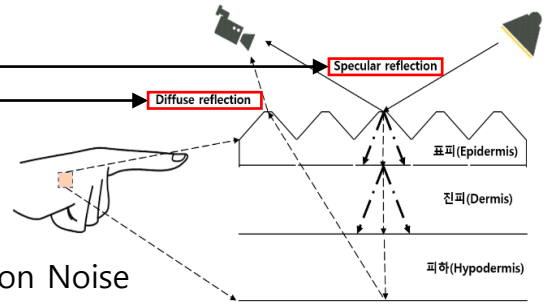
대표 연구 -mcduff

- DeepPhys
  - Dyschromatic Reflection Model(DRM)을 이용한 전처리 기법 제안 (빛을 고려한 모델)

$$C_k(t) = I(t) \cdot (v_s(t) + v_d(t)) + v_n(t)$$

Luminance Intensity Level

Camera Quantization Noise



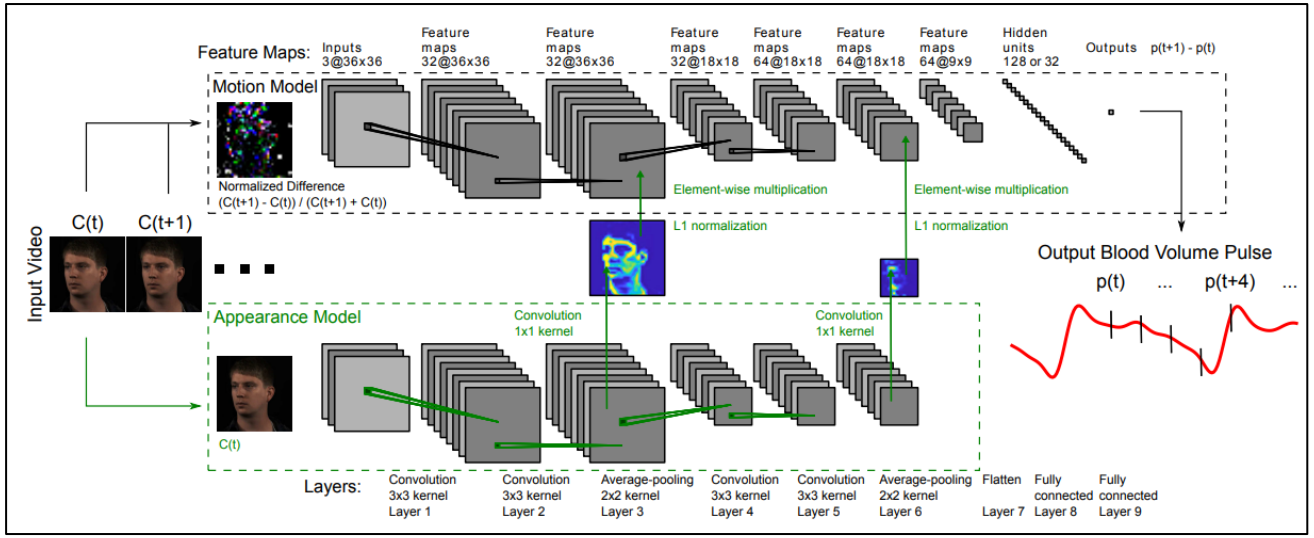


Fig 6. DeepPhys network

대표 연구 - mcduff

- High order
  - BVP를 두 번 미분하면 LVET를 쉽게 구할 수 있는 점을 착안
  - LVET는 심장의 상태를 관찰하는 요소
    - 허혈성 심장 질환, 심부전, 고혈압 및 대동맥 협착증 환자에서 LVET의 기울기가 감소, 심장 수축 기간이 늘어남

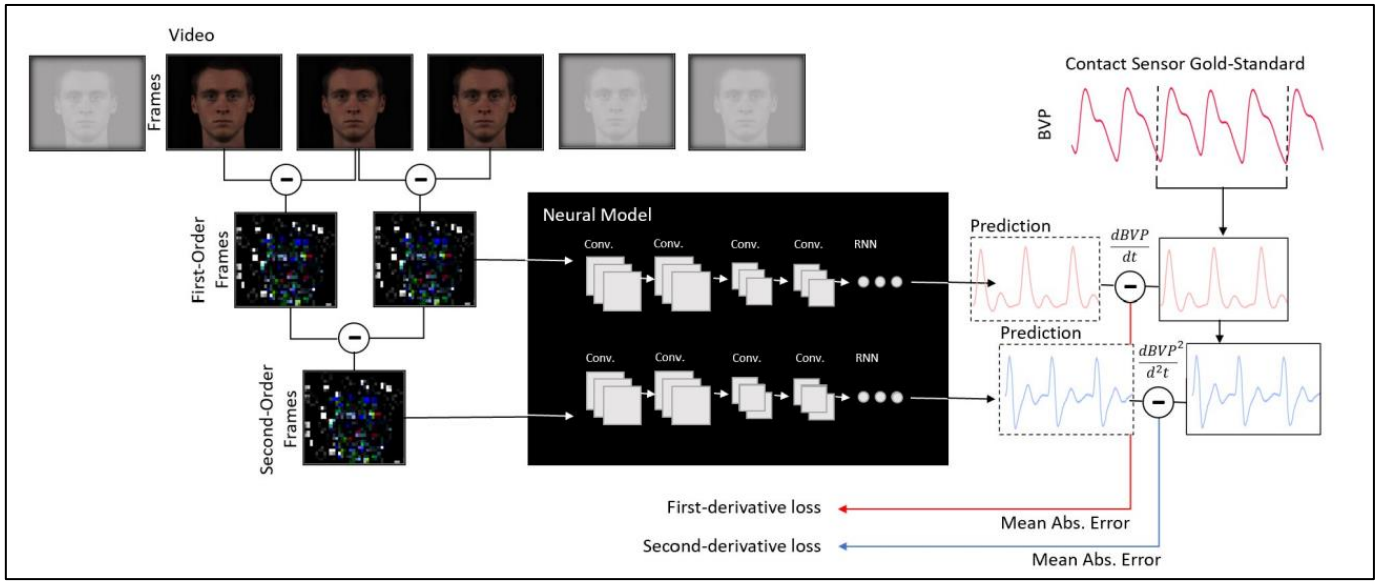


Fig 7. High order network

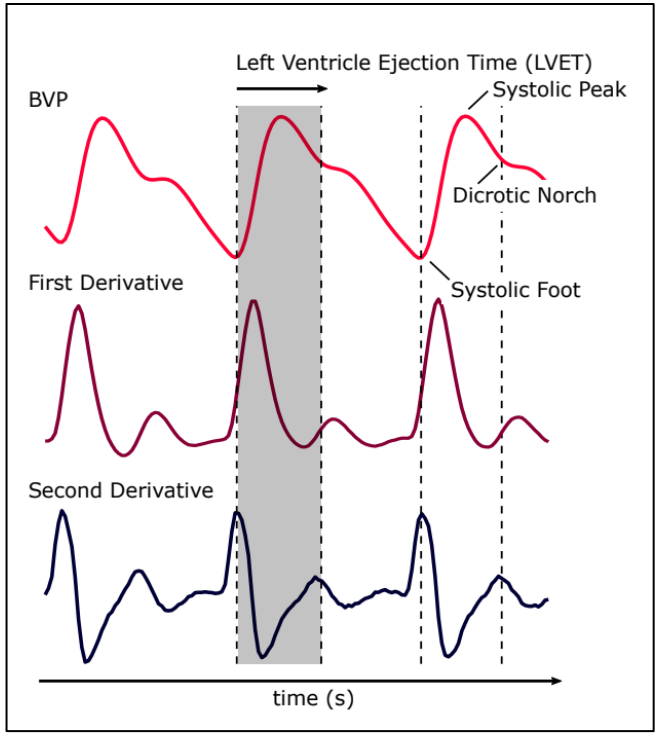


Fig 8. LVET

대표 연구 –zithong yu

- Physnet
  - 3D CNN이 시공간적 특징을 학습하는 점을 이용, 얼굴 영상을 분석하는 방법으로 접근

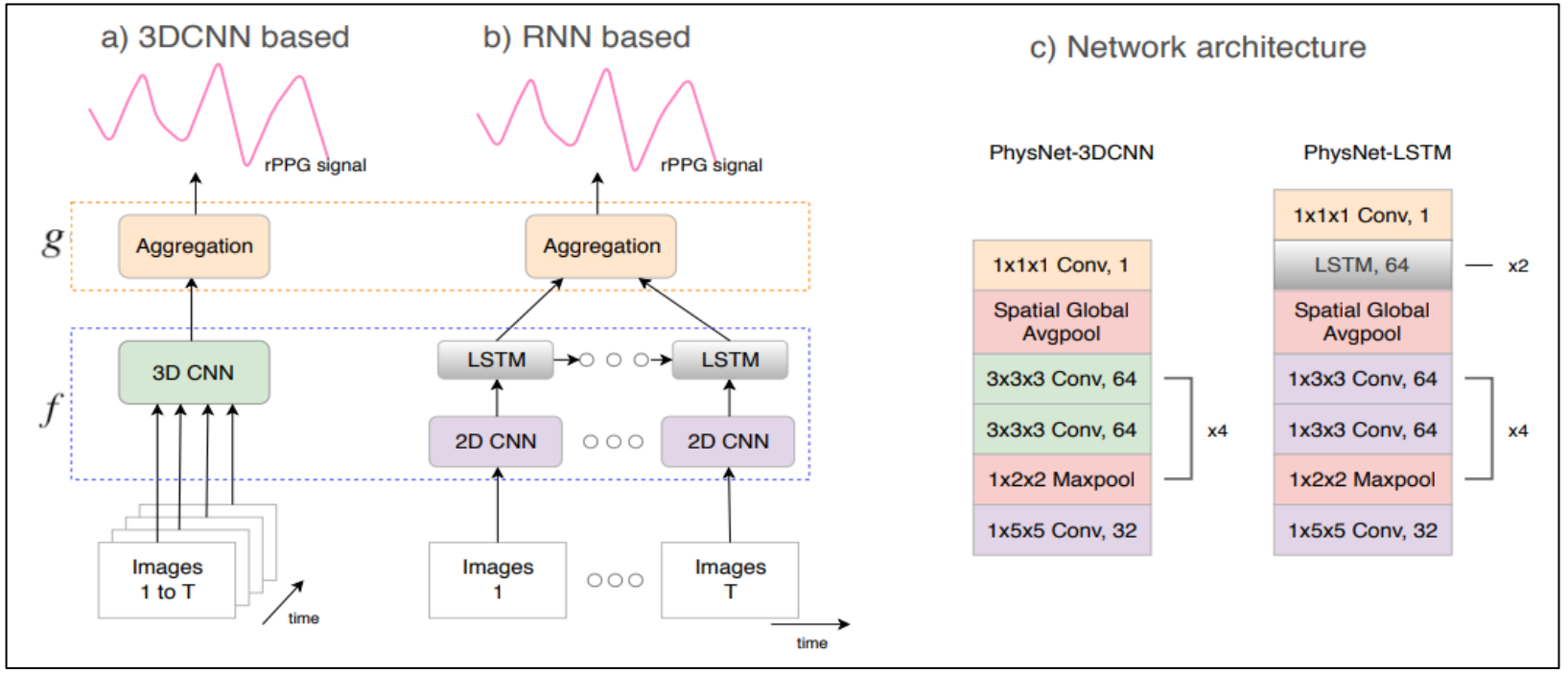


Fig 9. PhysNet Network Architecture

대표 연구 –zithong yu

- Physnet
  - Neg-Pearson loss 제안

Table 1: Performance comparison of two loss functions with negative Pearson and MSE. Smaller RMSE and bigger R values indicate better performance.

	HR(bpm)		RF(Hz)		LF(u.n)		HF(u.n)		LF/HF	
Method	RMSE	R	RMSE	R	RMSE	R	RMSE	R	RMSE	R
PhysNet64-3DCNN-MSE	4.012	0.955	0.069	0.435	0.169	0.689	0.169	0.689	0.721	0.659
PhysNet64-3DCNN-NegPea	2.143	0.985	0.067	0.494	0.15	0.749	0.15	0.749	0.647	0.72

Table 2: Performance comparison of spatio-temporal networks.

	HR(bpm)		RF(Hz)		LF(u.n)		HF(u.n)		LF/HF	
Method	RMSE	R	RMSE	R	RMSE	R	RMSE	R	RMSE	R
PhysNet64-2DCNN	10.237	0.928	0.092	0.104	0.247	0.321	0.247	0.321	0.962	0.318
PhysNet64-3DCNN	2.143	0.985	0.067	0.494	0.15	0.749	0.15	0.749	0.647	0.72
PhysNet64-3DCNN-ED	2.048	0.989	0.066	0.501	0.146	0.772	0.146	0.772	0.624	0.748
PhysNet64-LSTM	3.139	0.975	0.084	0.189	0.226	0.478	0.226	0.478	0.928	0.404
PhysNet64-BiLSTM	4.595	0.945	0.085	0.183	0.231	0.421	0.231	0.421	0.956	0.396
PhysNet64-ConvLSTM	2.937	0.977	0.083	0.191	0.22	0.485	0.22	0.485	0.896	0.44

Fig 10. PhysNet Network result

대표 연구 – zithong yu

- PhysFormer
  - rPPG 신호를 재구성 하는 방법 및 3d Vit 적용 방안 제안

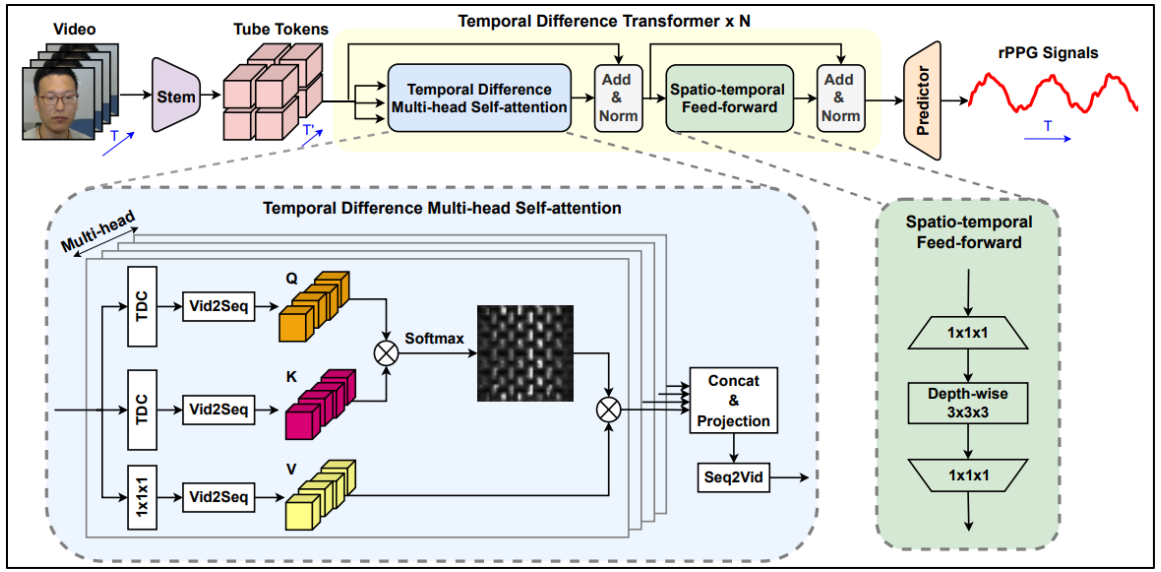


Fig 11. PhysFormer Network

Method	SD ↓ (bpm)	MAE ↓ (bpm)	RMSE ↓ (bpm)	r ↑
Tulyakov2016 [55]▲	18.0	15.9	21.0	0.11
POS [59]▲	15.3	11.5	17.2	0.30
CHROM [13]▲	15.1	11.4	16.9	0.28
RhythmNet [45]♦	8.11	5.30	8.14	0.76
ST-Attention [47]♦	7.99	5.40	7.99	0.66
NAS-HR [40]♦	8.10	5.12	8.01	0.79
CVD [46]♦	7.92	5.02	7.97	0.79
Dual-GAN [41]♦	**7.63**	**4.93**	**7.68**	**0.81**
I3D [7]⋆	15.9	12.0	15.9	0.07
PhysNet [65]⋆	14.9	10.8	14.8	0.20
DeepPhys [11]⋆	13.6	11.0	13.8	0.11
AutoHR [63]⋆	8.48	5.68	8.68	0.72
PhysFormer (Ours)⋆	7.74	4.97	7.79	0.78

Fig 12. PhysFormer Network result

대표 연구 – zithong yu

- RhythmNet
  - Spatial-Temporal Map approach를 처음으로 제안
  - 얼굴의 특징보다 시공간적인 특징에 집중

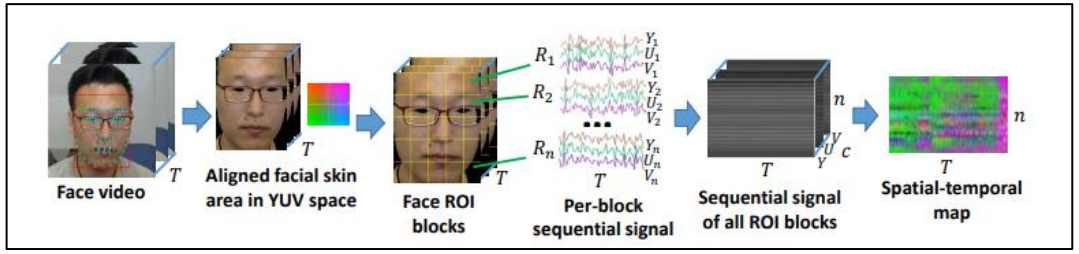


Fig 13. STMAP preprocessing

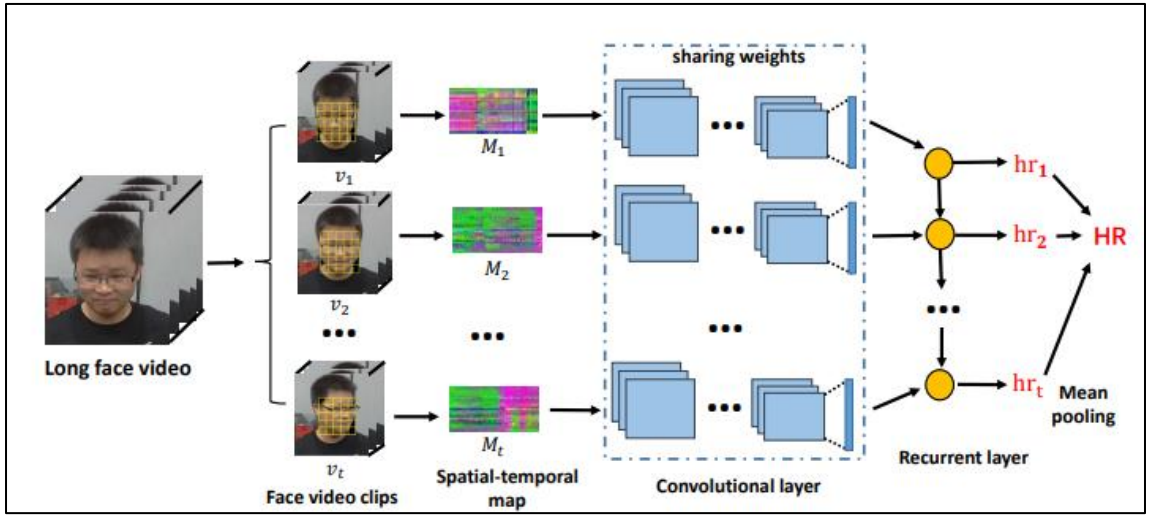


Fig 14. RhythmNet

대표 연구

- RhythmNet

TABLE VII: The HR estimation results by the proposed approach and several state-of-the-art methods on the MAHNOB-HCI database.

Method	Mean (bpm)	Std (bpm)	RMSE (bpm)	MER	r
Poh2010 [1]	-8.95	24.3	25.9	25.0%	0.08
Poh2011 [2]	2.04	13.5	13.6	13.2%	0.36
Balakrishnan2013 [3]	-14.4	15.2	21.0	20.7%	0.11
Li2014 [5]	-3.30	6.88	7.62	6.87%	0.81
Haan2013 [4]	-2.89	13.67	10.7	12.9%	0.82
Tulyakov2016 [6]	3.19	5.81	6.23	5.93%	0.83
RhythmNet(WithinDB)	**0.41**	3.98	4.00	4.18%	0.87
RhythmNet(CrossDB)	-5.66	6.06	8.28	8.00%	0.64
RhythmNet(Fine-tuned)	0.43	**3.97**	**3.99**	**4.06%**	**0.87**

Fig 15. RhythmNet Result

대표 연구

- TransRppg

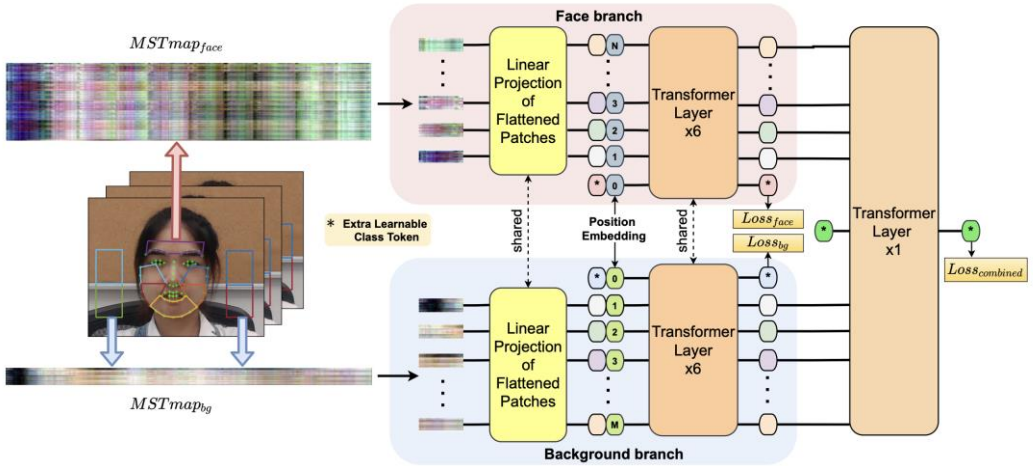


Fig 16. TransRppg

TABLE II: Cross-dataset results on 3DMAD and MARsV2.

Method	3DMAD→MARsV2 AUC(%)↑	3DMAD→MARsV2 FFR@FLR=0.01↓	MARsV2→3DMAD AUC(%)↑	MARsV2→3DMAD FFR@FLR=0.01↓
MS-LBP [17]▲	60.4	100.0	75.3	87.8
CTA [42]▲	62.1	98.3	60.5	96.5
VGG16 [43]▲	54.6	97.9	58.6	99.3
GrPPG [19]⋆	86.7	78.5	87.2	94.5
LrPPG [21]⋆	95.6	61.7	92.3	48.7
CFrPPG [22]⋆	**99.0**	**19.6**	98.0	**12.4**
TransRPPG (Ours)⋆	91.3	47.6	**98.3**	18.5

Fig 17. TransRppg result

이전 연구의 한계

- Methods
  - McDuffs Approach
    - DeepPhys[3] : Dyschromatic Reflection Model(DRM)을 이용한 전처리 기법 제안 (빛을 고려한 모델)
    - High order[4] : LVET의 특성 + DRM을 이용한 변곡점 학습 방안을 제안 (심장박동의 주기를 고려한 모델)

    => 차분 영상을 이용하여 공간 변화의 연속성을 학습하지 않음
  - Zithong Yu Approach
    - PhysNet[5] : 3D Convolution을 이용하여 시공간적인 특징을 학습하는 방법 제안(Deep Learning Module로써 접근)
    - PhysFormer[6] : Tube based Vision Transformer방법 제안(ViT를 사용하는 방법 제안)

    => 생체신호의 특징을 고려하지 않음
  - STMap Approach
    - RhythmNet[7] : STMAP을 이용한 방법 제안(전처리 기법 제안)

    => 얼굴의 특징을 학습하지 않음 / 전처리를 필요로 함

3. DeepPhys: Video-Based Physiological Measurement Using Convolutional Attention Networks https://arxiv.org/abs/1805.07888
4. Learning Higher-Order Dynamics in Video-Based Cardiac Measurement https://arxiv.org/pdf/2110.03690.pdf
5. Remote Photoplethysmograph Signal Measurement from Facial Videos Using Spatio-Temporal Networks : https://arxiv.org/abs/1905.02419
6. PhysFormer: Facial Video-based Physiological Measurement with Temporal Difference Transformer : https://arxiv.org/abs/2111.12082
7. RhythmNet: End-to-end Heart Rate Estimation from Face via Spatial-temporal Representation : https://arxiv.org/abs/1910.11515

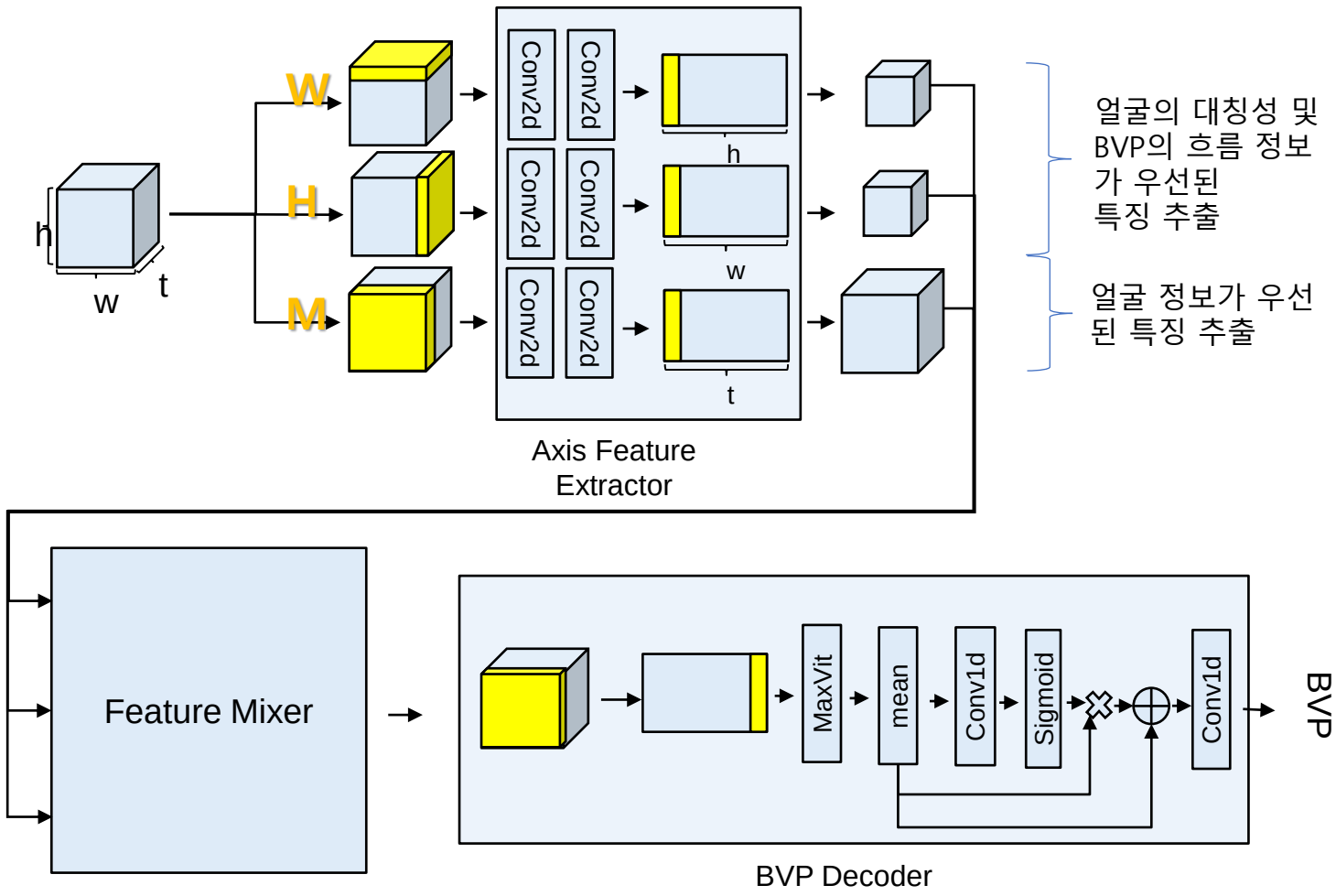
제안 방법

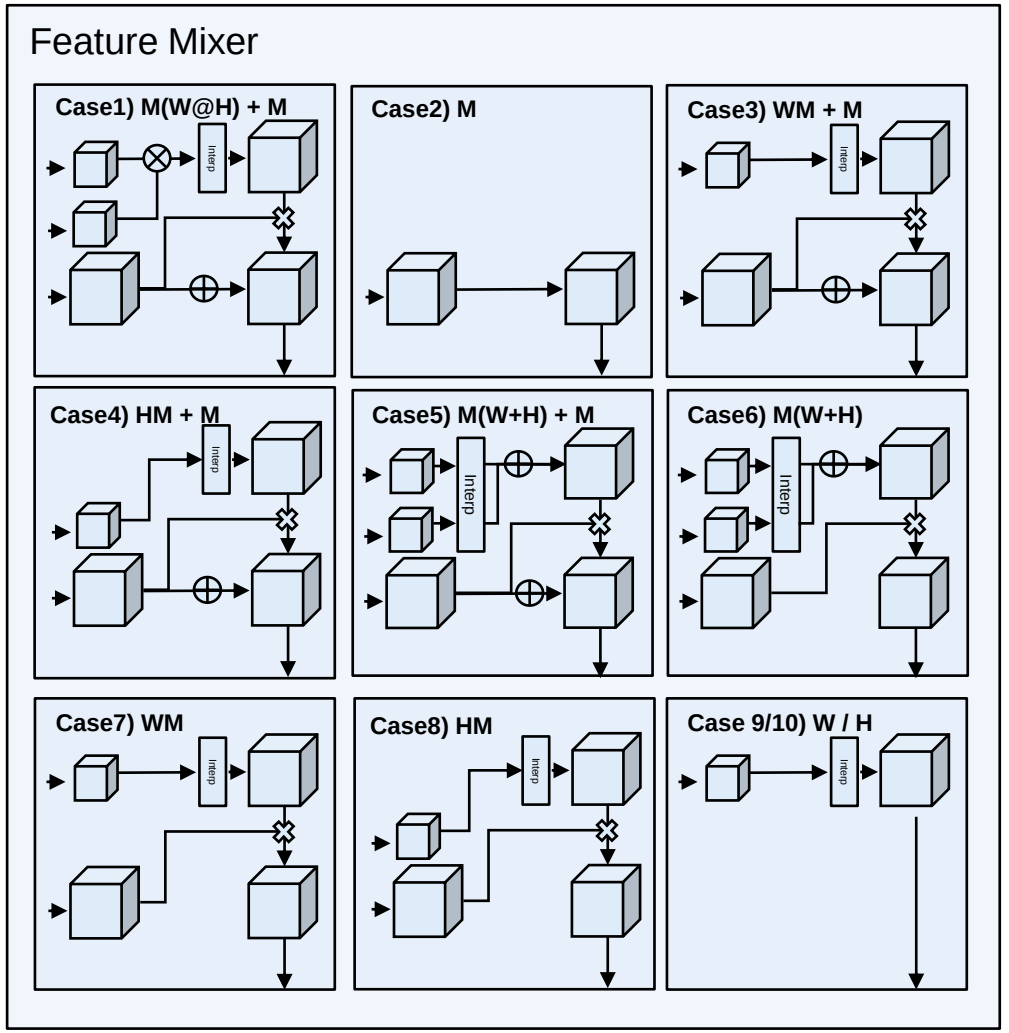
* 딥러닝 기반 rPPG 모델 사용을 위한 경량 모델 연구

- Methods
  - McDuffs Approach
    => 차분 영상을 이용하여 공간 변화의 연속성을 학습하지 않음
  - Zithong Yu Approach
    => 생체신호의 특징을 고려하지 않음
  - STMap Approach
    => 얼굴의 특징을 학습하지 않음 / 전처리를 필요로 함

- 공간변화의 연속성, 생체신호의 특징, 얼굴 특징을 고려하며 전처리를 필요로 하지 않는 E2E 모델 제안

제안방법


W
H
M
h
w
t
Conv2d
Conv2d
Conv2d
Conv2d
Conv2d
Conv2d
h
w
t
Axis Feature Extractor
얼굴의 대칭성 및 BVP의 흐름 정보가 우선된 특징 추출
얼굴 정보가 우선된 특징 추출
Feature Mixer
MaxVit
mean
Conv1d
Sigmoid
Conv1d
BVP Decoder
BVP


Feature Mixer
Case1) M(W@H) + M
Case2) M
Case3) WM + M
Case4) HM + M
Case5) M(W+H) + M
Case6) M(W+H)
Case7) WM
Case8) HM
Case 9/10) W / H
Interp

실험 평가

- 데이터셋
  - V4V
    - 여성 참여자 82명, 남성참여자 58명
    - 웃거나 눈을 찡그리는 등 10가 지 테스크 수행
    - 1000kHz로 bvp, 호흡, 심박 생체 징후 측정

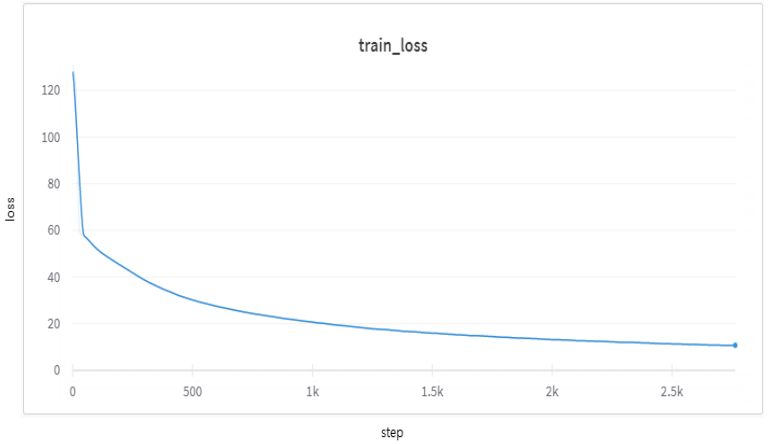


Fig 6. V4V train loss

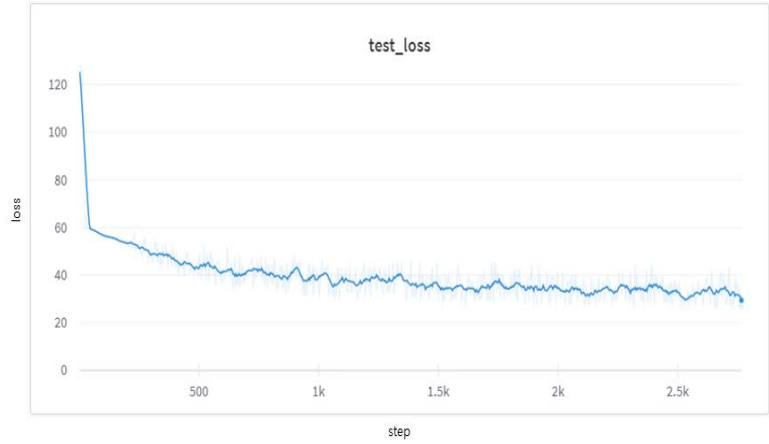


Fig 7. V4V test loss

실험 결과

- 실험평가

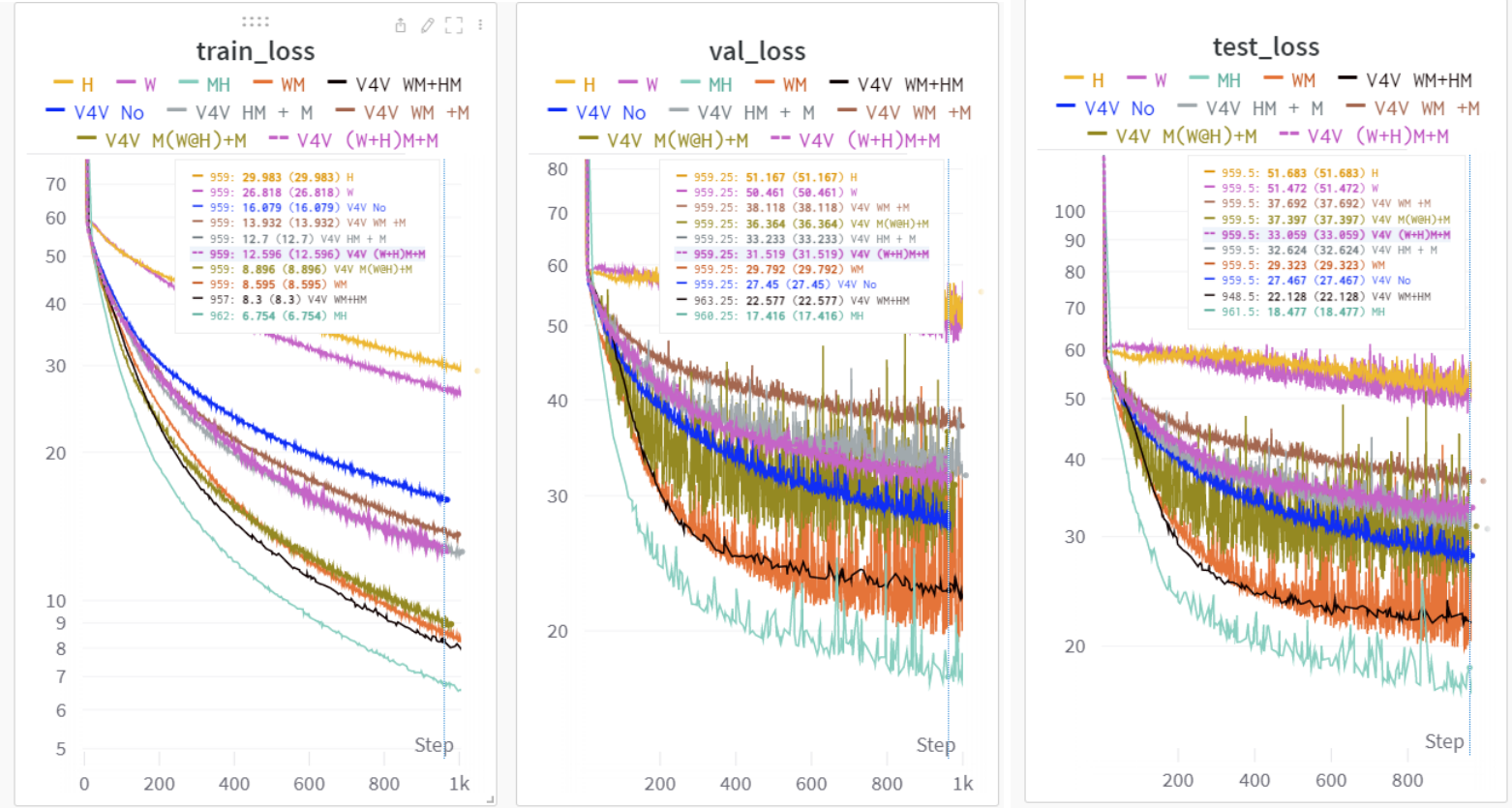


Fig 15. Loss Graph

실험 결과

- 실험평가
  - V4V challenge 참가 7개 비교군과 비교
  - Feature Mixer 별로 추가 비교
  - HR-MAE, HR-RMSE, BVP Pearson Correlation
  - Activation map 평가

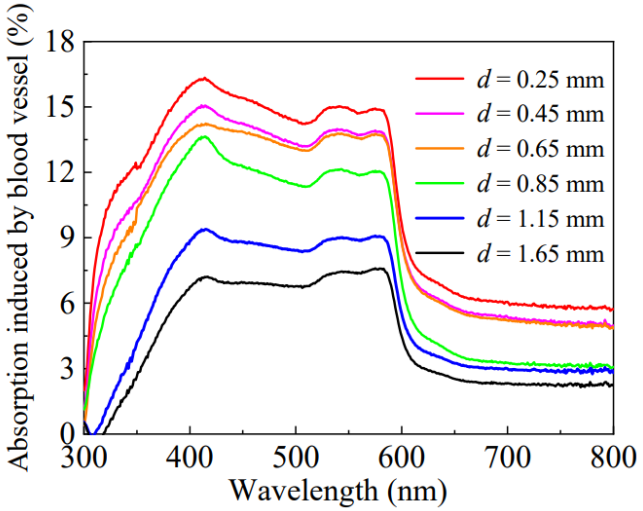


Fig 16. 피부 두께에 따른 빛 흡수 량

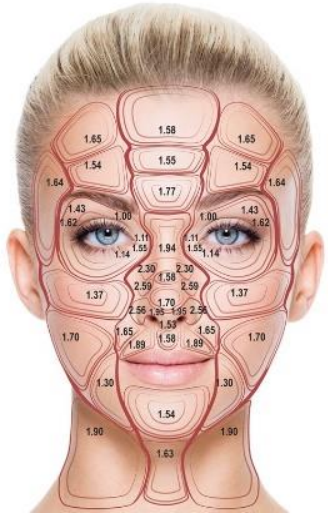


Fig 17. 해부학적으로 구분된 39개 영역에 대한 피부 두께

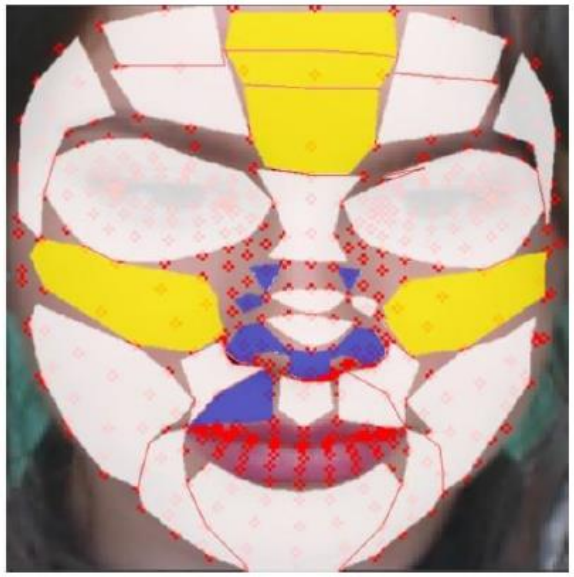


Fig 18. 피부 두께에 따른 BVP 추출이 우수한 영역

실험 결과

- 실험평가

Table 1. Experiment result

Method	HR-MAE	HR-RMSE	Correlation(r)
Brain et.al[67]	9.42	14.6	0.436
Benjamin et.al[68]	8.32	10.95	0.336
Stent et.al[69]	9.22	14.18	0.47
Hill et.al[70]	9.37	14.59	0.44
Ouzar et.al[71]	11.60	14.90	0.2
Green	15.45	20.73	0.05
Proposed	**7.66**	**10.08**	**0.69**

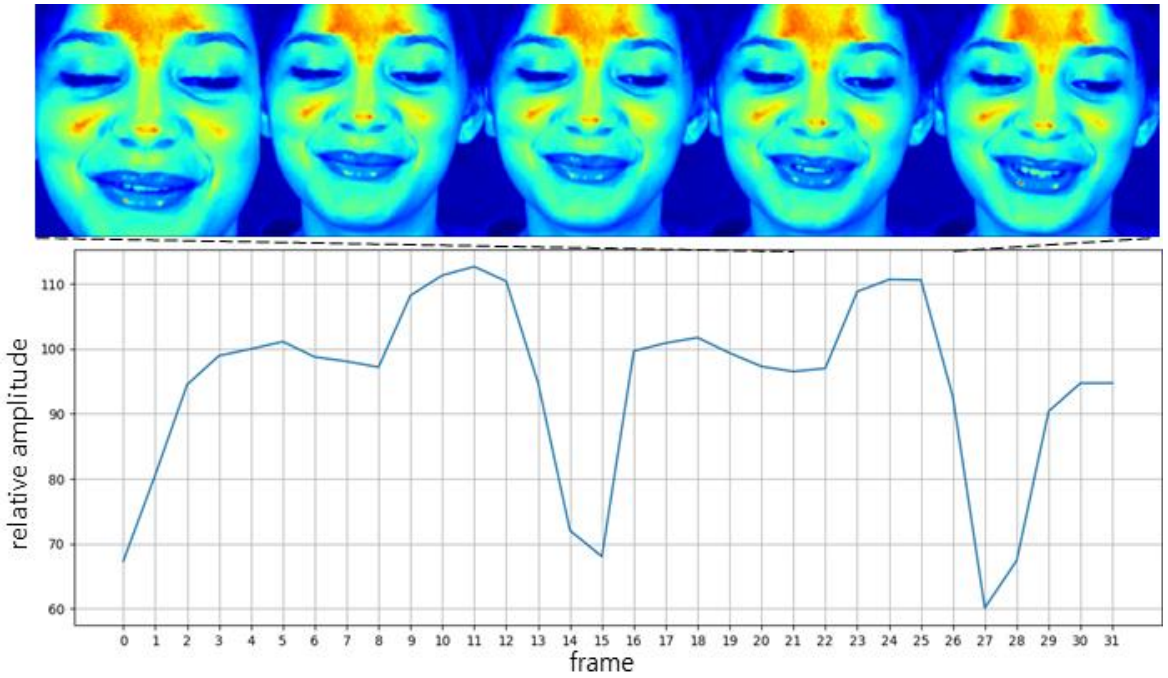


Fig 19. Regression Activation Map

논문을 위한 소스코드 작성법

- 딥 러닝 논문은 이론적인 접근도 중요하지만, 다양한 모델에 비해 얼마나 성능이 좋아졌는지도 중요함.

- 또, 논문 작성을 위해 실험할 때는 전처리 조건, 하이퍼파라미터 조건등 다양한 조건을 바꿔가면서 실험을 진행해야함.

- 딥 러닝 논문은 재사용성이 중요
  - 하이퍼파라미터 설정이 중요한 요인

main.py

```
bpm_flag = False
K_Fold_flag = False
model_save_flag = False
log_flag = False
wandb_flag = False
random_seed = 0
save_img_flag = False
```

```
# for Reproducible model
torch.manual_seed(random_seed)
torch.cuda.manual_seed(random_seed)
torch.cuda.manual_seed_all(random_seed)  # if use multi-GPU
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
np.random.seed(random_seed)
random.seed(random_seed)
```

- torch/tensorflow는 init 때마다 랜덤한 값으로 초기화하며, 매 실험마다 결과가 달라짐
- Weight initialize 뿐만아니라 torch.Tensor.index_add(), torch.Tensor.scatter_add_() 등.. 여러 함수가 random하고 nondeterministic 하게 동작하여 고정 값이 필요

main.py

```
with open('params.json') as f:
    jsonObject = json.load(f)
    __PREPROCESSING__ = jsonObject.get("__PREPROCESSING__")
    __TIME__ = jsonObject.get("__TIME__")
    __MODEL_SUMMARY__ = jsonObject.get("__MODEL_SUMMARY__")
    options = jsonObject.get("options")
    params = jsonObject.get("params")
    preprocessing_prams = jsonObject.get("preprocessing_params")
    hyper_params = jsonObject.get("hyper_params")
    model_params = jsonObject.get("model_params")
    wandb_params = jsonObject.get("wandb")
```

- 실험 때마다 전처리 여부/ 학습 소요시간/ 데이터셋의 종류/ 전처리 방법 등 정해야할 것이 많아 params로 관리

Params.json

```
{
    "__TIME__" : 0,
    "__PREPROCESSING__" : 1,
    "preprocessing_params": {
        "dataset_name": "V4V",
        "dataset_comment" :
                [
                    "V4V",
                    "PURE",
                    "UBFC",
                    "cuff_less_blood_pressure",
                    "VIPL_HR"
                ],
        "face_detect_algorithm" : 1,
        "face_detect_algorithm_comment" :
                [
                    "1: face_recognition_algorithm",
                    "2: FaceMeshDetector"
                ],
        "divide_flag": 1,
        "divide_flag_comment":
                [
                    "0: divide by subject",
                    "1: divide by number"
                ],
        "fixed_position": 1,
        "fixed_position_comment":
                [
                    "0: face Tracking",
                    "1: fixed Position"
                ],
        "time_length": 32,
        "image_size": 128
    },
    "__MODEL_SUMMARY__" : 0,
    "options":{
        "parallel_criterion" : 1,
        "parallel_criterion_comment" : "TODO need to verification",
        "set_gpu_device" : "6"
    },
```

```
    "params":
    {
        "model_root_path": "/media/hdd1/dy/model/",
        "save_root_path": "/media/hdd1/dy/dataset/",
        "data_root_path": "/media/hdd1/",
        "checkpoint_path" : "/media/hdd1/dy/checkpoint/",
        "train_ratio": 0.7,
        "train_ratio_comment" : "generate train dataset using train_ratio",
        "validation_ratio": 0.9,
        "validation_ratio_comment" : "split train dataset using validation_ratio",
        "train_batch_size" : 200,
        "train_batch_size_comment" :
                [
                    "PhysNet_LSTM : 8",
                    "PPNET :  100"
                ],
        "train_shuffle" : 1,
        "test_batch_size" : 200,
        "test_shuffle" : 0
    },
    "hyper_params":
    {
        "loss_fn": "neg_pearson",
        "loss_fn_comment":
                [
                    "mse","L1","neg_pearson","multi_margin","bce","huber","cosine_embedding",
                    "cross_entropy","ctc","bce_with_logits","gaussian_nll","hinge_embedding",
                    "KLDiv","margin_ranking","multi_label_margin","multi_label_soft_margin",
                    "nll","nll2d","pairwise","poisson_nll","smooth_l1","soft_margin",
                    "triplet_margin","triplet_margin_distance",
                    "PPNET : MSE"
                ],
        "optimizer": "ada_mw",
        "optimizer_comment":
                [
                    "adam","sgd","rms_prop","ada_delta","ada_grad","ada_max",
                    "ada_mw","a_sgd","lbfgs","n_adam","r_adam","rprop","sparse_adam",
                    "PPNET : adam"
                ],
        "learning_rate": 0.0001,
        "learning_rate_comment": [
            "DeepPhys : lr = 1",
            "PhysNet : lr = 0.001",
            "PPNet : lr = 0.001"
        ],

        "epochs" : 30000,
        "epochs_comment" :
                [
                    "PPNET : 100"
                ]
    },
```

Main.py

```
if __PREPROCESSING__:
    if __TIME__:
        start_time = time.time()

    preprocessing(save_root_path=params["save_root_path"],
                  model_name=model_params["name"],
                  data_root_path=params["data_root_path"],
                  dataset_name=wandb_params["dataset_name"],
                  train_ratio=params["train_ratio"],
                  face_detect_algorithm=preprocessing_prams["face_detect_algorithm"],
                  divide_flag=preprocessing_prams["divide_flag"],
                  fixed_position=preprocessing_prams["fixed_position"],
                  time_length=preprocessing_prams["time_length"],
                  imgae_size=preprocessing_prams["image_size"],
                  log_flag=log_flag)
```

- 전처리 때마다 고려해야하는 요소들을 param으로 관리하여 전처리 과정을 변경하였을 때 편하게 이용하도록 작성

dataset_preprocess.py

```
# multiprocessing
    if not split_flag:
        for index, data_path in enumerate(data_list):
            proc = multiprocessing.Process(target=preprocess_Dataset,
                                           args=(dataset_root_path + "/" + data_path, vid_name,
ground_truth_name,
                                                 face_detect_algorithm, divide_flag, fixed_position,
time_length, model_name, img_size, return_dict))
            process.append(proc)
            proc.start()

        for proc in process:
            proc.join()
```

- 데이터셋 전처리 과정은 데이터의 종류에 따라 억겁의 시간이 소요될 수 있음으로 멀티 프로세싱으로 처리

Dataset_preprocess.py

```
train_file_path = save_root_path + model_name + "_" + dataset_name + "_train.hdf5"
test_file_path = save_root_path + model_name + "_" + dataset_name + "_test.hdf5"
dt = h5py.special_dtype(vlen=np.float32)
train_file = h5py.File(save_root_path + model_name + "_" + dataset_name + "_train.hdf5", "w")


if model_name in ["DeepPhys", "PhysNet", "PhysNet_LSTM"]:

        for index, data_path in enumerate(return_dict.keys()[:train]):
            dset = train_file.create_group(data_path)
            dset['preprocessed_video'] = return_dict[data_path]['preprocessed_video']
            dset['preprocessed_label'] = return_dict[data_path]['preprocessed_label']
            dset['preprocessed_hr'] = return_dict[data_path]['preprocessed_hr']

        train_file.close()

        test_file = h5py.File(save_root_path + model_name + "_" + dataset_name + "_test.hdf5", "w")
        for index, data_path in enumerate(return_dict.keys()[train:]):
            dset = test_file.create_group(data_path)
            dset['preprocessed_video'] = return_dict[data_path]['preprocessed_video']
            dset['preprocessed_label'] = return_dict[data_path]['preprocessed_label']
            dset['preprocessed_hr'] = return_dict[data_path]['preprocessed_hr']
        test_file.close()
```

- 전처리된 데이터는 필요에 따라 다른 방식으로 이용 될 수 있음으로, h5py구조로 계층적 관리

models.py

```python
def get_model(model_name: str = 'DeepPhys', log_flag:bool = True):
    """
    :param model_name: model name
    :return: model
    """
    if log_flag:
        print("========= set model get_model() in"+ os.path.basename(__file__))

    if model_name == "DeepPhys":
        return DeepPhys()
    elif model_name == "DeepPhys_DA":
        return DeepPhys_DA()
    elif model_name == "PhysNet":
        return PhysNet()
    elif model_name == "PhysNet_LSTM":
        return PhysNet_2DCNN_LSTM()
    elif model_name == "PPNet":
        return PPNet()
    elif model_name == "GCN":
        return TEST()#Seq_GCN()#TEST()#
    elif model_name == "AxisNet":
        return AxisNet(),PhysiologicalGenerator()
    else:
        log_warning("use implemented model")
        raise NotImplementedError("implement a custom model(%s) in /nets/models/" % model_name)
```

- 모델을 불러올때도, 간단히 param만 변경해서 동작 가능하도록 함수 작성

main.py

```
for epoch in range(hyper_params["epochs"]):
    train_fn(epoch, model, optimizer, criterion, data_loaders[0], "Train",wandb_flag)
    if data_loaders.__len__() == 3:
        _ = test_fn(epoch, model, criterion, data_loaders[1], "Val", wandb_flag, save_img_flag )
    if epoch % 10 == 0:
        running_loss = test_fn(epoch, model, criterion, data_loaders[-1], "Test", wandb_flag,
save_img_flag )
```

- 모델 학습시에는 데이터를 train/val/test 로 구성했는지, train/test 로 구성했는지에 따라 알아서 동작하도록 차이를 둠

요약

- Reproducible 한 코드를 만들기위해 randomness를 제어 하게 코드 작성

- 논문에서 본인이 생각했을 때, 논문 결과의 차이를 줄 만한 요소들 그리고 논문 실험시 편의성을 줄만한 요소들을 params.json 에서 컨트롤
  - Ex) learning rate, batch size, preprocessing 유무, log 유무, 데이터셋 분할 비율 등.,

실습

- 대표적인 rPPG 알고리즘인 physnet을 간단히 학습하는 과정 진행

1. test.py 생성 후 reproducible 코드 생성
2. PhysNet_UBFC_train_9.hdf5 를 이용하는 test_param.json 생성
3. Test.py에서 test_param 파싱
4. Test.py 에서 모델 호출 후 학습 진행

- Randomness 에 따른 변화를 보기위해
  Reproducible code가 있는 환경에서 3번 실험
  Reproducible code가 없는 환경에서 3번 실험

- Loss의 변화 관측

실습

1. test.py 생성 후 reproducible 코드 생성

```
import random

import numpy as np
import torch

random_seed = 0

# for Reproducible model
torch.manual_seed(random_seed)
torch.cuda.manual_seed(random_seed)
torch.cuda.manual_seed_all(random_seed)  # if use multi-GPU
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False
np.random.seed(random_seed)
random.seed(random_seed)
```

실습

2. PhysNet_UBFC_train_9.hdf5 를 이용하는 test_param.json 생성

```
{
  "data_path" : "본인 데이터 경로",
  "train_ratio" : 0.7,
  "validation_ratio" : 0.1,
  "batch_size" : 100,
  "train_shuffle": 1,
  "test_shuffle" : 0,
  "loss_fn" : "neg_pearson",
  "optimizer" : "adam",
  "learning_rate" : 0.0001,
  "epochs" : 31,
  "model" : "physnet"
}
```

실습

3. Test.py에서 test_param 파싱

```
with open(test_param.json') as f:
    jsonObject = json.load(f)
    data_path = jsonObject.get("data_path")
    train_ratio = jsonObject.get("train_ratio")
    val_ratio = jsonObject.get("val_ratio")
    batch_size = jsonObject.get("batch_size")
    train_shuffle = jsonObject.get("train_shuffle")
    test_shuffle = jsonObject.get("test_shuffle")
    loss_fn = jsonObject.get("loss_fn")
    optimizer = jsonObject.get("optimizer")
    learning_rate = jsonObject.get("learning_rate")
    epochs = jsonObject.get("epochs")
    model = jsonObject.get("model")
```

실습

4. Test.py에 get_model 함수 생성 후 호출

```
from colorama import Fore, Style

from nets.models.PhysNet import PhysNet

def log_warning(message):
    print(Fore.LIGHTRED_EX + Style.BRIGHT + message + Style.RESET_ALL)

def get_model(model):
    if model == "physnet":
        return PhysNet()
    else:
        log_warning("use implemented model")
        raise NotImplementedError("implement a custom model(%s) in /nets/models/" % model)

model = get_model(model).to(“cuda”)
```

실습

5. Test.py에 dataset_loader 함수 생성 후 호출

```
from dataset.PhysNetDataset import PhysNetDataset

def dataset_loader(path):
    dataset_file = h5py.File(path,"r")
    video_data = []
    label_data = []

    for key in dataset_file.keys():
        if len(dataset_file[key]['preprocessed_video']) == len(dataset_file[key]['preprocessed_label']):
            video_data.extend(dataset_file[key]['preprocessed_video'])
            label_data.extend(dataset_file[key]['preprocessed_label'])

    dataset_file.close()

    dataset = PhysNetDataset(video_data=np.asarray(video_data),
                                label_data=np.asarray(label_data))
    return dataset

dataset = dataset_loader(data_path)
```

실습

6. Test.py에서 dataset 분리 진행

```
from torch.utils.data import random_split

def dataset_split(dataset, ratio):
    dataset_len = len(dataset)
    if ratio.__len__() == 3:
        train_len = int(np.floor(dataset_len * ratio[0]))
        val_len = int(np.floor(dataset_len * ratio[1]))
        test_len = dataset_len - train_len - val_len

        return random_split(dataset, [train_len, val_len, test_len])
    elif ratio.__len__() == 2:
        train_len = int(np.floor(dataset_len * ratio[0]))
        test_len = dataset_len - train_len
        return random_split(dataset, [train_len, test_len])

datasets = dataset_split(dataset, [0.7, 0.1, 0.2])
```

실습

7. Test.py에서 분리된 데이터를 호출하는 데이터 로더 생성

```
from torch.utils.data import DataLoader

def split_data_loader(datasets, batch_size, train_shuffle, test_shuffle=False):
    if datasets.__len__() == 3:
        train_loader = DataLoader(datasets[0], batch_size=batch_size, shuffle=train_shuffle)
        validation_loader = DataLoader(datasets[1], batch_size=batch_size, shuffle=test_shuffle)
        test_loader = DataLoader(datasets[2], batch_size=batch_size, shuffle=test_shuffle)
        return [train_loader, validation_loader, test_loader]
    elif datasets.__len__() == 2:
        train_loader = DataLoader(datasets[0], batch_size=batch_size, shuffle=train_shuffle)
        test_loader = DataLoader(datasets[1], batch_size=batch_size, shuffle=test_shuffle)
        return [train_loader, test_loader]

data_loaders = split_data_loader(datasets, batch_size, train_shuffle,test_shuffle)
```

실습

8. Test.py에서 loss function 생성

```
def neg_Pearson_Loss(predictions, targets):
    '''
    rst = 0
    targets = targets[:, :]
    predictions = torch.squeeze(predictions)
    # Pearson correlation can be performed on the premise of normalization of input data
    predictions = (predictions - torch.mean(predictions)) / torch.std(predictions)
    targets = (targets - torch.mean(targets)) / torch.std(targets)

    for i in range(predictions.shape[0]):
        sum_x = torch.sum(predictions[i])  # x
        sum_y = torch.sum(targets[i])  # y
        sum_xy = torch.sum(predictions[i] * targets[i])  # xy
        sum_x2 = torch.sum(torch.pow(predictions[i], 2))  # x^2
        sum_y2 = torch.sum(torch.pow(targets[i], 2))  # y^2
        N = predictions.shape[1]
        pearson = (N * sum_xy - sum_x * sum_y) / (
            torch.sqrt((N * sum_x2 - torch.pow(sum_x, 2)) * (N * sum_y2 - torch.pow(sum_y, 2))))

        rst += 1 - pearson

    rst = rst / predictions.shape[0]
    return rst
```

실습

8. Test.py에서 loss function 생성

```
class NegPearsonLoss(nn.Module):
    def __init__(self):
        super(NegPearsonLoss, self).__init__()

    def forward(self, predictions, targets):
        return neg_Pearson_Loss(predictions, targets)

def loss(loss_fn: str = "mse"):
    """
    :param loss_fn: implement loss function for training
    :return: loss function module(class)
    """

    if loss_fn == "neg_pearson":
        return NegPearsonLoss()

criterion = loss(loss_fn)
```

실습

9. Test.py에서 optimizer / scheduler 설정

```
import torch.optim as opt
from torch.optim import lr_scheduler

def optimizer(model_params, learning_rate: float = 1, optim: str = "mse"):


    if optim == "adam":
        return opt.Adam(model_params, learning_rate,weight_decay=0.00005)

optimizer = optimizer(model.parameters(), hyper_params["learning_rate"], hyper_params["optimizer"])
scheduler = lr_scheduler.ExponentialLR(optimizer, gamma=0.99)
```

실습

10. Test.py에서 train 함수 생성

```
from tqdm import tqdm

def train_fn(epoch, model, optimizer, criterion, dataloaders, step:str = "Train " , wandb_flag:bool = True):
    # TODO : Implement multiple loss
    with tqdm(dataloaders,desc= step, total=len(dataloaders)) as tepoch:
        model.train()
        running_loss = 0.0
        for inputs, target in tepoch:
            optimizer.zero_grad()
            tepoch.set_description(step + "%d" % epoch)
            p = model(inputs)
            loss = criterion(p,target)

            if ~torch.isfinite(loss):
                continue
            loss.backward()
            running_loss += loss.item()
            optimizer.step()

            tepoch.set_postfix(loss=running_loss / tepoch.__len__())
```

실습

10. Test.py에서 test_fn 함수 생성

```
def test_fn(epoch, model, criterion, dataloaders, step:str = "Test"):
    with tqdm(dataloaders,desc= step, total=len(dataloaders)) as tepoch:
        model.eval()
        running_loss = 0.0

        inference_array = []
        target_array = []

        with torch.no_grad():
            for inputs, target in tepoch:
                tepoch.set_description(step + "%d" % epoch)
                p = model(inputs)
                loss = criterion(p,target)

                if ~torch.isfinite(loss):
                    continue
                running_loss += loss.item()


                tepoch.set_postfix(loss=running_loss / tepoch.__len__())

        return running_loss
```

실습

10. Test.py에서 test_fn 함수 생성

```
def test_fn(epoch, model, criterion, dataloaders, step:str = "Test"):
    with tqdm(dataloaders,desc= step, total=len(dataloaders)) as tepoch:
        model.eval()
        running_loss = 0.0

        inference_array = []
        target_array = []

        with torch.no_grad():
            for inputs, target in tepoch:
                tepoch.set_description(step + "%d" % epoch)
                p = model(inputs)
                loss = criterion(p,target)

                if ~torch.isfinite(loss):
                    continue
                running_loss += loss.item()


                tepoch.set_postfix(loss=running_loss / tepoch.__len__())

        return running_loss
```

실습

10. Test.py에서 학습진행

```
for epoch in range(epochs):
    train_fn(epoch, model, optimizer, criterion, data_loaders[0], "Train")
    if data_loaders.__len__() == 3:
        _ = test_fn(epoch, model, criterion, data_loaders[1], "Val" )
    if epoch % 10 == 0:
        running_loss = test_fn(epoch, model, criterion, data_loaders[-1], "Test" )
```

reference

- https://news.samsungdisplay.com/30140
- 모바일 환경에서 안면 영상을 이용한 실시간 생체징후 측정 시스템
- DeepPhys: Video-Based Physiological Measurement Using Convolutional Attention Networks https://arxiv.org/abs/1805.07888
- Learning Higher-Order Dynamics in Video-Based Cardiac Measurement https://arxiv.org/pdf/2110.03690.pdf
- Remote Photoplethysmograph Signal Measurement from Facial Videos Using Spatio-Temporal Networks : https://arxiv.org/abs/1905.02419
- PhysFormer: Facial Video-based Physiological Measurement with Temporal Difference Transformer : https://arxiv.org/abs/2111.12082
- RhythmNet: End-to-end Heart Rate Estimation from Face via Spatial-temporal Representation : https://arxiv.org/abs/1910.11515
- Pulse transit time estimation of aortic pulse wave velocity and blood pressure using machine learning and simulated training data https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6711549/#pcbi.1007259.e026
- Phase Velocity of Facial Blood Volume Oscillation at a Frequency of 0.1 Hz https://www.frontiersin.org/articles/10.3389/fphys.2021.627354/full
- Assessment of ROI Selection for Facial Video-Based rPPG
- Kienle A, Lilge L, Vitkin AI, et al. Why do veins appear blue? A new look at an old question. Appl Optics 1996; 35:1151-1160. (Maybe more than you wanted to know, but here it is).