Deep Learning for Improving Power-Accuracy of Heart Rate Monitors

Albert Gural (agural)
agural@stanford.edu
Stanford University - Department of Electrical Engineering

***Abstract*—A deep learning methodology is applied to the problem of determining heart rate from low sample rate PPG and accelerometer signals in the presense of motion artifact. The approach taken in this work utilizes synthetically-generated data to augment a small public dataset and compares two general methods - a time-domain approach where signals are directly fed into an LSTM and a frequency-domain approach where FFTs of the signals are fed into a fully-connected network. Performance of these models is evaluated on the ICASSP signal processing cup 2015 PPG dataset and is compared to state-of-the-art signal processing schemes for accuracy and to PPG circuits found in the ISSCC for power. Our best model achieves an accuracy of 3.09 beats per minute (BPM) average absolute error on 10× decimated data, compared to state-of-the-art 1.04 BPM avAE on original un-decimated data. Additionally, it is estimated to be capable of 25μW operation compared to a state-of-the-art 38μW system that does not handle motion artifacts.**

I. Introduction

Heart rate (HR) monitors are becoming a ubiquitous technology on portable devices such as smartphones, smart watches, and fitness trackers. These HR monitors rely on photoplethysmographic (PPG) technology which determines HR by detecting changes in skin reflectivity as blood vessels contract and dilate. To do this, they emit short pulses of light from an LED and detect the reflected light on a photodiode whose current is then digitized and processed.

Despite its widespread usage, PPG technology continues to suffer from two main issues: motion artifact (MA) noise and power consumption. MA from walking, running, and other concussive activities interferes with the raw PPG signal in a particularly devestating way - it is usually larger in magnitude and close in frequency to the desired PPG signal, making it hard to separate using traditional signal processing techniques. Power is an issue for mobile devices, since energy is constrained by the battery and the LED required for PPG sensing traditionally requires a lot of power.

This work examines smarter algorithms using deep learning to tackle the challenges mentioned above. Deep learning helps in the following ways. First, the sequential nature of the data suggests that a recurrent neural network architecture may be able to learn the HR simply by feeding it normalized raw data. Second, many traditional techniques try to model the effects of MA on PPG with a linear filter on the accelerometer. It may be the case that a more powerful function approximator would serve as a better filter, since MAs are likely to stress the system into non-linear regimes. Third, deep learning is known for its robustness to noise suggesting possible routes to saving power by reducing LED intensity (higher thermal noise) or sampling rate (higher spectral noise).

Our work focuses on a dataset from the ICASSP 2015 challenge [1]. It consists of PPG, accelerometer, and electrocardiogram (ECG) data taken from patients following a specified exercise routine. This challenge allows for a quantitative comparison between algorithms designed by the competitors and any algorithms developed in this project. ECG data serves as a baseline "true" HR and is not used for heart rate predictions.

II. Prior Art

The ICASSP 2015 challenge drew many research papers. One of the best uses a complicated, though traditional signal processing pipeline consisting of a Wiener filter and phase vocoder [2]. It operates as follows. The FFTs for 8-second windows of PPG and accelerometer data are found, then a Wiener filter is applied to the FFTs in an obvious way - the "noisy signal spectral density" coming from the PPG and the "noise spectral density" coming from the accelerometer FFTs. The derived Wiener filter is applied to the signal and peaks of the cleaned signal are found. Finally, a phase vocoder is used to get slightly more accurate heart rate measurements. Their work achieved an average absolute error per subject of betwen 0.54 and 2.61 BPM, about half the error found in the original work presenting the data [1].

A number of groups have attempted reductions along the power dimension. Traditional analog circuit optimization can lead to very low powers in the analog circuitry, such as [5]. However, LED power is still a concern and is usually minimized by reducing sample rate. A few papers look at ways to achive this goal.

One attempt to reduce sample rate is through compressive sampling [3]. Compressive sampling is a technique in which samples are taken at a much lower rate than the Nyquist rate. If you know that it is sparse in some domain, then the required sampling rate can be reduced to a value proportional to the density of non-zero elements in the sparse domain. Using clean PPG data without MA, they achieve 30× LED power reduction, although at the expense of HR accuracy and robustness to MA. Even in an environment with no MA, they only achieve around 10 BPM accuracy. One other issue with this approach is that reconstruction of the signal requires solving a convex optimization problem, which is a computationally-expensive procedure.

A more recent result attempts to achieve low power by only sampling near the peak of the signal and attempting to maintain lock to the PPG peaks [4]. Using their approach, they report power consumptions as low as 38μW at an accuracy of around 5 BPM at 60 BPM. However, their system was tested on a motion-artifact-free signal and so would be unlikely to perform well in a continuous health monitoring environment.

	[1]	[2]	[3]	[4]	[5]	This work	
Method	LED duty-cycled using pulses	LED duty-cycled using pulses	LED duty-cycled using pulses	LED duty-cycled using pulses	compressive sampling (LED only)	LED duty-cycled using pulses & window, AFE duty-cycled using window	
Technology	1.5µm	0.35µm	0.35µm	0.18µm	0.18µm	0.18µm	
Supply Voltage	5V	3.3V	2.5V	1.8V	1.2V	3.3V	
Sampling Frequency	100Hz	100Hz	100Hz	165Hz	4Hz	40Hz	100Hz
Effective LED Duty Cycle	3%	3%	0.2%	0.7%	0.0125%[3]	0.01-1%[1]	0.0175-1.75%[1]
Power Consumption (ROIC)	400µW	528µW	600µW	216µW	172µW[4]	29.1µW[1, 2]	27.4µW[1, 2]
Power Consumption (LED)	4400µW	309-1360µW	N/A	120-1125µW	43-1200µW	9-480µW[1, 5]	16-520µW[1, 5]
Heart Rate Error	N/A	N/A	N/A	N/A	10bpm	5bpm	2.1bpm
Integrated Feature Extraction	Yes (SpO2)	No	No	No	Yes (HR/HRV)	Yes (HR)	

1) Measured when locked at 60bpm using HBLL, duty cycle = 0.1-10% (40Hz), 0.25-25% (100Hz).
2) Including AFE and HBLL power consumption.
3) Calculated from the given data.
4) ROIC power consumption without reconstruction.
5) Using DCM03 (LED+photodiode), LED current: 10mA, Photodiode size: 2.5mm × 2.5mm.

Fig. 1. Comparison table of various ISSCC circuits used for low power PPG-based heart rate detection [4].

III. ICASSP Dataset

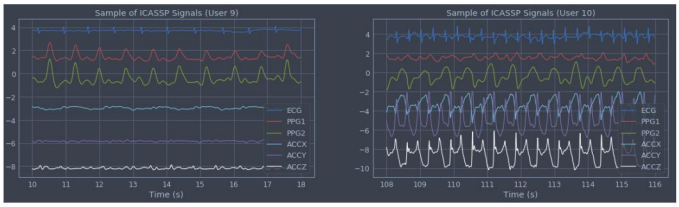

Fig. 2. Example of ICASSP data evaluation windows. Low motion artifact (left) versus high motion artifact (right).

The ICASSP dataset contains 12 samples of data from 12 different subjects. Each sample is 300-seconds long and contains two PPG signals, three-axis accelerometer signals, and an ECG signal. The ECG signal is for reference only and can not be used to determine heart rate. The ICASSP challenge was to determine the heart rate for 8-second evaluation windows every 2-seconds. Towards this goal, the dataset also contains "ground truth" heart rates for each of these evaluation windows.

Figure 2 shows some representative samples of evaluation windows from the dataset. The left shows a relatively clean signal with minimal motion artifact while the right shows a signal corrupted with heavy motion artifact. When looking at a spectrogram of the signal (fig. 3) over the full 300 seconds, it is easy to see the heart rate and motion artifact signals. However, distinguishing which is which can be difficult.

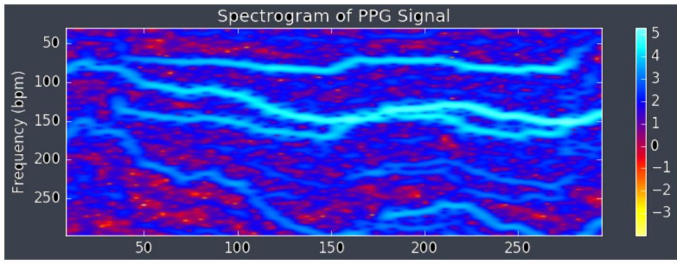

Fig. 3. Spectrogram of user 9's data.

Instead of working directly with this data, we decimate it 10× from a sample rate of 125Hz to a sample rate of 12.5Hz in order to replicate the low sample rate required for a power-efficient device. The problem, then, is to see whether deep learning is able to achieve anywhere near the state-of-the-art errors found in [2] while maintaining power levels comparable to those shown in fig. 1.

IV. Approach

One immediate problem becomes obvious when considering deep learning approaches for the ICASSP dataset: there is not enough data. In fact, there are only 1764 labeled evaluation windows. To tackle this problem, we employ synthetic data generation to augment the dataset in training. One additional benefit of generating data this way is that we can iteratively increase data difficulty to slowly converge on the best deep learning models.

Our training set consists of the first six subjects of the ICASSP data and 1.5 million synthetically-generated training examples (100k generated every two epochs of training for 30 epochs). The choice of 100k was enforced by computational memory constraints. Regenerating data every two epochs was chosen so that the effect of training could be seen while still keeping data as fresh as possible (that is, it is the smallest number larger than one). Finally, 30 epochs was found as the empirical point where synthetic data dev performance flattened out. Our test set comprises the last six subjects (7-12) of ICASSP data.

To decide on a deep learning model, we considered multiple possible inputs and outputs. One obvious input is to feed the time-domain signals directly into a network (we refer to this as a "time domain" approach). Because this is a signal processing problem which amounts to peak frequency detection, another obvious input is the signal FFTs ("frequency domain" approach).

A number of different outputs make sense. One example is a sequential output that outputs a "1" at PPG peaks and "0" elsewhere. For a more balanced dataset, another option is outputting a 50% duty-cycle squarewave whose rising edges are aligned with the PPG signal pulses. Either of these approaches result in a signal train of points where the PPG pulses occur. The heart rate can then be derived from these peak locations. This approach ended up failing because even small errors in the output could result in wildly inaccurate frequencies. For

example, each additional missplaced "1" could increase the heart rate by 7.5 BPM.

Another approach is to use the neural network to regress on a frequency directly. We found that regression worked poorly, which is consistent with other results examining the usefulness of neural networks for regression tasks.

The most promising results came from having the network output the frequency bin the signal belonged to as a category. By slowly increasing the difficulty of the dataset, we were able to find the limits of what frequency resolution could be reasonably classified. Surprisingly, even with very difficult datasets, some networks successfully classified $> 50\%$ of the evaluation windows to within 1 BPM out of 160 bins ranging from 40 BPM to 200 BPM.

Overall, we converged on frequency bin classification as the output and compared the time domain input approach to the frequency domain input approach. The models for each approach will be described in more detail.

Finally, after deciding on a model that would classify each evaluation window, the results for the classification windows for each subject were post-processed through an algorithm that stitches the output probabilities together into a single heart rate tracking curve. The algorithm iteratively updates a moving log-probability over all frequency bins starting at no prior `p = np.zeros(nout)`, then updating according to the probability distribution spit out by the deep learning model: `p = gaussian_filter1d(p, sigma=5)` `p += np.log(cur_proba[i] + eps)`, where `eps = 1 / nout`. At each window, the heart rate is chosen as as an average of the three bins closest to the argmax of `p`, weighted by the relative probabilities implied by `p`.

We compute three metrics for accuracy: classification accuracy, average absolute error (avAE, eq. 1), and standard deviation of absolute error (sdAE, eq. 2). Classification accuracy is used to assess performance on the synthetic dataset. avAE and sdAE are both used for ICASSP, and were chosen because they are common metrics found in the literature. Our main optimization metric is avAE, but the other metrics help guide performance.

$$\text{avAE} = \frac{1}{m}\sum_{i=1}^{m}|HR_{true}(i) - HR_{pred}(i)| \quad (1)$$

$$\text{sdAE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}|HR_{true}(i) - HR_{pred}(i)|^2} \quad (2)$$

A. Data Synthesis

Data synthesis occurred in successive stages. By slowly increasing the difficulty of the signals, it was easy to converge quickly on the approximate deep learning architectures needed.

1) Sine wave of random frequency/phase, 12.5 Sps.
2) Added Gaussian noise at -40dB to PPG.
3) Added MA as a sine wave at a different frequency and phase and a scale factor for MA influence on PPG signal.

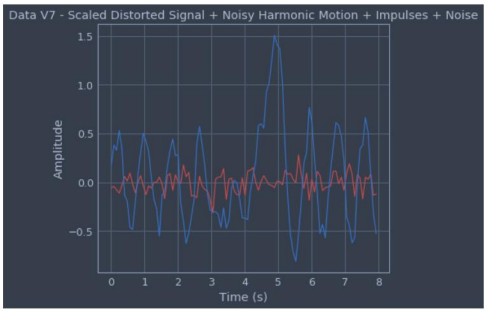

Fig. 4. Sample of synthetic data version 7.

4) Introduced a linear-varying envelope to the PPG signal.
5) Added harmonics to the motion artifact.
6) Added half- and double-frequencies to the PPG signal to distort it to match ICASSP signals.
7) Added sinc impulses to model "jumps" in the PPG signal not due to rapid motion.

An example final result is displayed in fig. 4. We now discuss a case study representative of the iteration process for elevating difficulty of the data synthesis model.

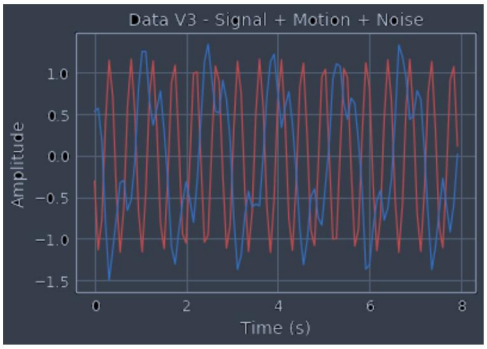

Fig. 5. Sample of synthetic data version 3. PPG (blue) and accelerometer (red) are plotted.

Fig. 5 shows a representative sample of the third version of the data synthesis model. If used to train a model, the ICASSP data will often fail at specific points such as the one shown in fig. 6. Notice that we mis-classify the frequency too high due to the high frequencies present in motion artifact. Since the model of fig. 5 assumes MA to be a pure sine wave, the model has been trained to see the high frequency signal in the PPG signal as the "true" PPG rather than as an artifact of MA.

A deeper look into exactly how MA interacts with the PPG signal shows the problem more clearly. In fig. 7, each acceleration vector is plotted as a point in 3D, colored by the PPG value at that same time. Notice the cluster of blue (low PPG value) and cluster of red (high PPG value). What the plot shows is that the MA appears to alternately raise, then lower the PPG signal during a single large period of MA rather than the smaller period of the 2nd harmonic, implying there is not merely a linear relationship between MA and PPG.

A new model is constructed to generate MA, where the impact of the MA occurs at a main cycle frequency, but the

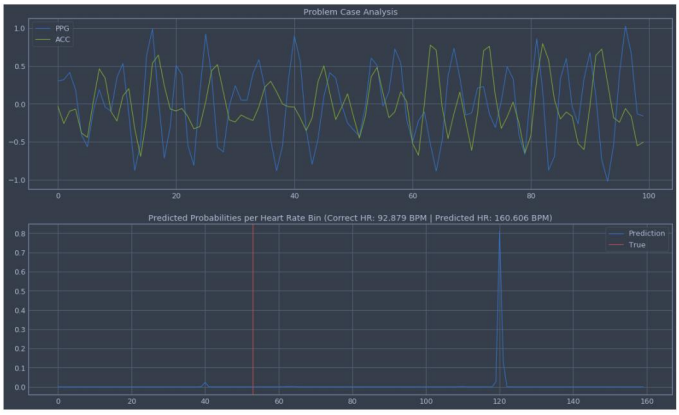

Fig. 6. A common problem occurring in the ICASSP data due to synthetic data - real data mismatch.

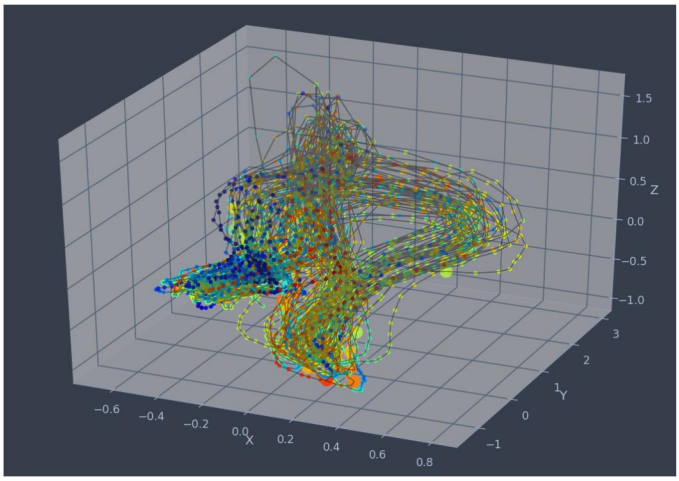

Fig. 7. Accelerometer XYZ parameterized plot colored by PPG value. Note that certain accelerometer values correlate with lower (blue) or higher (red) PPG values consistently. Heart beats occur at the enlarged dots.

MA itself contains 2nd harmonic components. A representative sample of the new synthetic data is shown in 8.

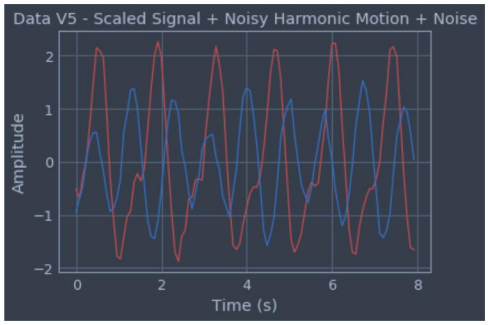

Fig. 8. Sample of synthetic data version 5. PPG (blue) and accelerometer (red) are plotted.

B. Time Domain Approach

The key idea in the time domain approach is to feed PPG and accelerometer signals for each 8-second evaluation window directly into an LSTM and output the heart rate as a category. Starting with very simple synthetic data (data version 1), it became obvious that the best approach would be at most a few layers deep, would include a sequence model, and might start with some convolutional layers. For more difficult data, convolutional layers helped less. LSTMs and more units performed better than simple RNNs.

The best model had PPG and accelerometer input (2×100) feed directly into a 512-unit LSTM, then to a 512-neuron dense layer, a dropout(0.5) layer, and finally a softmax layer with 160 outputs.

On the most challenging synthetic dataset, the network achieves a 70.1% accuracy on the synthetic dataset - recall that it is trying to predict within 1 BPM using only an 8-second long window. However, these results do not generalize well to the ICASSP data where there is a high avAE and sdAE of 10.2 and 11.2, respectively. See table I for more details and fig. 10 for qualitative and quantitative performance on the training and test ICASSP datasets.

C. Frequency Domain Approach

The key idea in the frequency domain approach is to feed FFTs of the PPG and accelerometer signals in each evaluation window to a neural network with categorical heart rate bins as outputs.

Inspired by [2], we chose to feed two 64-point FFTs per signal - one left justified `fft(sig[:64])` and one right justified `fft(sig[-64:])` to allow time-varying information to seep into the classification task. Our best network takes 4 FFTs (4×32) through three 512-node dense layers, a dropout(0.2) layer, and a final softmax layer with 160 outputs.

On the most challenging synthetic dataset, the network achieves 58.1% accuracy. These results generalize weakly to the ICASSP data. See table I for more details and fig. 10 for qualitative and quantitative performance on the training and test ICASSP datasets.

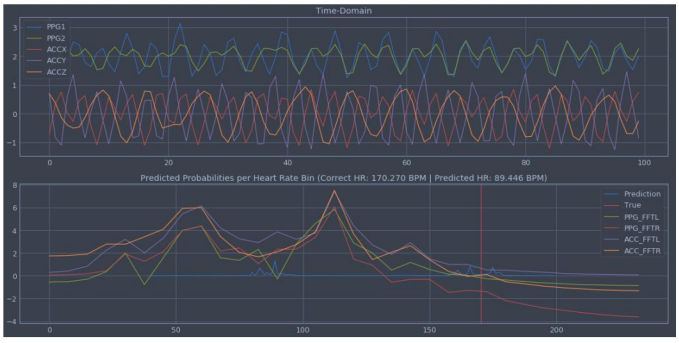

Fig. 9. FFTs of a problematic evaluation window. The true heart rate (vertical red line) and network output probabilities (blue) are plotted.

V. Results

Results from the time domain and frequency domain approaches are shown in figures 10 and 11, respectively. The top six plots were part of the training set, while the bottom

TABLE I
COMPARISON TO HIGH-PERFORMANCE ALGORITHMS AND LOW-POWER CIRCUITS

	Synth Train	Synth Dev	ICASSP avAE (7-12)	ICASSP sdAE (7-12)	Power
TROIKA [1]	-	-	2.45 BPM	1.76 BPM	-
TEMKO [2]	-	-	1.04 BPM	0.79 BPM	-
Compressive sampling [3]	-	-	10 BPM	-	172μW
Heart-beat locked loop [4]	-	-	5 BPM	-	38μW
This work Time-Domain	70.3%	70.1%	10.2 BPM	11.2 BPM	25μW*
This work Frequency-Domain	59.6%	58.1%	3.09 BPM	3.37 BPM	25μW*

*Estimated power from experiments with personal hardware.

six comprised the test set. The quantitative results avAE and sdAE are found for just the data in the bottom six plots after applying the post-processing step described in sec. IV and are reported in table I.

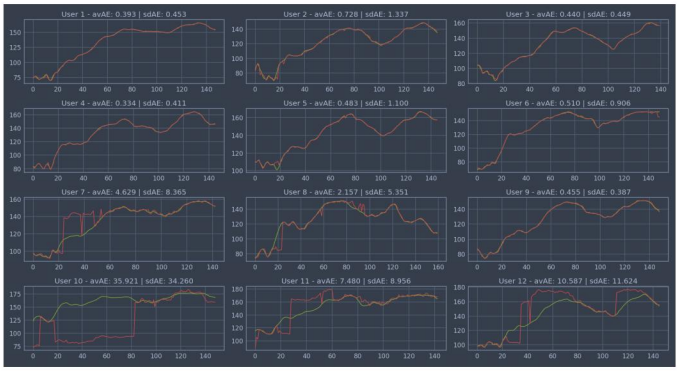

Fig. 10. Results of running the time domain algorithm on all ICASSP datasets. For the test set, avAE = 10.2; sdAE = 11.2.

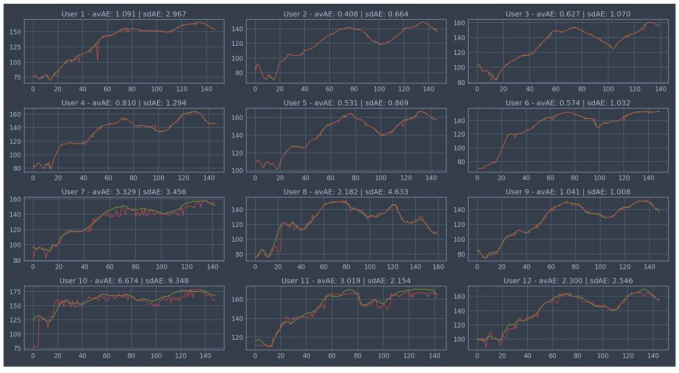

Fig. 11. Results of running the frequency domain algorithm on all ICASSP datasets. For the test set, avAE = 3.09; sdAE = 3.37.

VI. DISCUSSION AND FUTURE WORK

The time-domain approach is beneficial in that it requires no up-front processing compared to the frequency-domain approach. Looking at table I, it is clear that the time-domain approach more accurately classifies heart rate in the synthetic dataset. However, it is also more sensitive and fails often when run on the ICASSP data, as seen in fig. 10.

The frequency-domain approach requires FFT-preprocessing, but is a bit more robust on ICASSP. Its lower relative performance on the synthetic dataset is likely because the evaluation windows are short, causing the FFTs to have low frequency resolution. The essential difference is that the time-domain approach includes phase information.

More generally, it can be inferred that the time-domain approach is more sensitive to differences in the statistical properties between the synthetic dataset and the real data, but if the synthetic dataset can be improved sufficiently, it might be able to surpass the frequency domain performance. It is possible that a combined time domain and frequency domain approach can achieve both high precision and high reliability.

Compared to the state-of-the-art algorithms, such as [2], the results found here have about 3$\times$ larger error. However, part of that is due to the significantly reduced sample rate - 12.5 Hz instead of the original 125 Hz. Additionally, it is possible that improvements such as combining the time domain and frequency domain approaches and improving the data synthesis model to more closely match ICASSP data could further reduce errors to close the gap with state-of-the-art methods, despite the lower sample rate.

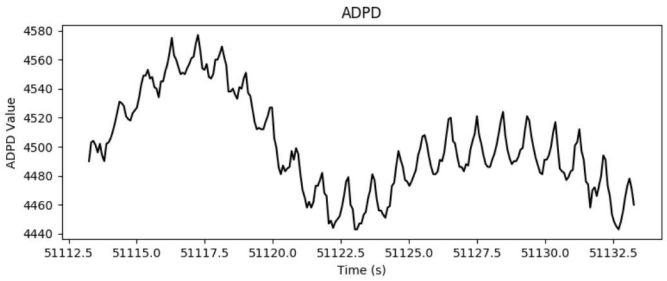

Fig. 12. Output of personal hardware with a sampling rate of 12.5 Hz and an approximate energy of 2μJ per sample.

Finally, we consider power consumption. Empirical testing on hardware suggests that the SNRs required for both of these algorithms implies an energy expenditure of approximately 2μJ/sample (fig. 12 demonstrates signal fidelity with these parameters). With these algorithms demonstrated at 12.5Hz, a system running at 25μW could achieve approximately 3 BPM avAE, surpassing [3] and [4] in power, accuracy, and robustness. However, this does not include the computational costs of performing multiple 512x512 matrix multiplies. When factoring the computational cost in, it is unlikely the system could achieve sub-100μW operation.

REFERENCES

[1] Z. Zhang, Z. Pi, B. Liu, TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise, IEEE Transactions on Biomedical Engineering, vol. 62, no. 2, pp. 522-531, February 2015, DOI: 10.1109/TBME.2014.2359372

[2] Temko, Andriy. "Estimation of heart rate from photoplethysmography during physical exercise using wiener filtering and the phase vocoder." Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE. IEEE, 2015.

[3] P. V. Rajesh et al., "22.4 A 172μW compressive sampling photoplethysmographic readout with embedded direct heart-rate and variability extraction from compressively sampled data," 2016 IEEE International Solid-State Circuits Conference (ISSCC), San Francisco, CA, 2016, pp. 386-387.

[4] Jang, Do-Hun, and SeongHwan Cho. "A 43.4 W photoplethysmogram-based heart-rate sensor using heart-beat-locked loop." Solid-State Circuits Conference-(ISSCC), 2018 IEEE International. IEEE, 2018.

[5] Sharma, Ajit, et al. "A Sub-60-A Multimodal Smart Biosensing SoC With ¿80-dB SNR, 35-A Photoplethysmography Signal Chain." IEEE Journal of Solid-State Circuits (2017).

[6] Abadi, Martn, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016).

[7] Chollet, François, et al. Keras. https://keras.io (2015).