PhysBench: A Benchmark Framework for Remote Physiological Sensing with New Dataset and Baseline

Kegang Wang, Yantao Wei, Mingwen Tong, Jie Gao, Yi Tian, YuJian Ma, ZhongJin Zhao
Faculty Of Artificial Intelligence in Education, Central China Normal University
Wuhan, Hubei, China
{kegangwang, gaojie9812, yitian, Msilent, zhongjinzhao}@mails.ccnu.edu.cn
{yantaowei, mingwentong}@ccnu.edu.cn

Abstract

In recent years, due to the widespread use of internet videos, Remote Physiological Sensing has gained more and more attention in the fields of affective computing and telemedicine. Recovering physiological signals from facial videos is a challenging task that involves a series of preprocessing, image algorithms, and post-processing to finally restore waveforms. We propose a complete and efficient end-to-end training and testing framework that provides fair comparisons for different algorithms through unified preprocessing and post-processing. In addition, we introduce a highly synchronized lossless format dataset along with a lightweight algorithm. The dataset contains over 32 hours (3.53M frames) of video from 58 subjects; by training on our collected dataset both our proposed algorithm as well as existing ones can achieve improvements. PhysBench source code:https://github.com/KegangWangCCNU/PhysBench.

1 Introduction

In recent years, many publicly available datasets have emerged in the field of remote physiological sensing[1; 2; 3; 4; 5; 6; 7; 8; 9; 10], with many datasets focusing on providing benchmark tests for scenarios with greater diversity in motion, age, ethnicity, gender, and lighting conditions. These datasets can fully test the performance of various algorithms under different situations. A larger and more diverse dataset is always desired by the community. However, in the process of collecting datasets, two important issues have not received enough attention in previous work. ① **Whether the dataset uses lossless formats.** ② **Whether the videos in the dataset are highly synchronized with physiological signals.**

Restoring Blood Volume Pulse (BVP) signals from compressed videos is challenging. McDuff, et al.(2017)[11] and Yu, et al.(2019)[12] focus on the discussion and processing of compressed video. Compressed video formats can cause irreversible damage to physiological signals, making it difficult for remote Photoplethysmography (rPPG) algorithms to work on compressed videos. Many large datasets may not have collected uncompressed data in the correct way, which blurs the challenges of the dataset: is it the complexity of the scene or the compression format of the video that makes the dataset challenging? To obtain raw RGB or YUV format videos, professional equipment is usually required, for example, as in PURE[4] and AFRL[1]. In AFRL, a total of 12 GigE interfaces were used to capture BG bayer data from 9 cameras, providing a bandwidth of 2.605 Gb/s. While webcams compress the video during capture, which is independent of the video storage format. Under normal circumstances, the camera for mobile phones cannot directly capture raw pictures. Some brands of mobile phones, such as Samsung, allow saving raw photos (RAW format), but this feature is not applicable to videos. Therefore, it is difficult to obtain uncompressed video from a camera with a mobile phone without using dedicated software to read the underlying data and to obtain

Preprint. Under review.

lossless signals from USB webcams, specific APIs (rather than default settings) need to be used as in UBFC[6].

Some articles have mentioned data synchronization issues[13; 14; 15]. For the rPPG task, required filtering noise from facial videos and restoring BVP signals. The most commonly used losses are Mean Absolute Error (MAE) and Mean Squared Error (MSE). However, these loss functions are very sensitive to latency. If the synchronization between video and sensor signals is low, the training performance will be significantly reduced. In response to the issue of data synchronization, the community has designed some loss functions[13; 14; 15; 16; 17] to replace MAE and MSE, which can achieve better results on most datasets. A more common practice is to manually align during the preprocessing stage, obtain coarse signals using unsupervised methods, and synchronize them with the video. These preprocessing steps are almost essential for all rPPG studies, increasing the complexity of data processing and unsupervised methods cannot guarantee optimal results. liu, et al.(2022)[18] have pointed out that in cross-dataset-test, models trained on different training sets have significant performance differences. Therefore, collecting appropriate training sets may be more urgent than developing more complex and powerful models. Based on the above two points, we believe that it is necessary to collect a highly synchronized lossless format dataset, We call it the Remote Learning Affect and Physiologic dataset(RLAP). We have collected some basic information about RLAP, and a comparison with other datasets can be found in Table 1.

Table 1: Basic information of some major datasets

Dataset	Participants	Frames	Hours	PPG	Signal offset	Lossless format
AFRL[1][1]	25	97.2M[2]	25	YES	0	YES
PURE[4]	10	106K	1	YES	0	YES
UBFC[6]	42	75K	0.7	YES	>0.5s[3]	YES
MMSE-HR[2]	58	435K	4.8	ECG	0	NO
MAHNOB-HCI[8]	30	25.2M[4]	19.4	ECG	0	NO
VIPL-HR[3]	107	2.14M[5]	19.8	YES	>0.5s	NO
MMPD[10]	33	1.15M	10.6	YES	<0.2s	NO
RLAP	58	3.53M[6]	32.7[6]	YES	0	YES

[1] The authors did not specify that this is a public dataset.
[2] Use 9 RGB cameras to synchronously record at 120fps.
[3] Only a portion of the videos have significant offsets.
[4] Use 1 RGB camera and 5 BW cameras to synchronously record at 60fps.
[5] Using an estimate of 30fps, the actual fps fluctuates between 15 and 30.
[6] If used only for studying emotions or engagement, more frames are available.

On small devices, such as mobile phones, capturing physiological signals through cameras is exciting. Many works focus on lightweight end-to-end models[13; 19; 20; 16; 14], which implies the possibility of real-time operation on lightweight devices. Considering the battery capacity and heat dissipation conditions of mobile devices, it is worth paying attention to the development of much smaller light-load algorithms. In this paper, we designed a lightweight network based on 1D-CNN, which was difficult to train on previous datasets, however, under our developed PhysBench framework and RLAP dataset, this model has been fully trained.

Although some works[21; 22; 23; 18] focus on testing rPPG algorithms, early works does not concentrate on training and testing neural algorithms. To the best of our knowledge, only rPPG Toolbox[18] can train neural models; however, their framework is mainly used for reproducing existing lightweight models and has a high degree of coupling between preprocessing, model and post-processing as well as deep learning frameworks, without specifically optimizing for the development of new models. In order to facilitate the development, training, and testing of new models, make experiments easy to reproduce, and help subsequent researchers quickly get started in this field, we have developed a new training and testing framework for neural models called PhysBench.

This paper makes the following contributions:

- A highly synchronized lossless format dataset called RLAP is collected, and by training on the collected dataset, several neural algorithms(DeepPhys[19], TS-CAN[20], PhysNet[13],

PhysFormer[17]) can achieve accuracy improvements. The data collection uses a self-written program, and the dataset and collection tools will be made public.

- A very small end-to-end model is proposed with only 1/30 of the computational overhead of past small models; it can achieve comparable or better performance when trained on the RLAP dataset.
- An end-to-end training and testing framework called PhysBench is proposed, which unifies preprocessing and postprocessing while offering faster training and testing speeds. This framework allows for convenient visualization of model outputs, the addition of new datasets, new algorithms, or new metrics.

2 Dataset

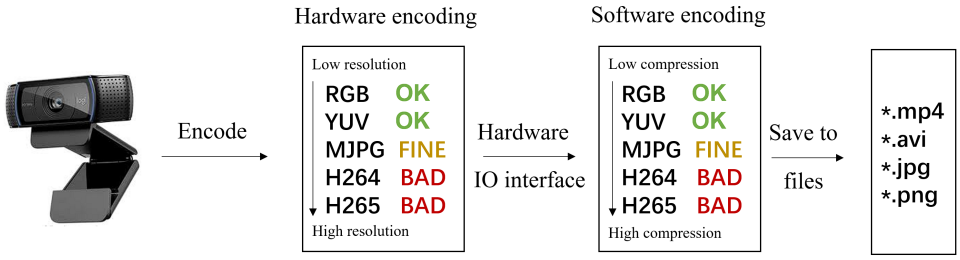

Figure 1: Video encoding process. Usually, software encoding is set to lossless format, but once hardware uses compression encoding, the rPPG signal may be damaged or even lost.

As shown in Fig.1 and UVC document[1], the acquisition of video signals involves two encoding processes. Usually, cameras cannot directly transmit raw RGB or YUV signals. For example, a 1920x1080@30fps RGB video requires about 180MB/s of bandwidth for transmission. Therefore, to transmit images within a limited bandwidth, the camera's built-in hardware will compress the video stream. The compression algorithm depends on resolution and frame rate, and high-end cameras usually have higher resolutions and frame rates; thus they use algorithms with higher compression ratios by default. **This might be fatal for rPPG because inter-frame compression algorithms like H264 can severely damage rPPG signals[11]**. We used a Logitech C930c webcam to capture videos that support MJPG and YUY2 (YUV422) formats; by default, it uses MJPG format to achieve 1920x1080@30fps video transmission while specifying YUY2 format through API allows for transmitting raw images at 640x480@30fps.

To collect the RLAP dataset, We developed a data recording program that collects raw data from Logitech C930c webcam and Contec CMS50E pulse oximeter. We recorded two types of videos: the first type is common online course scenarios, using 1920x1080 resolution, MJPG encoding for acquisition, and saved in MJPG format. The second type is designed for rPPG-specific scenes, using 640x480 resolution, YUY2 lossless encoding for acquisition, and saved in both MJPG and RGB formats. In the rPPG sub-dataset, we designed four different scenarios including varying brightness levels, focus levels, and facial muscle movements. Compared to the entire dataset, its scenes are more complex with higher data quality due to not using compression algorithms. To ensure strict synchronization between video and ground truth BVP signal, just like the method used in PURE[4], we saved UNIX timestamps of all signals during collection that are provided within the dataset.

During the data recording process, the subjects need to complete a series of tasks or watch videos. After completing the specified task, the subject will rest for a while, and then the experimenter will assign him/her the next task. All 58 subjects (16 males and 42 females) were Chinese students, mainly master's degree students, and some girls might wear makeup. Excluding unusable data due to format errors, unfinished tasks or equipment failures, RLAP provided more than 32 hours (3.53 million frames) of available video. More details about RLAP can be found in Table 2. Some subject images can be seen in Fig. 2

[1] https://www.usb.org/document-library/video-class-v15-document-set

The data collection environment is facing the window and has indoor artificial light sources. The subject is sitting in front of the computer, about 1 meter away from the camera. The subject is usually looking at the computer screen, but sometimes he or she may read or write on paper forms in front of him or her. During the collection process, the subject is holding a mouse or pen with their right hand to complete tasks, wearing a CMS50E pulse oximeter on their left hand, and being instructed to minimize left-hand movement to ensure stable signal acquisition. The subjects' heads are not fixed, so they can move naturally.

Table 2: Data collection workflows

Sub-dataset	Task or induction video	Duration(S)	Camera codec	Video codec	Resolution
rPPG	Relaxed	120	YUY2	RGB,MJPG,H264	640×480
	Relaxed & Dark	120			
	Play a game	120			
	Read an article	120			
Emotion	Natural scenery	120	MJPG	MJPG,H264	1920×1080
	Puzzle game	180			
	Comedy	120			
	Illusion picture	20			
	Academic paper	60			
	Video about yawn	60			
Engagement	Video-based learning	240	MJPG	MJPG,H264	1920×1080
	Textbook-based learning	480			
	Watch a public class	420			

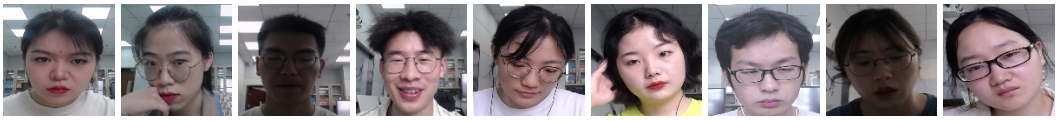

Figure 2: Overview of our dataset. These data are collected from webcams, with students sitting in front of computers completing different tasks. The data include changes in facial expressions, attention levels, lighting conditions, and facial muscle movements.

3 Algorithm and Experimental Framework

Based on the above work, we designed a very small, one-dimensional convolution-based algorithm. To our knowledge, no one has attempted to use 1D CNN for rPPG in the past, and most convolutional models[16; 14; 13; 24; 19; 20; 25; 26; 11; 27] have used 3D CNN or 2D CNN. Training of 1D CNN is not easy because it only models the time dimension and all spatial information is ignored, making it particularly sensitive to temporal offsets. Since many physiological signals in the main public datasets are not strictly synchronized with frames or even though the dataset is strictly synchronized but smaller in scale, training this model is quite challenging. However, through the PhysBench framework and RLAP dataset, this model is eventually fully trained.

3.1 Algorithm

Handcrafted algorithms[28; 29; 30; 31; 32] based on separate reflection components. Shafer.(1985)[33] assume that the BVP signal in the RGB image comes through a linear combination of different frequency rays, while the skin-reflected light contains a specular reflection component and a diffuse reflection component. In post-processing, stationary signals, and noise are filtered, while periodic signals generated by fluctuations in hemoglobin concentration are passed. Therefore, our algorithm is divided into two parts. The first part combines the RGB channels and frame numbers of the original video (450x8x8x3), and merges the width and height to obtain a 1350x64 RGB sequence. A convolution kernel with a size of 3 and a stride of 3 is used to mix the RGB sequence, resulting in a coarse signal of 450x64. The second part involves using multiple convolution filters with activation functions to perform nonlinear filtering on the coarse signal, obtaining BVP signals.

We drew inspiration from the Fast Fourier Convolution (FFC)[34; 35], which is very effective in processing periodic signals, such as audio information. We designed a spectral transformation module and added it to the 1D CNN, enhancing its performance. The final model alternates between four temporal-domain CNN layers and spectral transformations, see Fig.3.

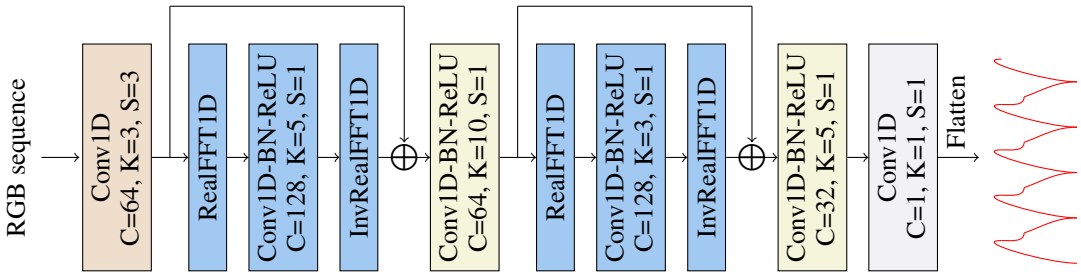

Figure 3: Proposed algorithm. This is a method for extracting BVP signals directly from RGB sequences using 1D CNN.

We implemented the spectral transformation module using Fast Fourier Transform (FFT) and 1D CNN. For a signal $\mathbf{Y} \in \mathbb{R}^{N \times C}$, the spectral filtering layer first performs a Real Fast Fourier Transform (RFFT) on each channel, obtains frequency domain signal $\mathbf{Y}_{Freq} \in \mathbb{Z}^{\frac{\lfloor N+1 \rfloor}{2} \times C}$, then decomposes it into a real part $\mathbf{Y}_{Real}$ and an imaginary part $\mathbf{Y}_{Img}$, then combines them on the channel as $\mathbf{Y}_{Comb} \in \mathbb{R}^{\frac{\lfloor N+1 \rfloor}{2} \times 2C}$. Then a convolution layer is applied to it and re-decomposes the output into real and imaginary parts. It is converted to complex numbers and recovered to the time domain signal by Inverse Real Fast Fourier Transform (IRFFT). The output signal is mixed with the original signal through the residual connection, and the number of channels remains constant throughout the process.

In order to achieve efficient development and fair testing, we have written a new Remote Physiological Sensing framework that uses the HDF5 file format to define a unified dataset, training data, and test results, as well as unifying preprocessing and post-processing processes. Since it is developed for HDF5 files, each data processing step is decoupled and can be added by performing additional operations on the HDF5 file. Based on this, we developed our model and reproduced PhysNet[13], DeepPhys[19], TS-CAN[20], and PhysFormer[17].

3.2 PhysBench Framework

In order to implement a complete, fair, and scalable training framework, we have defined some basic concepts and specifications in Remote Physiological Sensing tasks. The most important of these are the three types of files generated during the experiment:

- Standard dataset files, which are datasets saved in HDF5 format that have undergone facial detection and cropping and only contain RGB videos of the face area saved as unsigned int8. In addition to this, it also contains some metadata such as lighting conditions, skin color, motion etc.
- Standard training set and validation set files, which stored continuously on disk in HDF5 format that conform to model input shapes; they are obtained from standard dataset files through segmentation and data augmentation. These files can be read at high speed continuously to meet training needs; PhysBench has built-in optional data augmentation algorithms. For the sake of convenience, we vividly call it datatape.
- Standard result files, which are HDF5 formatted files containing concatenated model outputs including Ground Truth values from labels, concatenated model output, and timestamps from labels. This type of file can be used directly for visualization where visualization programs can draw waveforms while playing original videos simultaneously. It can also be used for calculating indicators, where the indicator algorithm only needs to read two types of waveforms to obtain any required indicator. Moreover, during this process, post-processing algorithms such as band-pass filtering can be uniformly applied.

The PhysBench framework is based on the above basic file definitions, and around these definitions, we have properly specified the entire experimental process.

3.2.1 Preprocessing and Unified Dataset Format

Typically, end-to-end models cannot completely avoid preprocessing, and face detection is necessary. During the preprocessing stage, facial tracking needs to be implemented using face detection tools and capturing facial images from each frame. At the same time, it is also necessary to read the waveform signals corresponding to each frame from the dataset. For most models, only facial videos are needed, so it is appropriate to store facial images in arrays. We use MediaPipe to capture facial images frame by frame and apply a filter to avoid detection box disturbances. For each video, three sets of data can be obtained: facial image sequences, physiological signal sequences, and timestamp sequences. These three sets of data will be saved in an HDF5 group and resampled to align the signals with the timestamps. Facial images can be used at different resolutions; by default, it use a 128x128 resolution. All data is stored in standard dataset files.

According to the PhysBench framework, this form of the dataset can be used to generate training data and can also be directly used for testing model performance.

3.2.2 Data Augmentation and Model Training

Facial videos for model training requires two operations: 1. Video slicing and sampling to conform to the input dimensions of the model. 2. Regularization. We have defined a data type called datatape to meet the needs of these two operations. First, the algorithm divides and samples the original video according to requirements, obtaining the training data needed by the model, and then HR expansion algorithm obtains some additional samples by changing their heart rate through time dimension sampling. Subsequently, these two types of data will be continuously saved in HDF5 files. Datatape files are read asynchronously and sequentially, allowing them to load training data from SSDs at very high speeds. This is especially important for large datasets, as it is not possible to load all the data into memory. According to our tests, the loading speed during training is higher than the GPU throughput, ensuring that training is not slowed down by data loading.

The PhysBench framework does not depend on any deep learning framework, so it needs to be wrapped when loading datatape files. We wrote a generator to load data from datatape files, which can be wrapped into a dedicated format for deep learning frameworks. During the wrapping process, the dataset can be further augmented, such as adding Gaussian noise, flipping images, changing color temperature and so on. These operations can be directly performed on GPUs using deep learning frameworks.

3.2.3 Post-processing and Visualization

Since the model only inputs a small segment of video at a time, when evaluating on the entire video, it is necessary to first divide, input into the model, obtain output, and then concatenate to form a coarse signal. Many models also require additional filters to obtain smooth signals. In the evaluation function, segmentation and concatenation will be automatically completed and save concatenated results, ground truth, timestamp, and meta-labels in result files. Therefore result in files do not directly include metrics; an additional function is needed to estimate accuracy.

In the estimation function, an optional band-pass filter is first added to ensure the uniformity of post-processing filtering, heart rate is extracted from the Power Spectrum Density (PSD) using the Welch method. Three metrics are also added: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Pearson correlation coefficient ρ , with a moving window (default 30s) for continuous evaluation. The function returns two types of results, one for the entire video and one for the moving window. Thanks to the efficient execution framework, testing speed is very fast; for example, our model takes only 3 seconds to perform full inference on UBFC or PURE datasets and obtain metrics while being immediately visualizable.

The visualization page is completed using the Panel library. Since each result file contains meta-tags, these files are linked to the dataset files, and the program can read the facial images corresponding to each physiological signal at the same time. This is different from Liu, et al.(2022)[18], as PhysBench helps researchers observe the impact of different head movements on model output.

4 Experiment and Results

All experiments were conducted in the PhysBench framework. The experimental platform used was CPU: AMD Ryzen 9 5950X, GPU: Nvidia Tesla M40 24G, OS: Windows 11. The proposed models, PhysNet[13], DeepPhys[19], TS-CAN[20] were trained on Tensorflow 2.6; PhysFormer[17] was trained on Pytorch 2.0. For pre-trained models, we tested their mobile CPU inference performance on Raspberry Pi 4B (CPU: Cortex-A72 4 cores, OS: Debian 11).

In PhysBench, PhysNet[13], PhysFormer[17], TS-CAN[20], and DeepPhys[19] were reproduced with some differences from the open-source code. Both TS-CAN and DeepPhys used a resolution of 36x36 instead of 72x72 in the open-source code to ensure mobile inference performance claimed in the MTTS-CAN paper and fair comparison. The training parameters for PhysNet and PhysFormer were adjusted to achieve better performance. For specific details, please refer to the source code.

4.1 Datasets and Metrics

In addition to the RLAP dataset, there are two other datasets used for cross-dataset testing: UBFC[6] includes 42 video clips from 42 subjects, each clip lasting 1 minute. PURE[4] includes 59 videos from 10 subjects, each clip lasting 1 minute, with each subject required to perform six types of head movements: steady, talking, slow/fast translation between head movements and the camera plane, small/medium head rotation.

There are three indicators used to evaluate test results: Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Pearson Correlation Coefficient (ρ).

4.2 Intra-dataset Testing

We randomly divided the RLAP dataset into training, validation, and testing sets according to the subjects, and the partition details will be provided in the dataset. All algorithms are tested on both the entire test set and rPPG subset. The entire test set can better reflect algorithm accuracy in remote learning scenarios, while rPPG subset is only for Remote Physiological Sensing. Their differences lie in that under remote learning scenarios, faces usually do not face cameras directly and compression algorithms are generally used; however, lighting conditions and head movements are relatively stable. The more general rPPG subset uses lossless formats and includes lighting conditions as well as facial muscle activity. In the RLAP dataset, due to the varying and longer video lengths, all test results are calculated using a 30s moving window, and using the entire video in UBFC[6] and PURE[4]. Test results are shown in Table 3.

Table 3: Intra-dataset Testing on RLAP

Method	**Entire dataset**			**rPPG subset**		
	MAE↓	RMSE↓	ρ↑	MAE↓	RMSE↓	ρ↑
Ours	**1.07**	4.15	0.917	0.81	2.97	0.953
DeepPhys[19]	1.52	4.40	0.906	1.76	4.87	0.877
TS-CAN[20]	1.23	**3.59**	**0.937**	1.23	3.82	0.922
PhysNet[13]	1.12	4.13	0.916	1.04	3.80	0.923
PhysFormer[17]	1.56	6.28	0.803	**0.78**	**2.83**	**0.957**
CHROM[30]	6.86	15.57	0.357	6.30	14.52	0.441
POS[31]	4.25	12.06	0.501	3.56	10.14	0.634
ICA[29]	6.05	13.3	0.380	6.45	13.00	0.470

4.3 Cross-dataset Testing

Thanks to the high-quality data of the RLAP dataset, models trained on RLAP have better transferability. We use UBFC[6] and PURE[4] as test sets separately; when using UBFC as the testing set, the training sets are RLAP, PURE, and SCAMPS[9]; when using PURE as the testing set, the training sets are RLAP, UBFC, and SCAMPS. The training results on SCAMPS comes from Liu et al.(2022)[18]. Cross-dataset testing is commonly used to evaluate the generalization ability of

models, but here we mainly compare the transferability of training sets: that is, whether a model trained on a certain dataset has the potential to be applied to other scenarios. We have additionally annotated the differences between training results on the RLAP dataset and past training sets to reflect the contribution of the dataset.

Table 4: Cross-dataset Testing on UBFC

Training set	**SCAMPS**			**PURE**			**RLAP**		
Method	MAE↓	RMSE↓	ρ↑	MAE↓	RMSE↓	ρ↑	MAE↓	RMSE↓	ρ↑
Ours	-	-	-	1.16	1.86	0.994	**0.87**$^{-0.29}$	**1.40**$^{-0.46}$	**0.997**$^{+.003}$
DeepPhys[19]	3.83	12.5	0.82	3.12	8.78	0.897	1.06$^{-2.06}$	1.51$^{-7.27}$	**0.997**$^{+.100}$
TS-CAN[20]	6.86	16.1	0.76	1.03	1.63	0.996	0.99$^{-0.04}$	1.44$^{-0.19}$	**0.997**$^{+.001}$
PhysNet[13]	6.46	12.2	0.823	1.02	1.65	0.996	0.92$^{-0.10}$	1.46$^{-0.19}$	**0.997**$^{+.001}$
PhysFormer[17]	-	-	-	1.66	3.11	0.988	1.06$^{-0.60}$	1.53$^{-1.58}$	**0.997**$^{+.009}$

Table 5: Cross-dataset Testing on PURE

Training set	**SCAMPS**			**UBFC**			**RLAP**		
Method	MAE↓	RMSE↓	ρ↑	MAE↓	RMSE↓	ρ↑	MAE↓	RMSE↓	ρ↑
Ours	-	-	-	25.61	43.39	0.248	**0.37**$^{-25.2}$	**0.63**$^{-42.9}$	**1.000**$^{+.752}$
DeepPhys[19]	3.46	12.9	0.84	10.04	17.86	0.627	2.80$^{-0.66}$	8.31$^{-4.59}$	0.937$^{+.097}$
TS-CAN[20]	6.67	18.9	0.62	4.79	11.87	0.773	2.12$^{-2.67}$	6.67$^{-5.20}$	0.960$^{+.187}$
PhysNet[13]	19.95	27.57	0.16	8.82	19.43	0.694	0.51$^{-8.31}$	0.91$^{-18.5}$	0.999$^{+.305}$
PhysFormer[17]	-	-	-	17.60	29.02	0.398	1.63$^{-16.0}$	9.45$^{-19.6}$	0.914$^{+.516}$

4.4 Computational Overhead

Small models imply the possibility of running on low-cost devices, and due to the ubiquity of embedded CPUs, algorithms with low computational overhead will have broad application prospects. We tested the average frame time of several algorithms on a typical mobile device: Raspberry Pi 4B (CPU: Cortex-A72 4 core). Compared with the best models PhysNet and TS-CAN, our model has a lower computational overhead, about 1/30 of theirs, while having similar or better performance, see Table 6. In addition, our model uses 8x8 resolution input to make facial contours as blurred as possible in scenarios where user information analysis is required to protect user privacy. It should be noted that during the testing phase, our model (and any other small-scale input models) must use area average sampling (cv2.INTER_AREA) to achieve the performance reported in this paper; please refer to PhysBench source code for sampling algorithms.

Table 6: Computational Overhead on Mobile CPUs

Model	Resolution	Frame FLOPs (M)	Frame Time (ms)
ours	8x8	0.26	0.36
DeepPhys[19]	36x36	52.16	9.09
TS-CAN[20]	36x36	52.16	10.72
PhysNet[13]	32x32	54.26	9.69
PhysFormer[17]	128x128	323.80	150

5 Discussion and Conclusion

In this paper, we mention some previous works on datasets that may have flaws, such as not strictly synchronizing physiological signals with frames or neglecting the impact of compression algorithms. these defects may affect the training of models, so we raise a question: if the quality of the dataset is higher, how much improvement can be achieved in model performance? Is it possible to develop

smaller, more accurate and efficient models? Therefore, we design a 1D CNN-based model that is difficult to reproduce with previous work. In order to train it fully on the collected dataset, we construct the PhysBench framework that allows for easy development of new models and fair testing. Ultimately proving that with high-quality datasets and well-established training frameworks, very few parameters are needed to achieve better performance than state-of-the-art models.

In the intra-dataset testing, according to Table 3, our algorithm was defeated by TS-CAN and PhysFormer on both the entire test set and the rPPG subset. However, the metrics are close. For example, although the RMSE is higher than TS-CAN on the entire test set, it is lower than TS-CAN on the rPPG subset; in terms of all metrics in rPPG subset, our model performs worse than PhysFormer but better than PhysFormer on all metrics for entire test set. Therefore, it can be considered that our model's performance is close to that of state-of-the-art models.

In cross-dataset testing, our algorithm outperforms all baselines, which implies that our algorithm may have better generalization. In addition, all models are trained and compared on multiple training sets, and the performance of models trained on different training sets is quite different. For example, our model can be trained on PURE and tested on UBFC with reasonable performance, but once the training set and test set are flipped, the performance becomes very poor. This phenomenon may indicate the contribution of high-quality training sets: because according to Table 1, PURE is a highly synchronized dataset while UBFC is not. Our model can be trained on PhysBench in PURE but still needs to pay attention to some details; please refer to the tutorial section in the source code. According to Table 4 and Table 5, all baselines can be trained on RLAP dataset and improve their performances, which sufficiently reflects this paper's contribution in terms of datasets.

In the computational overhead test, according to Table 6, our model has a very small computational overhead, which is almost negligible for inexpensive mobile CPUs. This is important for implementing algorithms on embedded devices: since computing resources are very precious for embedded devices and need to be carefully allocated, we hope that the algorithm will hardly increase the system load. In addition, many battery-powered devices are very sensitive to load, and extremely low computational overhead can extend battery life as much as possible. On chips smaller than Raspberry Pi, such as control chips inside cameras, our algorithm still has the possibility of implementation so that physiological signals can be obtained directly from raw RGB signals inside cameras and transmitted through UVC extension. Considering widely used compression algorithms on cameras, implementing rPPG algorithms internally may have higher accuracy and be more convenient to use.

We acknowledge that our dataset has certain limitations, as our laboratory's past research focused on the educational field, so this dataset was designed to analyze the relationship between physiological signals and students' learning emotions or engagement. It is not a comprehensive Remote Physiological Sensing dataset. However, our experimental results prove that even so, it is a high-quality training set. In addition, we will make public the data collection tools for this dataset, which rely only on very inexpensive devices (Logitech C930c, Contec CMS50E), costing less than 50 US dollars in total to collect highly synchronized lossless format datasets. This will greatly facilitate data collection work and be directly compatible with the PhysBench framework. We hope that our work can inspire others and lead to more large-scale datasets in the future to establish a more complete baseline for the field of Remote Physiological Sensing.

References

[1] Justin R. Estepp, Ethan B. Blackford, and Christopher M. Meier. Recovering pulse rate during motion artifact with a multi-imager array for non-contact imaging photoplethysmography. In 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pages 1462–1469, 2014.

[2] Zheng Zhang, Jeff M. Girard, Yue Wu, Xing Zhang, Peng Liu, Umur Ciftci, Shaun Canavan, Michael Reale, Andy Horowitz, Huiyuan Yang, Jeffrey F. Cohn, Qiang Ji, and Lijun Yin. Multimodal spontaneous emotion corpus for human behavior analysis. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[3] Xuesong Niu, Hu Han, Shiguang Shan, and Xilin Chen. Vipl-hr: A multi-modal database for pulse estimation from less-constrained face video. In Asian conference on computer vision, pages 562–576. Springer, 2018.

[4] Ronny Stricker, Steffen Müller, and Horst-Michael Gross. Non-contact video-based pulse rate measurement on a mobile service robot. In The 23rd IEEE International Symposium on Robot and Human Interactive Communication, pages 1056–1062, 2014.

[5] Rita Meziati Sabour, Yannick Benezeth, Pierre De Oliveira, Julien Chappe, and Fan Yang. Ubfc-phys: A multimodal database for psychophysiological studies of social stress. IEEE Transactions on Affective Computing, pages 1–1, 2021.

[6] Serge Bobbia, Richard Macwan, Yannick Benezeth, Alamin Mansouri, and Julien Dubois. Unsupervised skin tissue segmentation for remote photoplethysmography. Pattern Recognition Letters, 124:82–90, 2019.

[7] Xiaobai Li, Iman Alikhani, Jingang Shi, Tapio Seppanen, Juhani Junttila, Kirsi Majamaa-Voltti, Mikko Tulppo, and Guoying Zhao. The obf database: A large face video database for remote physiological signal measurement and atrial fibrillation detection. In 2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018), pages 242–249, 2018.

[8] Wenjin Wang, Albertus C. den Brinker, Sander Stuijk, and Gerard de Haan. Algorithmic principles of remote ppg. IEEE Transactions on Biomedical Engineering, 64(7):1479–1491, 2017.

[9] Daniel McDuff, Miah Wander, Xin Liu, Brian L Hill, Javier Hernandez, Jonathan Lester, and Tadas Baltrusaitis. Scamps: Synthetics for camera measurement of physiological signals. arXiv preprint arXiv:2206.04197, 2022.

[10] Jiankai Tang, Kequan Chen, Yuntao Wang, Yuanchun Shi, Shwetak Patel, Daniel McDuff, and Xin Liu. Mmpd: Multi-domain mobile video physiology dataset, 2023.

[11] Daniel J. McDuff, Ethan B. Blackford, and Justin R. Estepp. The impact of video compression on remote cardiac pulse measurement using imaging photoplethysmography. In 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), pages 63–70, 2017.

[12] Zitong Yu*, Wei Peng*, Xiaobai Li, Xiaopeng Hong, and Guoying Zhao. Remote heart rate measurement from highly compressed facial videos: an end-to-end deep learning solution with video enhancement. In International Conference on Computer Vision (ICCV), 2019.

[13] Zitong Yu, Xiaobai Li, and Guoying Zhao. Remote photoplethysmograph signal measurement from facial videos using spatio-temporal networks. arXiv preprint arXiv:1905.02419, 2019.

[14] J. Comas, A. Ruiz, and F. Sukno. Efficient remote photoplethysmography with temporal derivative modules and time-shift invariant loss. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 2181–2190, Los Alamitos, CA, USA, jun 2022. IEEE Computer Society.

[15] Weiyu Sun, Xinyu Zhang, Ying Chen, Yun Ge, Chunyu Ji, and Xiaolin Huang. Byhe: A simple framework for boosting end-to-end video-based heart rate measurement network, 2022.

[16] Deivid Botina-Monsalve, Yannick Benezeth, and Johel Miteran. Rtrppg: An ultra light 3dcnn for real-time remote photoplethysmography. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, pages 2146–2154, June 2022.

[17] Zitong Yu, Yuming Shen, Jingang Shi, Hengshuang Zhao, Philip Torr, and Guoying Zhao. Physformer: Facial video-based physiological measurement with temporal difference transformer. In CVPR, 2022.

[18] Xin Liu, Xiaoyu Zhang, Girish Narayanswamy, Yuzhe Zhang, Yuntao Wang, Shwetak Patel, and Daniel McDuff. Deep physiological sensing toolbox. arXiv preprint arXiv:2210.00716, 2022.

[19] Weixuan Chen and Daniel McDuff. Deepphys: Video-based physiological measurement using convolutional attention networks. In Proceedings of the European Conference on Computer Vision (ECCV), September 2018.

[20] Xin Liu, Josh Fromm, Shwetak Patel, and Daniel McDuff. Multi-task temporal shift attention networks for on-device contactless vitals measurement. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 19400–19411. Curran Associates, Inc., 2020.

[21] Giuseppe Boccignone, Donatello Conte, Vittorio Cuculo, Alessandro D'Amelio, Giuliano Grossi, Raffaella Lanzarotti, and Edoardo Mortara. pyvhr: a python framework for remote photoplethysmography. PeerJ Computer Science, 8:e929, 2022.

[22] Giuseppe Boccignone, Donatello Conte, Vittorio Cuculo, Alessandro D'Amelio, Giuliano Grossi, and Raffaella Lanzarotti. An open framework for remote-PPG methods and their assessment. IEEE Access, pages 1–1, 2020.

[23] Daniel McDuff, Sarah Gontarek, and Rosalind W Picard. Remote detection of photoplethysmographic systolic and diastolic peaks using a digital camera. IEEE Transactions on Biomedical Engineering, 61(12):2948–2954, 2014.

[24] Xin Liu, Brian L. Hill, Ziheng Jiang, Shwetak Patel, and Daniel McDuff. Efficientphys: Enabling simple, fast and accurate camera-based vitals measurement, 2021.

[25] Xuesong Niu, Shiguang Shan, Hu Han, and Xilin Chen. Rhythmnet: End-to-end heart rate estimation from face via spatial-temporal representation. IEEE Transactions on Image Processing, 29:2409–2423, 2020.

[26] Xuesong Niu, Hu Han, Shiguang Shan, and Xilin Chen. Synrhythm: Learning a deep heart rate estimator from general to specific. In 2018 24th International Conference on Pattern Recognition (ICPR), pages 3580–3585, 2018.

[27] Xuesong Niu, Zitong Yu, Hu Han, Xiaobai Li, Shiguang Shan, and Guoying Zhao. Video-based remote physiological measurement via cross-verified feature disentangling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, Computer Vision – ECCV 2020, pages 295–310, Cham, 2020. Springer International Publishing.

[28] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Non-contact, automated cardiac pulse measurements using video imaging and blind source separation. Optics express, 18(10):10762–10774, 2010.

[29] Ming-Zher Poh, Daniel J McDuff, and Rosalind W Picard. Advancements in noncontact, multiparameter physiological measurements using a webcam. IEEE transactions on biomedical engineering, 58(1):7–11, 2010.

[30] Gerard de Haan and Vincent Jeanne. Robust pulse rate from chrominance-based rppg. IEEE Transactions on Biomedical Engineering, 60(10):2878–2886, 2013.

[31] Wenjin Wang, Albertus C. den Brinker, Sander Stuijk, and Gerard de Haan. Algorithmic principles of remote ppg. IEEE Transactions on Biomedical Engineering, 64(7):1479–1491, 2017.

[32] Sergey Tulyakov, Xavier Alameda-Pineda, Elisa Ricci, Lijun Yin, Jeffrey F. Cohn, and Nicu Sebe. Self-adaptive matrix completion for heart rate estimation from face videos under realistic conditions. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2396–2404, 2016.

[33] Steven A. Shafer. Using color to separate reflection components. Color Research and Application, 10:210–218, 1985.

[34] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, Advances in Neural Information Processing Systems, volume 33, pages 4479–4488. Curran Associates, Inc., 2020.

[35] Ivan Shchekotov, Pavel Andreev, Oleg Ivanov, Aibek Alanov, and Dmitry Vetrov. Ffc-se: Fast fourier convolution for speech enhancement, 2022.