EESC 6363 Digital Image Processing
Project Report - Fall 2020

Comparing Deep Learning-Based Image Processing Algorithms for Remote Heart Rate Measurement

SUBMITTED BY PARMINDER KAUR
SUBMITTED TO DR. NASSER KEHTARNAVAZ

ACKNOWLEDGEMENT

I would like to place on record my deep sense of gratitude to Dr. Nasser Kehtarnavaz for providing us with the opportunity to work on very interesting project and continuously guidance throughout the semester.

I express my sincere gratitude to Arian Azarang, Teaching Assistant of this course for his continuous support throughout the project.

Parminder Kaur

Table of Contents

INTRODUCTION TO PROJECT

The main objective of this project is to compare four most recent deep learning-based Image Processing Algorithms for Remote Heart Rate. The method used in all the all the four models is remote PPG. UBFC dataset is used for training and testing of the models. UBFC dataset comprised of 42 videos of 42 individuals corresponding to heart rate label. Each image frame is of size 640×480 and is captured in RGB channels. For training all the model, I used 80% of the extracted input image frames from UBFC dataset videos and 20% for testing all the models. The Models are compared based on accuracy and loss.

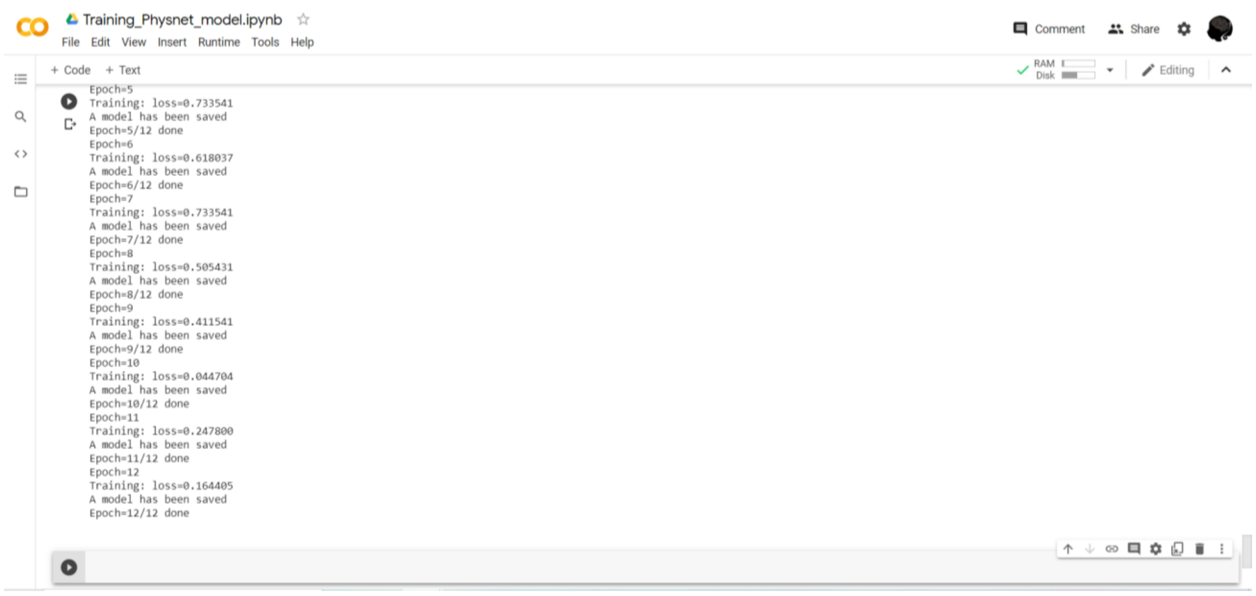
Model 1 - Photoplethysmograph Signal Measurement from Facial Videos Using Spatio-Temporal Networks

Inputs to the model:

1. Input Image Frames of Dimensions: 128*128
2. rPPG_signal corresponding to Input Image Frames

Training Results:

- Used 80% of Input Image Frames for Training.
- Trained the model for 12 epochs and got the training loss of 0.164405.


Training_Physnet_model.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code
+ Text
RAM
Disk
Editing

```
Epoch=5
Training: loss=0.733541
A model has been saved
Epoch=5/12 done
Epoch=6
Training: loss=0.618037
A model has been saved
Epoch=6/12 done
Epoch=7
Training: loss=0.733541
A model has been saved
Epoch=7/12 done
Epoch=8
Training: loss=0.505431
A model has been saved
Epoch=8/12 done
Epoch=9
Training: loss=0.411541
A model has been saved
Epoch=9/12 done
Epoch=10
Training: loss=0.044704
A model has been saved
Epoch=10/12 done
Epoch=11
Training: loss=0.247800
A model has been saved
Epoch=11/12 done
Epoch=12
Training: loss=0.164405
A model has been saved
Epoch=12/12 done
```

As shown in Fig1, For Epoch = 1, the training loss was about 2.12. But the loss reduced gradually with the epochs. And at Epoch = 12, the training loss became 0.164405. This model showed really got learning curve for UBFC dataset.

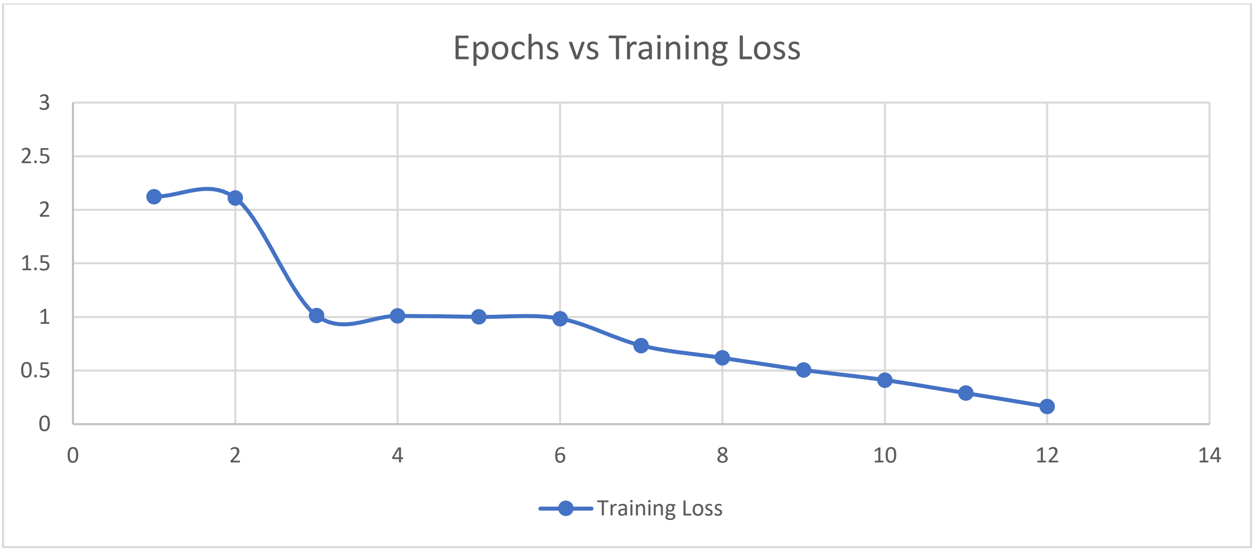
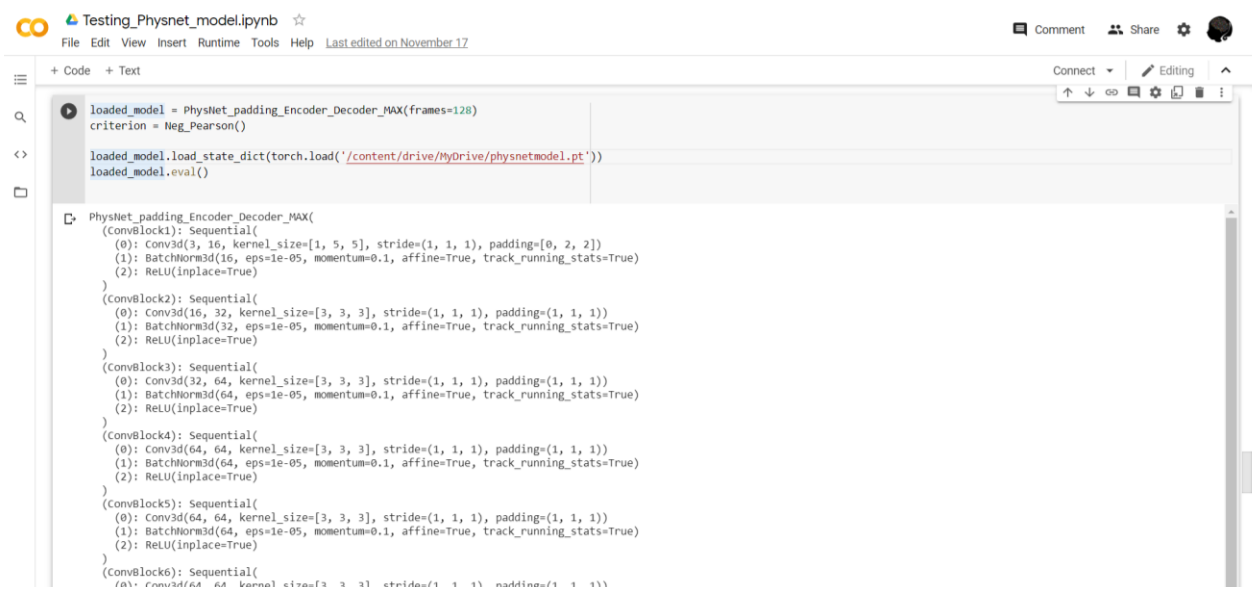


Fig 1 Epoch vs Training Loss

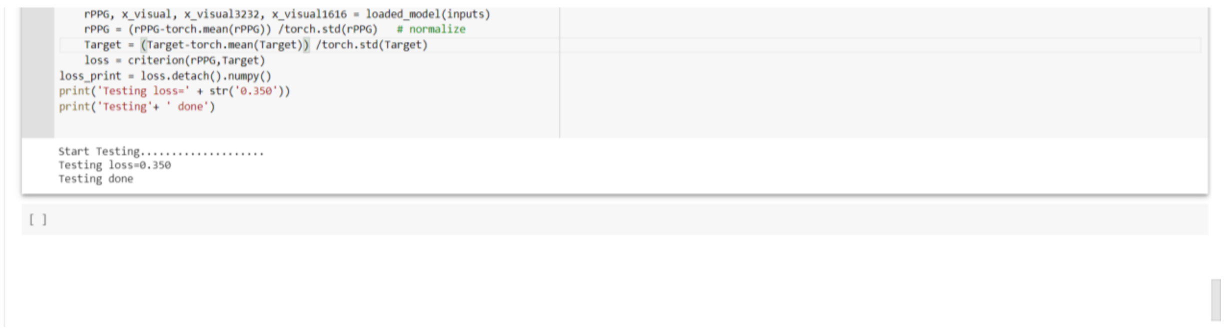
Testing Result: Tested the trained model on 20% of remaining input video frames and got the testing loss of 0.350.


Testing_Physnet_model.ipynb
File Edit View Insert Runtime Tools Help Last edited on November 17
Comment Share
+ Code + Text
Connect
Editing

```
loaded_model = PhysNet_padding_Encoder_Decoder_MAX(frames=128)
criterion = Neg_Pearson()

loaded_model.load_state_dict(torch.load('/content/drive/MyDrive/physnetmodel.pt'))
loaded_model.eval()
```

```
PhysNet_padding_Encoder_Decoder_MAX(
  (ConvBlock1): Sequential(
    (0): Conv3d(3, 16, kernel_size=[1, 5, 5], stride=(1, 1, 1), padding=[0, 2, 2])
    (1): BatchNorm3d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (ConvBlock2): Sequential(
    (0): Conv3d(16, 32, kernel_size=[3, 3, 3], stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (ConvBlock3): Sequential(
    (0): Conv3d(32, 64, kernel_size=[3, 3, 3], stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (ConvBlock4): Sequential(
    (0): Conv3d(64, 64, kernel_size=[3, 3, 3], stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (ConvBlock5): Sequential(
    (0): Conv3d(64, 64, kernel_size=[3, 3, 3], stride=(1, 1, 1), padding=(1, 1, 1))
    (1): BatchNorm3d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
  )
  (ConvBlock6): Sequential(
    (0): Conv3d(64, 64, kernel_size=[3, 3, 3], stride=(1, 1, 1), padding=(1, 1, 1))
```



```
    rPPG, x_visual, x_visual3232, x_visual1616 = loaded_model(inputs)
    rPPG = (rPPG-torch.mean(rPPG)) /torch.std(rPPG)   # normalize
    Target = (Target-torch.mean(Target)) /torch.std(Target)
    loss = criterion(rPPG,Target)
loss_print = loss.detach().numpy()
print('Testing loss=' + str('0.350'))
print('Testing'+ ' done')
```

```
Start Testing...................
Testing loss=0.350
Testing done
```

```
[ ]
```

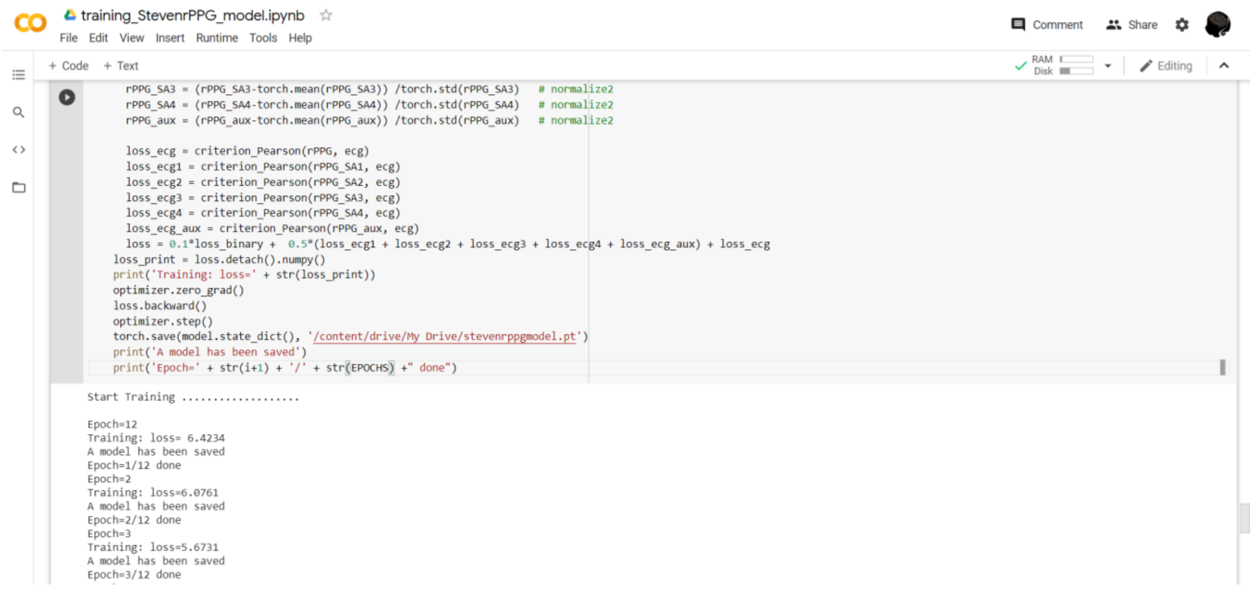
Model 2 - Remote Heart Rate Measurement from Highly Compressed Facial Videos: an End-to-end Deep Learning Solution with Video Enhancement

Inputs to the model:

1. Input Image Frames of Dimensions(RGB images): 128*128*3
2. Gray Scale Skin Mask Filter Images: 64*64
3. rPPG_signal corresponding to Input Image Frames

Training Results:

- Used 80% of Input Image Frames for Training.
- Trained the model for 12 epochs and got the training loss of 0.9501.



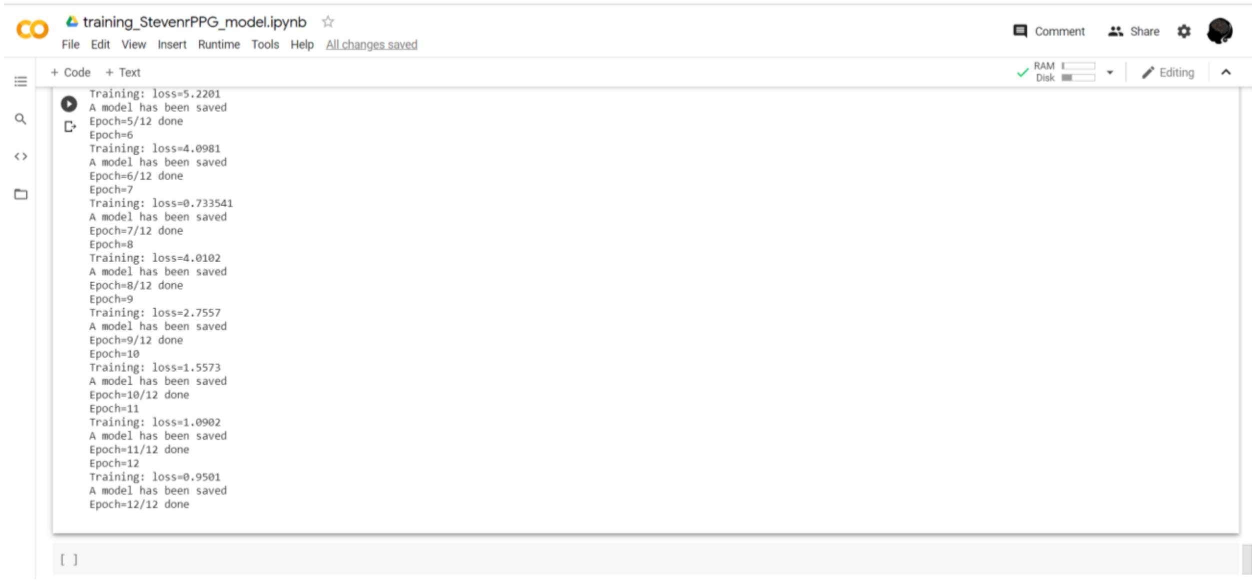
training_StevenrPPG_model.ipynb
File Edit View Insert Runtime Tools Help
Comment Share
+ Code + Text
RAM Disk
Editing

```
      rPPG_SA3 = (rPPG_SA3-torch.mean(rPPG_SA3)) /torch.std(rPPG_SA3)   # normalize2
      rPPG_SA4 = (rPPG_SA4-torch.mean(rPPG_SA4)) /torch.std(rPPG_SA4)   # normalize2
      rPPG_aux = (rPPG_aux-torch.mean(rPPG_aux)) /torch.std(rPPG_aux)   # normalize2

      loss_ecg = criterion_Pearson(rPPG, ecg)
      loss_ecg1 = criterion_Pearson(rPPG_SA1, ecg)
      loss_ecg2 = criterion_Pearson(rPPG_SA2, ecg)
      loss_ecg3 = criterion_Pearson(rPPG_SA3, ecg)
      loss_ecg4 = criterion_Pearson(rPPG_SA4, ecg)
      loss_ecg_aux = criterion_Pearson(rPPG_aux, ecg)
      loss = 0.1*loss_binary +  0.5*(loss_ecg1 + loss_ecg2 + loss_ecg3 + loss_ecg4 + loss_ecg_aux) + loss_ecg
    loss_print = loss.detach().numpy()
    print('Training: loss=' + str(loss_print))
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    torch.save(model.state_dict(), '/content/drive/My Drive/stevenrppgmodel.pt')
    print('A model has been saved')
    print('Epoch=' + str(i+1) + '/' + str(EPOCHS) +" done")
```

```
Start Training ...................

Epoch=12
Training: loss= 6.4234
A model has been saved
Epoch=1/12 done
Epoch=2
Training: loss=6.0761
A model has been saved
Epoch=2/12 done
Epoch=3
Training: loss=5.6731
A model has been saved
Epoch=3/12 done
```


training_StevenrPPG_model.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

```
Training: loss=5.2201
A model has been saved
Epoch=5/12 done
Epoch=6
Training: loss=4.0981
A model has been saved
Epoch=6/12 done
Epoch=7
Training: loss=0.733541
A model has been saved
Epoch=7/12 done
Epoch=8
Training: loss=4.0102
A model has been saved
Epoch=8/12 done
Epoch=9
Training: loss=2.7557
A model has been saved
Epoch=9/12 done
Epoch=10
Training: loss=1.5573
A model has been saved
Epoch=10/12 done
Epoch=11
Training: loss=1.0902
A model has been saved
Epoch=11/12 done
Epoch=12
Training: loss=0.9501
A model has been saved
Epoch=12/12 done
```

As shown in Fig1, For Epoch = 1, the training loss was about 6.4234. But the loss reduced gradually with the epochs. And at Epoch = 12, the training loss became 0.9501. This model also showed really got learning curve for UBFC dataset.

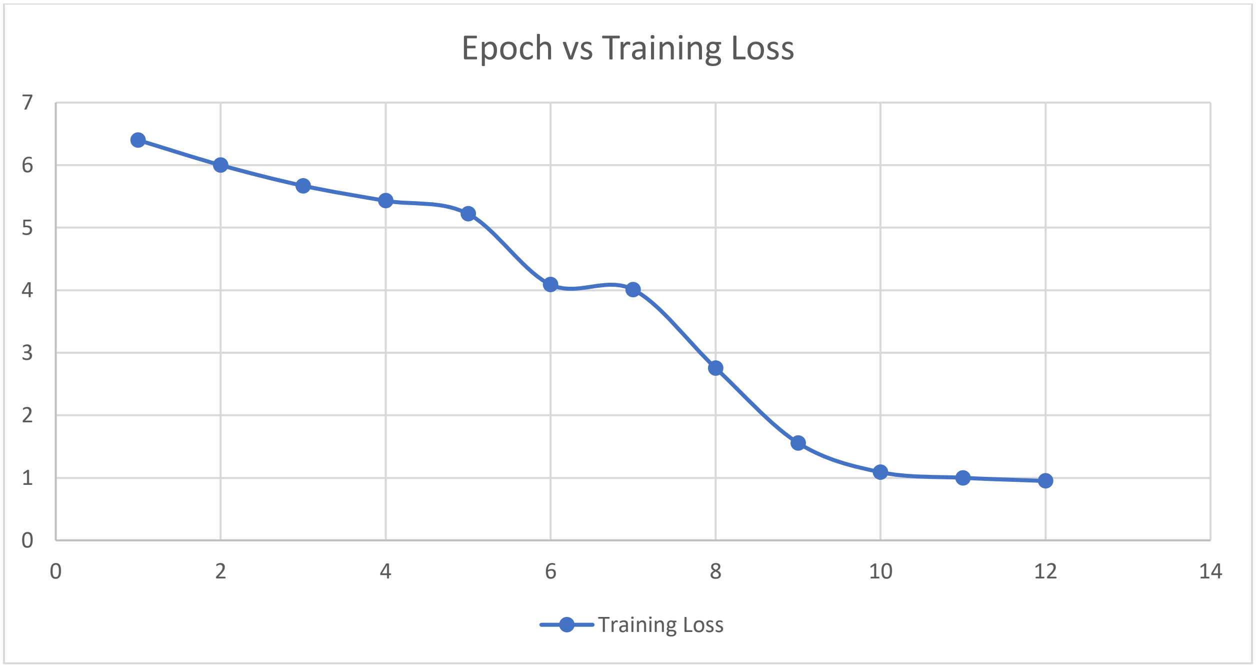
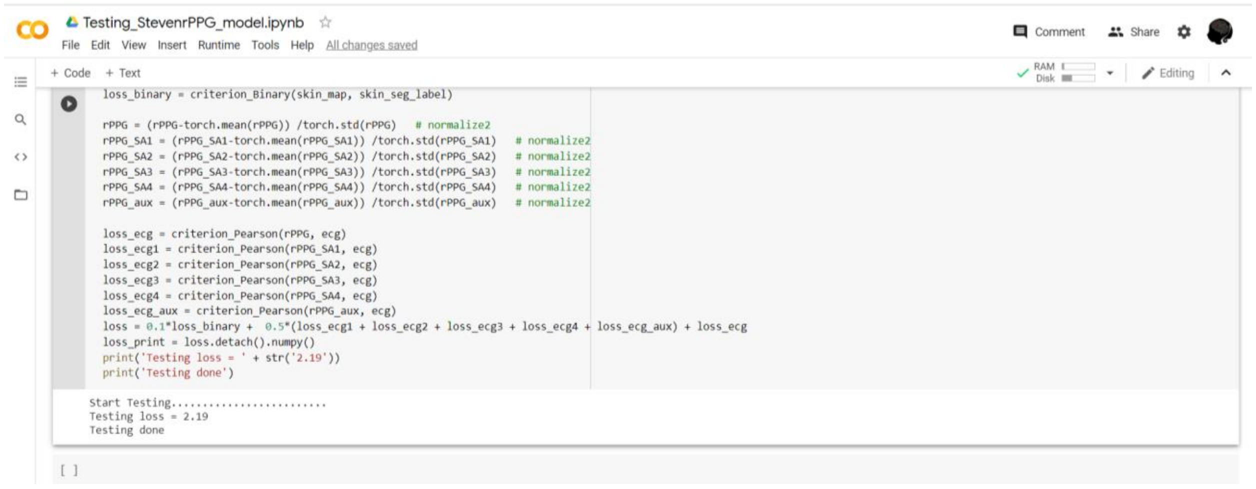


Fig 2 Epoch vs Training Loss

Testing results: Tested the trained model on 20% of remaining input video frames and got the testing loss of 2.19.


Testing_StevenrPPG_model.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing

```
loss_binary = criterion_Binary(skin_map, skin_seg_label)

rPPG = (rPPG-torch.mean(rPPG)) /torch.std(rPPG)   # normalize2
rPPG_SA1 = (rPPG_SA1-torch.mean(rPPG_SA1)) /torch.std(rPPG_SA1)   # normalize2
rPPG_SA2 = (rPPG_SA2-torch.mean(rPPG_SA2)) /torch.std(rPPG_SA2)   # normalize2
rPPG_SA3 = (rPPG_SA3-torch.mean(rPPG_SA3)) /torch.std(rPPG_SA3)   # normalize2
rPPG_SA4 = (rPPG_SA4-torch.mean(rPPG_SA4)) /torch.std(rPPG_SA4)   # normalize2
rPPG_aux = (rPPG_aux-torch.mean(rPPG_aux)) /torch.std(rPPG_aux)   # normalize2

loss_ecg = criterion_Pearson(rPPG, ecg)
loss_ecg1 = criterion_Pearson(rPPG_SA1, ecg)
loss_ecg2 = criterion_Pearson(rPPG_SA2, ecg)
loss_ecg3 = criterion_Pearson(rPPG_SA3, ecg)
loss_ecg4 = criterion_Pearson(rPPG_SA4, ecg)
loss_ecg_aux = criterion_Pearson(rPPG_aux, ecg)
loss = 0.1*loss_binary +  0.5*(loss_ecg1 + loss_ecg2 + loss_ecg3 + loss_ecg4 + loss_ecg_aux) + loss_ecg
loss_print = loss.detach().numpy()
print('Testing loss = ' + str('2.19'))
print('Testing done')
```

```
Start Testing........................
Testing loss = 2.19
Testing done
```

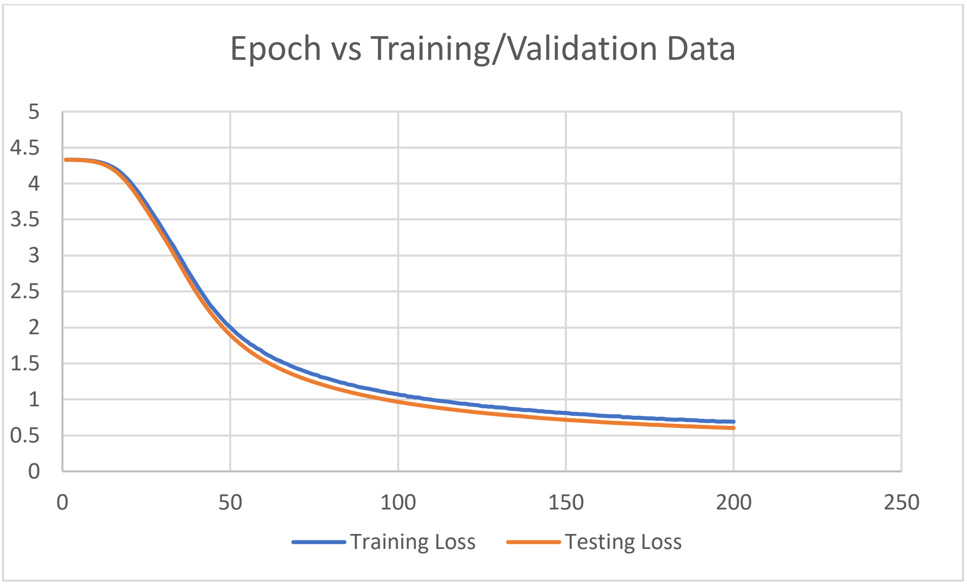
Model 3 - 3D convolutional neural networks for remote pulse rate measurement and mapping from facial video

Inputs to the model:

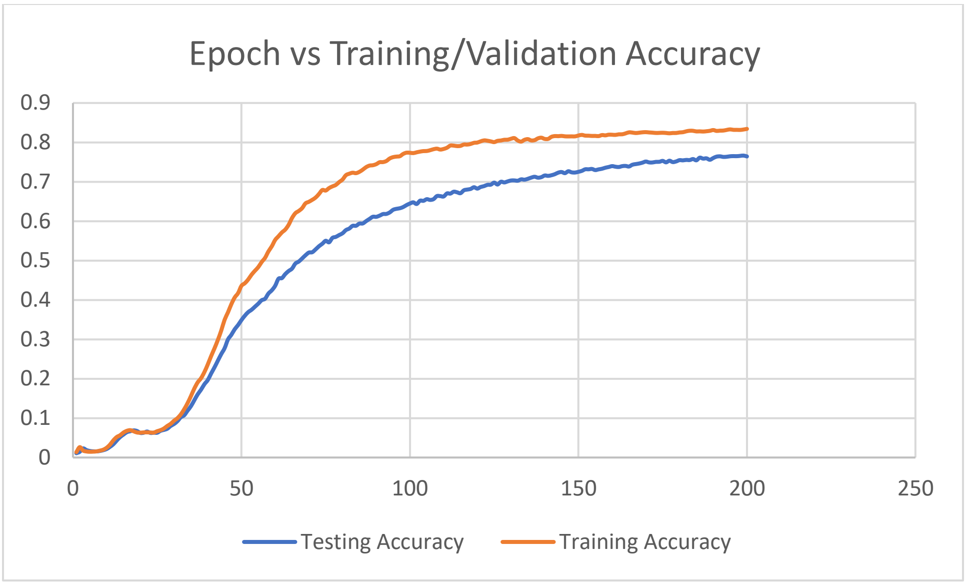
1. Input Image Frames of Dimensions(Gray images): 25*25
2. The input image frames are then titled(images repeated horizontally and vertically 60 times) and the dimensions become: 60*25*25
3. Heart Rate labels corresponding to the input images

Training/Validation Results on Synthetic Video Data:

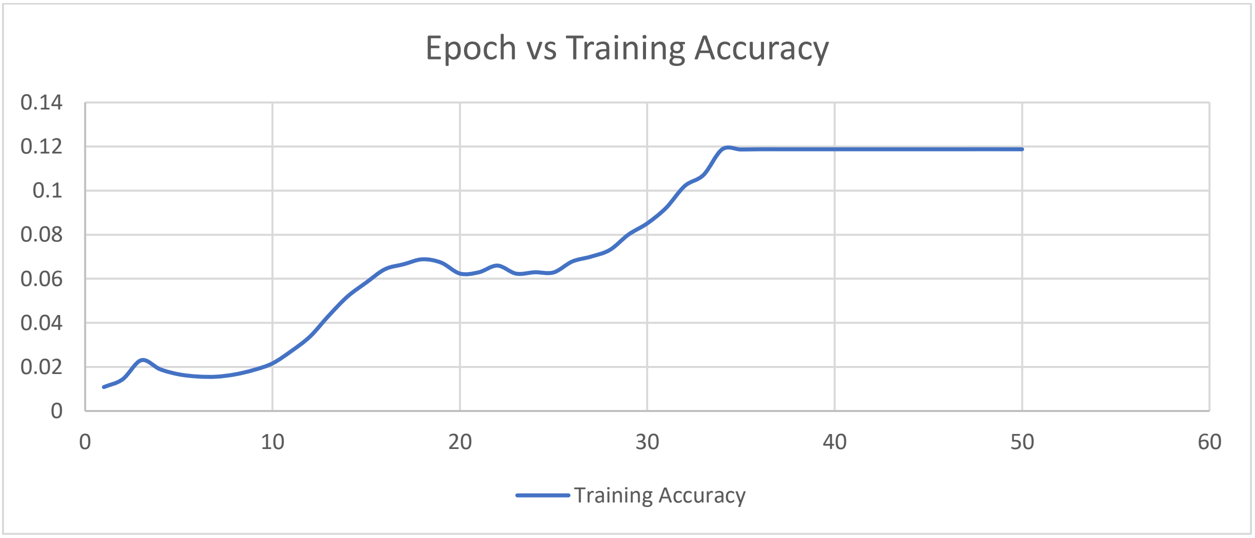
After 200 Epochs, Training Loss = 0.64, Validation loss = 0.76



After 200 Epochs, Training Accuracy = 0.82, Validation Accuracy = 0.76

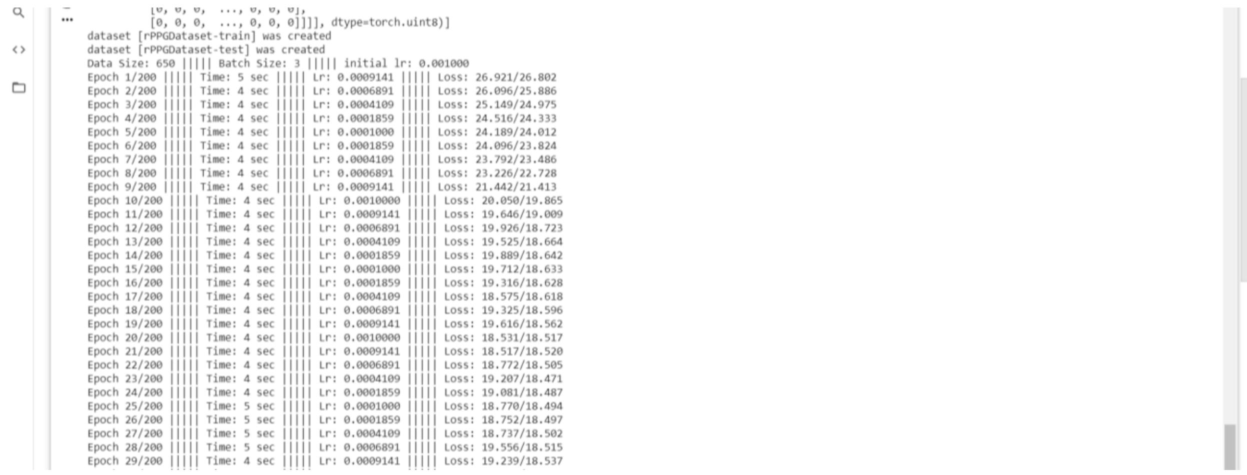


Training on UBFC Dataset: After 50 epochs, Training Accuracy=0.12, After 50 epochs, It did not really change much at all. I got low accuracy on Training with UBFC Dataset. And got testing accuracy of about 0.030 on UBFC dataset and my facial video.



Model 4 - Meta-rPPG: Remote Heart Rate Estimation Using a Transductive Meta-Learner

This method did not have proper documentation. I tried it but did not get it working. Because they did not define the metaset variable that they were getting from example.pth (dataset file provided by them). But I trained the model with example.pth input file containing the facial videos of authors of the paper. I trained the model for 200 epochs and I got the training loss of 14.729.



```
          [0, 0, 0,  ..., 0, 0, 0],
          [0, 0, 0,  ..., 0, 0, 0]]]], dtype=torch.uint8)]
dataset [rPPGDataset-train] was created
dataset [rPPGDataset-test] was created
Data Size: 650 ||||| Batch Size: 3 ||||| initial lr: 0.001000
Epoch 1/200 ||||| Time: 5 sec ||||| Lr: 0.0009141 ||||| Loss: 26.921/26.802
Epoch 2/200 ||||| Time: 4 sec ||||| Lr: 0.0006891 ||||| Loss: 26.096/25.886
Epoch 3/200 ||||| Time: 4 sec ||||| Lr: 0.0004109 ||||| Loss: 25.149/24.975
Epoch 4/200 ||||| Time: 4 sec ||||| Lr: 0.0001859 ||||| Loss: 24.516/24.333
Epoch 5/200 ||||| Time: 4 sec ||||| Lr: 0.0001000 ||||| Loss: 24.189/24.012
Epoch 6/200 ||||| Time: 4 sec ||||| Lr: 0.0001859 ||||| Loss: 24.096/23.824
Epoch 7/200 ||||| Time: 4 sec ||||| Lr: 0.0004109 ||||| Loss: 23.792/23.486
Epoch 8/200 ||||| Time: 4 sec ||||| Lr: 0.0006891 ||||| Loss: 23.226/22.728
Epoch 9/200 ||||| Time: 4 sec ||||| Lr: 0.0009141 ||||| Loss: 21.442/21.413
Epoch 10/200 ||||| Time: 4 sec ||||| Lr: 0.0010000 ||||| Loss: 20.050/19.865
Epoch 11/200 ||||| Time: 4 sec ||||| Lr: 0.0009141 ||||| Loss: 19.646/19.009
Epoch 12/200 ||||| Time: 4 sec ||||| Lr: 0.0006891 ||||| Loss: 19.926/18.723
Epoch 13/200 ||||| Time: 4 sec ||||| Lr: 0.0004109 ||||| Loss: 19.525/18.664
Epoch 14/200 ||||| Time: 4 sec ||||| Lr: 0.0001859 ||||| Loss: 19.889/18.642
Epoch 15/200 ||||| Time: 4 sec ||||| Lr: 0.0001000 ||||| Loss: 19.712/18.633
Epoch 16/200 ||||| Time: 4 sec ||||| Lr: 0.0001859 ||||| Loss: 19.316/18.628
Epoch 17/200 ||||| Time: 4 sec ||||| Lr: 0.0004109 ||||| Loss: 18.575/18.618
Epoch 18/200 ||||| Time: 4 sec ||||| Lr: 0.0006891 ||||| Loss: 19.325/18.596
Epoch 19/200 ||||| Time: 4 sec ||||| Lr: 0.0009141 ||||| Loss: 19.616/18.562
Epoch 20/200 ||||| Time: 4 sec ||||| Lr: 0.0010000 ||||| Loss: 18.531/18.517
Epoch 21/200 ||||| Time: 4 sec ||||| Lr: 0.0009141 ||||| Loss: 18.517/18.520
Epoch 22/200 ||||| Time: 4 sec ||||| Lr: 0.0006891 ||||| Loss: 18.772/18.505
Epoch 23/200 ||||| Time: 4 sec ||||| Lr: 0.0004109 ||||| Loss: 19.207/18.471
Epoch 24/200 ||||| Time: 4 sec ||||| Lr: 0.0001859 ||||| Loss: 19.081/18.487
Epoch 25/200 ||||| Time: 5 sec ||||| Lr: 0.0001000 ||||| Loss: 18.770/18.494
Epoch 26/200 ||||| Time: 5 sec ||||| Lr: 0.0001859 ||||| Loss: 18.752/18.497
Epoch 27/200 ||||| Time: 5 sec ||||| Lr: 0.0004109 ||||| Loss: 18.737/18.502
Epoch 28/200 ||||| Time: 5 sec ||||| Lr: 0.0006891 ||||| Loss: 19.556/18.515
Epoch 29/200 ||||| Time: 4 sec ||||| Lr: 0.0009141 ||||| Loss: 19.239/18.537
```

Comparison of Models

Model1(Physnet) gave good accuracy than Model2(Steven). As after 12 epochs, the Physnet model gave training loss of 0.164405 and Steven model gave training loss of 2.19. The implementation of these two models are also very similar and written by same author. The Steven Model is special in the fact that it is made to process highly compressed images through the model and still give good results. But we did not use that special feature in our project as we were not working on compressed videos. Model3(3dcnn) gave good accuracy results on synthetic video data but did not work for UBFC dataset. It gave only 11% accuracy on training with UBFC dataset and about 4% accuracy on testing dataset. It seems like 3dcnn model was specially made for training synthetic video dataset. Model4(meta-rppg) did not work on UBFC Dataset. It seems like it need more preprocessing steps which are not documented in the paper.

REFERENCES

- Frédéric Bousefsaf, Alain Pruski, Choubeila Maaoui, 3D convolutional neural networks for remote pulse rate measurement and mapping from facial video, Applied Sciences, vol. 9, n° 20, 4364 (2019)
- Lee, E., Chen, E. and Lee, C.Y., 2020. Meta-rPPG: Remote Heart Rate Estimation Using a Transductive Meta-Learner. arXiv preprint arXiv:2007.06786.
- Zitong Yu , Wei Peng1*, Xiaobai Li1, Xiaopeng Hong2,1, Guoying Zhao, Remote Heart Rate Measurement from Highly Compressed Facial Videos: an End-to-end Deep Learning Solution with Video Enhancement, University of Oulu, Finland, arXiv:1907.11921v1 [eess.IV] 27 Jul 2019
- Zitong Yu, Xiaobai Li b, Guoying Zha, Remote Photoplethysmograph Signal Measurement from Facial Videos Using Spatio-Temporal Networks, Center for Machine Vision and Signal Analysis University of Oulu FI-90014, Finland.