REST: Robust and Efficient Neural Networks for Sleep Monitoring in the Wild

Rahul Duggal*[1], Scott Freitas*[1], Cao Xiao[2], Duen Horng (Polo) Chau[1], Jimeng Sun[1,3]
{rahulduggal,safreita,polo}@gatech.edu,cao.xiao@iqvia.com,jimeng@illinois.edu
Georgia Institute of Technology[1], IQVIA[2], University of Illinois Urbana-Champaign[3]

ABSTRACT

In recent years, significant attention has been devoted towards integrating deep learning technologies in the healthcare domain. However, to safely and practically deploy deep learning models for home health monitoring, two significant challenges must be addressed: the models should be (1) robust against noise; and (2) compact and energy-efficient. We propose REST, a new method that simultaneously tackles both issues via 1) *adversarial training* and controlling the Lipschitz constant of the neural network through *spectral regularization* while 2) enabling neural network compression through *sparsity regularization*. We demonstrate that REST produces highly-robust and efficient models that substantially outperform the original full-sized models in the presence of noise. For the sleep staging task over single-channel electroencephalogram (EEG), the REST model achieves a macro-F1 score of 0.67 vs. 0.39 achieved by a state-of-the-art model in the presence of Gaussian noise while obtaining 19× parameter reduction and 15× MFLOPS reduction on two large, real-world EEG datasets. By deploying these models to an Android application on a smartphone, we quantitatively observe that REST allows models to achieve up to 17× energy reduction and 9× faster inference. We open source the code repository with this paper: https://github.com/duggalrahul/REST.

CCS CONCEPTS

• **Applied computing** → **Health informatics**; • **Computing methodologies** → *Supervised learning by classification.*

KEYWORDS

deep learning, compression, adversarial, sleep staging

1 INTRODUCTION

As many as 70 million Americans suffer from sleep disorders that affects their daily functioning, long-term health and longevity. The long-term effects of sleep deprivation and sleep disorders include an increased risk of hypertension, diabetes, obesity, depression, heart attack, and stroke [1]. The cost of undiagnosed sleep apnea alone is estimated to exceed 100 billion in the US [28].

A central tool in identifying sleep disorders is the **hypnogram**—which documents the progression of sleep stages (**REM** stage, **Non-REM** stages **N1** to **N3**, and **Wake** stage) over an entire night (see Fig. 1, top). The process of acquiring a hypnogram from raw sensor data is called **sleep staging**, which is the focus of this work. Traditionally, to reliably obtain a hypnogram the patient has to undergo an overnight sleep study—called *polysomnography* (PSG)—at a sleep lab while wearing bio-sensors that measure physiological signals, which include electroencephalogram (EEG), eye movements (EOG), muscle activity or skeletal muscle activation (EMG), and heart rhythm (ECG). The PSG data is then analyzed by a trained sleep technician and a certified sleep doctor to produce a PSG report. The hypnogram plays an essential role in the PSG report, where it is used to derive many important metrics such as sleep efficiency and apnea index. Unfortunately, manually annotating this PSG is both costly and time consuming for the doctors. Recent research has proposed to alleviate these issues by automatically generating the hypnogram directly from the PSG using deep neural networks [6, 34]. However, the process of obtaining a PSG report is still *costly* and *invasive* to patients, reducing their participation, which ultimately leads to undiagnosed sleep disorders [33].

One promising direction to reduce undiagnosed sleep disorders is to enable sleep monitoring at the home using commercial wearables (e.g., Fitbit, Apple Watch, Emotiv) [21]. However, despite significant research advances, a recent study shows that wearables using a

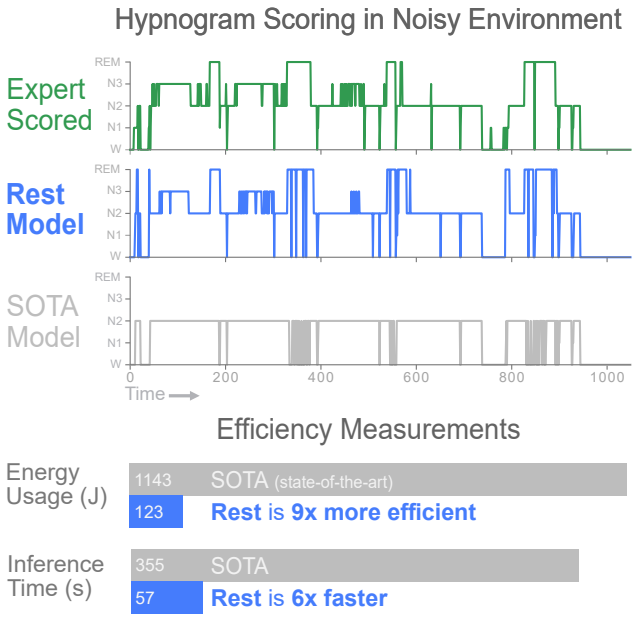

Figure 1: Top: we generate hypnograms for a patient in the SHHS test set. In the presence of Gaussian noise, our REST-generated hypnogram closely matches the contours of the expert-scored hypnogram. Hypnogram generated by a state-of-the-art (SOTA) model by Sors et al. [32] is considerably worse. Bottom: we measure energy consumed (in Joules) and inference time (in seconds) on a smartphone to score one night of EEG recordings. REST is 9X more energy efficient and 6X faster than the SOTA model.

* Both authors contributed equally to this research.

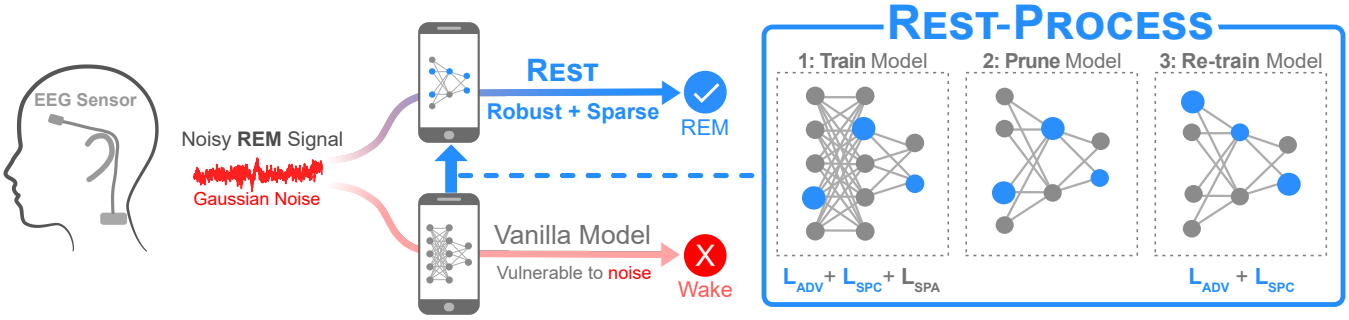

Figure 2: REST Overview: (from left) When a noisy EEG signal belonging to the REM (rapid eye movement) sleep stage enters a traditional neural network which is vulnerable to noise, it gets wrongly classified as a Wake sleep stage. On the other hand, the same signal is correctly classified as the REM sleep stage by the REST model which is both robust and sparse. (From right) REST is a three step process involving (1) training the model with adversarial training, spectral regularization and sparsity regularization (2) pruning the model and (3) re-training the compact model.

single sensor (e.g., single lead EEG) often have lower performance for sleep staging, indicating a large room for improvement [3].

1.1 Contributions

Our contributions are two-fold—(i) we identify emerging research challenges for the task of sleep monitoring in the wild; and (ii) we propose REST, a novel framework that addresses these issues.

I. New Research Challenges for Sleep Monitoring.

- **C1. Robustness to Noise.** We observe that state-of-the-art deep neural networks (DNN) are highly susceptible to environmental noise (Fig. 1, top). In the case of wearables, noise is a serious consideration since bioelectrical signal sensors (e.g., electroencephalogram "EEG", electrocardiogram "ECG") are commonly susceptible to *Gaussian* and *shot* noise, which can be introduced by electrical interferences (e.g., power-line) and user motions (e.g., muscle contraction, respiration) [5, 8, 10, 11]. This poses a need for noise-tolerant models. In this paper, we show that adversarial training and spectral regularization can impart significant noise robustness to sleep staging DNNs (see top of Fig 1).
- **C2. Energy and Computational Efficiency.** Mobile deep learning systems have traditionally offloaded compute intensive inference to cloud servers, requiring transfer of sensitive data and assumption of available Internet. However, this data uploading process is difficult for many healthcare scenarios because of—(1) **privacy**: individuals are often reluctant to share health information as they consider it highly sensitive; and (2) **accessibility:** real-time home monitoring is most needed in resource-poor environments where high-speed Internet may not be reliably available. Directly deploying a neural network to a mobile phone bypasses these issues. However, due to the constrained computation and energy budget of mobile devices, these models need to be fast in speed and parsimonious with their energy consumption.

II. Noise-robust and Efficient Sleep Monitoring. Having identified these two new research challenges, we propose **REST**, the first framework for developing noise-robust and efficient neural networks for home sleep monitoring (Fig. 2). Through REST, our major contributions include:

- "**R**obust and **E**fficient Neural Networks for **S**leep Moni**t**oring" By integrating a novel combination of three training objectives, REST endows a model with noise robustness through (1) *adversarial training* and (2) *spectral regularization*; and promotes energy and computational efficiency by enabling compression through (3) *sparsity regularization*.
- **Extensive evaluation** We benchmark the performance of REST against competitive baselines, on two real-world sleep staging EEG datasets—Sleep-EDF from Physionet and Sleep Heart Health Study (SHHS). We demonstrate that REST produces highly compact models that substantially outperform the original full-sized models in the presence of noise. REST models achieves a macro-F1 score of 0.67 vs. 0.39 for the state-of-the-art model in the presence of Gaussian noise, with 19× parameter and 15× MFLOPS reduction.
- **Real-world deployment.** We deploy a REST model onto a Pixel 2 smartphone through an Android application performing sleep staging. Our experiments reveal REST achieves 17× energy reduction and 9× faster inference on a smartphone, compared to uncompressed models.

2 RELATED WORK

In this section we discuss related work from three areas—(1) the task of sleep stage prediction, (2) robustness of deep neural networks and (3) compression of deep learning models.

2.1 Sleep-Stage Prediction

Sleep staging is the task of annotating a polysomnography (PSG) report into a hypnogram, where 30 second sleep intervals are annotated into one of five sleep stages (W, N1, N2, N3, REM). Recently, significant effort has been devoted towards automating this annotation process using deep learning [2, 6, 9, 29, 32, 39], to name a few. While there exists a large body of research in this area—two works in particular look at both single channel [6] and multi-channel [9] deep learning architectures for sleep stage prediction on EEG. In [6], the authors develop a deep learning architecture (SLEEPNET) for sleep stage prediction that achieves expert-level accuracy on

EEG data. In [9], the authors develop a multi-modal deep learning architecture for sleep stage prediction that achieves state-of-the-art accuracy. As we demonstrate later in this paper (Section 4.5), these sleep staging models are frequently susceptible to noise and suffer a large performance drop in its presence (see Figure 1). In addition, these DNNs are often overparameterized (Section 4.6), making deployment to mobile devices and wearables difficult. Through REST, we address these limitations and develop noise robust and efficient neural networks for edge computing.

2.2 Noise & Adversarial Robustness

Adversarial robustness seeks to ensure that the output of a neural network remains unchanged under a bounded perturbation of the input; or in other words, prevent an adveresary from maliciously perturbing the data to fool a neural network. Adversarial deep learning was popularized by [17], where they showed it was possible to alter the class prediction of deep neural network models by carefully crafting an adversarially perturbed input. Since then, research suggests a strong link between adversarial robustness and noise robustness [15, 20, 35]. In particular, [15] found that by performing adversarial training on a deep neural network, it becomes robust to many forms of noise (e.g., Gaussian, blur, shot, etc.). In contrast, they found that training a model on Gaussian augmented data led to models that were less robust to adversarial perturbations. We build upon this finding of adversarial robustness as a proxy for noise robustness and improve upon it through the use of spectral regularization; while simultaneously compressing the model to a fraction of its original size for mobile devices.

2.3 Model Compression

Model compression aims to learn a reduced representation of the weights that parameterize a neural network; shrinking the computational requirements for memory, floating point operations (FLOPS), inference time and energy. Broadly, prior art can be classified into four directions—pruning [19], quantization [31], low rank approximation [37] and knowledge distillation [22]. For REST, we focus on structured (channel) pruning thanks to its performance benefits (speedup, FLOP reduction) and ease of deployment with regular hardware. In structured channel pruning, the idea is to assign a measure of importance to each filter of a convolutional neural network (CNN) and achieve desired sparsity by pruning the least important ones. Prior work demonstrates several ways to estimate filter importance—magnitude of weights [24], structured sparsity regularization [36], regularization on activation scaling factors [26], filter similarity [13] and discriminative power of filters [40]. Recently there has been an attempt to bridge the area of model compression with adversarial robustness through connection pruning [18] and quantization [25]. Different from previous work, REST aims to compress a model by pruning whole filters while imparting noise tolerance through adversarial training and spectral regularization. REST can be further compressed through quantization [25].

3 REST: NOISE-ROBUST & EFFICIENT MODELS

REST is a new method that simultaneously compresses a neural network while developing both noise and adversarial robustness.

3.1 Overview

Our main idea is to enable REST to endow models with these properties by integrating three careful modifications of the traditional training loss function. (1) The *adversarial training* term, which builds noise robustness by training on adversarial examples (Section 3.2); (2) the *spectral regularization* term, which adds to the noise robustness by constraining the Lipschitz constant of the neural network (Section 3.3); and (3) the sparsity regularization term that helps to identify important neurons and enables compression (Section 3.4). Throughout the paper, we follow standard notation and use capital bold letters for matrices (e.g., $\mathbf{A}$), lower-case bold letters for vectors (e.g., $\mathbf{a}$).

3.2 Adversarial Training

The goal of adversarial training is to generate noise robustness by exposing the neural network to adversarially perturbed inputs during the training process. Given a neural network $f(\mathbf{X}; \mathbf{W})$ with input $\mathbf{X}$, weights $\mathbf{W}$ and corresponding loss function $L(f(\mathbf{X}; \mathbf{W}), y)$, adversarial training aims at solving the following min-max problem:

$$\min_{\mathbf{W}} \mathop{\mathbb{E}}_{\mathbf{X}, y \sim D} \max_{\delta \in S} L(f(\mathbf{X} + \delta; \mathbf{W}), y) \tag{1}$$

Here D is the unperturbed dataset consisting of the clean EEG signals $\mathbf{X} \in \mathbb{R}^{K_{in} \times K_L}$ (K_{in} is the number of channels and K_L is the length of the signal) along with their corresponding label y. The inner maximization problem in (1) embodies the goal of the adversary—that is, produce adversarially perturbed inputs (i.e., $\mathbf{X} + \delta$) that maximize the loss function L. On the other hand, the outer minimization term aims to build robustness by countering the adversary through minimizing the expected loss on perturbed inputs.

Maximizing the inner loss term in (1) is equivalent to finding the adversarial signal $\mathbf{X}_p = \mathbf{X} + \delta$ that maximally alters the loss function L within some bounded perturbation $\delta \in S$. Here S is the set of allowable perturbations. Several choices exist for such an adversary. For REST, we use the iterative Projected Gradient Descent (PGD) adversary since it's one of the strongest first order attacks [27]. Its operation is described below in Equation 2.

$$\mathbf{X}_p^{(t+1)} = \mathbf{X}_p^{(t)} + \Pi_\tau \left[\epsilon \cdot sign \left(\nabla_{\mathbf{X}_p^{(t)}} L \left(f(\mathbf{X}_p^{(t)}; \mathbf{W}), y \right) \right) \right] \tag{2}$$

Here $\mathbf{X}_p^{(0)} = \mathbf{X}$ and at every step t, the previous perturbed input $\mathbf{X}_p^{(t-1)}$ is modified with the sign of the gradient of the loss, multiplied by ϵ (controls attack strength). Π_τ is a function that clips the input at the positions where it exceeds the predefined L_∞ bound τ. Finally, after n_{iter} iterations we have the REST adversarial training term L_{adv} in Equation 3.

$$L_{adv} = L(f(\mathbf{X}_p^{(n_{iter})}; \mathbf{W}), y) \tag{3}$$

3.3 Spectral Regularizer

The second term in the objective function is the spectral regularization term, which aims to constrain the change in output of a neural network for some change in input. The intuition is to suppress the amplification of noise as it passes through the successive layers of

Algorithm 1: Noise Robust & Efficient Neural Network Training (REST)

Input: Model weights $\mathbf{W}$, EEG signal $\mathbf{X}$ and label y from dataset $\mathbf{D}$, spectral regularization λ_o, sparsity regularization λ_∂, learning rate α, perturbation strength ϵ, maximum PGD iterations n_{iter} and model sparsity s

Output: Noise robust, compressed neural network

(1) **Train the full model with REST loss L_R:**

for *epoch = 1 to N* **do**
  for *minibatch* $B \subset D$ **do**
    for $X \in B$ **do**
      $\mathbf{X}_p^{(1)} = \mathbf{X}$
      for *k=1 to* n_{iter} **do**
        $\mathbf{X}_p^{(k+1)} = \mathbf{X}_p^{(k)} + \Pi_\tau(\epsilon \cdot sign(\nabla_{\mathbf{X}_p^{(k)}} L(f(\mathbf{X}_p^{(k)}; \mathbf{W}), y)))$
      $\mathbf{W}_{grad} \leftarrow \mathop{\mathbb{E}}_{\mathbf{X}, y \sim D} |\nabla_{\mathbf{W}} L_R(\mathbf{X}_p, y; \mathbf{W})|$
      where $L_R = \underbrace{L(f(\mathbf{X}_p; \mathbf{W}), y)}_{\text{adversarial training}} + \lambda_o \underbrace{\sum_{\text{layer l=1}}^{\tilde{N}} \|(\mathbf{W}^{(l)})^T \mathbf{W}^{(l)} - \mathbf{I}\|_2}_{\text{spectral regularization}} + \lambda_\partial \underbrace{\sum_{\text{layer l=1}}^{\tilde{N}} \|y^{(l)}\|_1}_{\text{sparsity regularization}}$
      $\mathbf{W} \leftarrow \mathbf{W} - \alpha \cdot \mathbf{W}_{grad}$

(2) **Prune the trained model:**

Globally prune filters from $\mathbf{W}$ having smallest y values until $\frac{n_f(\mathbf{W}')}{n_f(\mathbf{W})} \leq s$. Constrain layerwise sparsity so $\frac{n_f(\mathbf{W}'^{(l)})}{n_f(\mathbf{W}^{(l)})} \geq 0.1$.

(3) **Re-train the pruned model:**

Retrain compressed network $f(\mathbf{X}; \mathbf{W}')$ using *adversarial training* and *spectral regularization* (no sparsity regularization).

a neural network. In this section we show that an effective way to achieve this is via constraining the Lipschitz constant of each layer's weights.

For a real valued function $f : \mathbb{R} \rightarrow \mathbb{R}$ the Lipschitz constant is a positive real value C such that $|f(x_1) - f(x_2)| \leq C|x_1 - x_2|$. If $C > 1$ then the change in input is magnified through the function f. For a neural net, this can lead to input noise amplification. On the other hand, if $C < 1$ then the noise amplification effect is diminished. This can have the unintended consequence of reducing the discriminative capability of a neural net. Therefore our goal is to set the Lipschitz constant $C = 1$. The Lipschitz constant for the l^{th} fully connected layer parameterized by the weight matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{K_{in} \times K_{out}}$ is equivalent to its spectral norm $\rho(\mathbf{W}^{(l)})$ [12]. Here the spectral norm of a matrix $\mathbf{W}$ is the square root of the largest singular value of $\mathbf{W}^T\mathbf{W}$. The spectral norm of a 1-D convolutional layer parameterized by the tensor $\mathbf{W}^{(l)} \in \mathbb{R}^{K_{out} \times K_{in} \times K_l}$ can be realized by reshaping it to a matrix $\mathbf{W}^{(l)} = \mathbb{R}^{K_{out} \times (K_{in} K_l)}$ and then computing the largest singular value.

A neural network of N layers can be viewed as a function $f(\cdot)$ composed of N sub-functions $f(x) = f_1(\cdot) \circ f_2(\cdot) \circ \ldots f_N(x)$. A loose upper bound for the Lipschitz constant of f is the product of Lipschitz constants of individual layers or $\rho(f) \leq \prod_{i=1}^{N} \rho(f_i)$ [12]. The overall Lipschitz constant can grow exponentially if the spectral norm of each layer is greater than 1. On the contrary, it could go to 0 if spectral norm of each layer is between 0 and 1. Thus the ideal case arises when the spectral norm for each layer equals 1. This can be achieved in several ways [12, 14, 38], however, one effective way is to encourage orthonormality in the columns of the weight matrix $\mathbf{W}$ through the minimization of $\|\mathbf{W}^T\mathbf{W} - \mathbf{I}\|$ where $\mathbf{I}$ is the identity matrix. This additional loss term helps regulate the singular values and bring them close to 1. Thus we incorporate the following spectral regularization term into our loss objective, where λ_o is a hyperparameter controlling the strength of the spectral regularization.

$$L_{Spectral} = \lambda_o \sum_{i=1}^{\tilde{N}} \|(\mathbf{W}^{(i)})^T \mathbf{W}^{(i)} - \mathbf{I}\|_2 \tag{4}$$

3.4 Sparsity Regularizer & REST Loss Function

The third term of the REST objective function consists of the sparsity regularizer. With this term, we aim to learn the important filters in the neural network. Once these are determined, the original neural network can be pruned to the desired level of sparsity.

The incoming weights for filter i in the l^{th} fully connected (or 1-D convolutional) layer can be specified as $\mathbf{W}_{i,:}^{(l)} \in \mathbb{R}^{K_{in}}$ (or $\mathbf{W}_{i,:,:}^{(l)} \in \mathbb{R}^{K_{in} \times K_L}$). We introduce a per filter multiplicand $y_i^{(l)}$ that scales the output activation of the i^{th} neuron in layer l. By controlling the value of this multiplicand, we realize the importance of the neuron. In particular, zeroing it amounts to dropping the entire filter. Note that the L_0 norm on the multiplicand vector $\|y^{(l)}\|_0$, where $y^{(l)} \in \mathbb{R}^{K_{out}}$, can naturally satisfy the sparsity objective since it counts the number of non zero entries in a vector. However since the L_0 norm is a nondifferentiable function, we use the L_1 norm as a surrogate [23, 26, 36] which is amenable to backpropagation through its subgradient.

To realize the per filter multiplicand $y_i^{(l)}$, we leverage the per filter multiplier within the batch normalization layer [26]. In most

modern networks, a batchnorm layer immediately follows the convolutional/linear layers and implements the following operation.

$$B_i^{(l)} = \frac{\mathbf{A}^{(l)} - \mu_i^{(l)}}{\sigma_i^{(l)}} \gamma_i^{(l)} + \beta_i^{(l)} \tag{5}$$

Here $\mathbf{A}_i^{(l)}$ denotes output activation of filter i in layer l while $\mathbf{B}_i^{(l)}$ denotes its transformation through batchnorm layer l; $\mu^{(l)} \in R^{K_{out}}$, $\sigma^{(l)} \in R^{K_{out}}$ denote the mini-batch mean and standard deviation for layer l's activations; and $\gamma^{(l)} \in R^{K_{out}}$ and $\beta^{(l)} \in R^{K_{out}}$ are learnable parameters. Our sparsity regularization is defined on $\gamma^{(l)}$ as below, where λ_∂ is a hyperparameter controlling the strength of sparsity regularization.

$$L_{Sparsity} = \lambda_\partial \sum_{i=1}^{N} \|\gamma^{(l)}\|_1 \tag{6}$$

The sparsity regularization term (6) promotes learning a subset of important filters while training the model. Compression then amounts to globally pruning filters with the smallest value of multiplicands in (5) to achieve the desired model compression. Pruning typically causes a large drop in accuracy. Once the pruned model is identified, we fine-tune it via retraining.

Now that we have discussed each component of REST, we present the full loss function in (7) and the training process in Algorithm 1. A pictorial overview of the process can be seen in Figure 2.

$$\begin{aligned} L_R = \underbrace{L(f(\mathbf{X}_p; \mathbf{W}), y)}_{\text{adversarial training}} + \lambda_o \underbrace{\sum_{i=1}^{N} \|(\mathbf{W}^{(i)})^T \mathbf{W}^{(i)} - \mathbf{I}\|_2}_{\text{spectral regularization}} \\ + \underbrace{\lambda_\partial \sum_{i=1}^{N} \|\gamma^{(l)}\|_1}_{\text{sparsity regularization}} \end{aligned} \tag{7}$$

4 EXPERIMENTS

We compare the efficacy of REST neural networks to four baseline models (Section 4.2) on two publicly available EEG datasets—Sleep-EDF from Physionet [16] and Sleep Heart Health Study (SHHS) [30]. Our evaluation focuses on two broad directions—**noise robustness** and **model efficiency**. Noise robustness compares the efficacy of each model when EEG data is corrupted with three types of noise: *adversarial*, *Gaussian* and *shot*. Model efficiency compares both static (e.g., model size, floating point operations) and dynamic measurements (e.g., inference time, energy consumption). For dynamic measurements which depend on device hardware, we deploy each model to a Pixel 2 smartphone.

4.1 Datasets

Our evaluation uses two real-world sleep staging EEG datasets.

- **Sleep-EDF**: This dataset consists of data from two studies—age effect in healthy subjects (SC) and Temazepam effects on sleep (ST). Following [34], we use whole-night polysomnographic sleep recordings on 40 healthy subjects (one night per patient) from SC. It is important to note that the SC study is conducted in the subject's homes, not a sleep center and hence this dataset is inherently noisy. However, the sensing environment is still relatively controlled since sleep doctors visited the patient's home to setup the wearable EEG sensors. After obtaining the data, the recordings are manually classified into one of eight classes (W, N1, N2, N3, N4, REM, MOVEMENT, UNKNOWN); we follow the steps in [34] and merge stages N3 and N4 into a single N3 stage and exclude MOVEMENT and UNKNOWN stages to match the five stages of sleep according to the American Academy of Sleep Medicine (AASM) [4]. Each single channel EEG recording of 30 seconds corresponds to a vector of dimension 1×3000. Similar to [32], while scoring at time i, we include EEG recordings from times $i-3, i-2, i-1, i$. Thus we expand the EEG vector by concatenating the previous three time steps to create a vector of size 1×12000. After pre-processing the data, our dataset consists of 42,191 EEG recordings, each described by a 12,000 length vector and assigned a sleep stage label from Wake, N1, N2, N3 and REM using the Fpz-Cz EEG sensor (see Table 1 for sleep stage breakdown). Following standard practice [34], we divide the dataset on a *per-patient, whole-night* basis, using 80 % for training, 10 % for validation, and 10 % for testing. That is, a single patient is recorded for one night and can only be in one of the three sets (training, validation, testing). The final number of EEG recordings in their respective splits are 34,820, 5345 and 3908. While the number of recordings appear to differ from the 80-10-10 ratio, this is because the data is split over the total number of *patients*, where each patient is monitored for a time period of variable length (9 hours ± few minutes.)
- **Sleep Heart Health Study (SHHS)**: The Sleep Heart Health Study consists of two rounds of polysomnographic recordings (SHHS-1 and SHHS-2) sampled at 125 Hz in a sleep center environment. Following [32], we use only the first round (SHHS-1) containing 5,793 polysomnographic records over two channels (C4-A1 and C3-A2). Recordings are manually classified into one of six classes (W, N1, N2, N3, N4 and REM). As suggested in [4], we merge N3 and N4 stages into a single N3 stage (see Table 1 for sleep stage breakdown). We use 100 distinct patients randomly sampled from the original dataset (one night per patient). Similar to [32], we look at three previous time steps in order to score the EEG recording at the current time step. This amounts to concatenating the current EEG recording of size 1×3750 (equal to 125 Hz $\times$ 30 Hz) to generate an EEG recording of size 1×15000. After this pre-processing, our dataset consists of 100,065 EEG recordings, each described by a 15,000 length vector and assigned a sleep stage label from the same 5 classes using the Fpz-Cz EEG sensor. We use the same 80-10-10 data split as in Sleep-EDF, resulting

Dataset	W	N1	N2	N3(N4)	REM	Total
Sleep-EDF	8,168	2,804	17,799	5,703	7,717	42,191
SHHS	28,854	3,377	41,246	13,409	13,179	100,065

Table 1: Dataset summary outlining the number of 30 second EEG recordings belonging to each sleep stage class.

in 79,940 EEG recordings for training, 9999 for validation, and 10,126 for testing.

4.2 Model Architecture and Configurations

We use the sleep staging CNN architecture proposed by [32], since it achieves state-of-the-art accuracy for sleep stage classification using single channel EEG. We implement all models in PyTorch 0.4. For training and evaluation, we use a server equipped with an Intel Xeon E5-2690 CPU, 250GB RAM and 8 Nvidia Titan Xp GPUs. Mobile device measurements use a Pixel 2 smartphone with an Android application running Tensorflow Lite[1]. With [32] as the architecture for all baselines below, we compare the following 6 configurations:

(1) **Sors** [32]: Baseline neural network model trained on unperturbed data. This model contains 12 1-D convolutional layers followed by 2 fully connected layers and achieves state-of-the-art performance on sleep staging using single channel EEG.
(2) **Liu** [26]: We train on unperturbed data and compress the Sors model using sparsity regularization as proposed in [26].
(3) **Blanco** [7]: We use same setup from Liu above. During test time, the noisy test input is filtered using a bandpass filter with cutoff 0.5Hz-40Hz This technique is commonly used for removing noise in EEG analysis [7].
(4) **Ford** [15]: We train and compress the Sors model with sparsity regularization on input data perturbed by Gaussian noise. Gaussian training parameter $c_\partial = 0.2$ controls the perturbation strength during training; identified through a line search in Section 4.4.
(5) **Rest (A)**: Our compressed Sors model obtained through adversarial training and sparsity regularization. We use the hyperparameters: $\epsilon = 10, n_{iter} = 5/10$ (SHHS/Sleep-EDF), where ϵ is a *key* variable controlling the strength of adversarial perturbation during training. The optimal ϵ value is determined through a line search described in Section 4.4.
(6) **Rest (A+S)**: Our compressed Sors model obtained through adversarial training, spectral and sparsity regularization. We set the spectral regularization parameter $\lambda_\partial = 3 \times 10^{-3}$ and sparsity regularization parameter $\lambda_\partial = 10^{-5}$ based on a grid search in Section 4.4.

All models are trained for 30 epochs using SGD. The initial learning rate is set to 0.1 and multiplied by 0.1 at epochs 10 and 20; the weight decay is set to 0.0002. All compressed models use the same compression method, consisting of weight pruning followed by model re-training. The sparsity regularization parameter $\lambda_\partial = 10^{-5}$ is identified through a grid search with λ_o (after determining ϵ through a line search). Detailed analysis of the hyperparameter selection for ϵ, λ_o and λ_∂ can be found in Section 4.4. Finally, we set a high sparsity level $s = 0.8$ (80% neurons from the original networks were pruned) after observation that the models are overparametrized for the task of sleep stage classification.

[1]TensorFlow Lite: https://www.tensorflow.org/lite

4.3 Evaluation Metrics

Noise robustness metrics To study the noise robustness of each model configuration, we evaluate macro-F1 score in the presence of three types of noise: adversarial, Gaussian and shot. We select macro-F1 since it is a standard metric for evaluating classification performance in imbalanced datasets. Adversarial noise is defined at three strength levels through $\epsilon = 2/6/12$ in Equation 2; Gaussian noise at three levels through $c_\partial = 0.1/0.2/0.3$ in Equation 8; and shot noise at three levels through $c_s = 5000/2500/1000$ in Equation 9. These parameter values are chosen based on prior work [20, 27] and empirical observation. For evaluating robustness to adversarial noise, we assume the white box setting where the attacker has access to model weights. The formulation for Gaussian and shot noise is in Equation 8 and 9, respectively.

$$\mathbf{X}_{gauss} = \mathbf{X} + N(0, c_g \cdot \sigma_{train}) \tag{8}$$

In Equation 8, σ_{train} is the standard deviation of the training data and N is the normal distribution. The noise strength—low, medium and high—corresponds to $c_\partial = 0.1/0.2/0.3$.

$$\begin{aligned} \mathbf{X}_{norm} &= \frac{\mathbf{X} - x_{min}}{x_{max} - x_{min}} \\ \mathbf{X}' &= clip_{0,1}\ \frac{Poisson(\mathbf{X}_{norm}.c_s)}{c_s} \\ \mathbf{X}_{shot} &= \mathbf{X}'.(x_{max} - x_{min}) + x_{min} \end{aligned} \tag{9}$$

In Equation 9, x_{min}, x_{max} denote the minimum and maximum values in the training data; and $clip_{0,1}$ is a function that projects the input to the range [0,1].

Model efficiency metrics To evaluate the efficiency of each model configuration, we use the following measures:

- **Parameter Reduction**: Memory consumed (in KB) for storing the weights of a model.
- **Floating point operations (FLOPS)**: Number of multiply and add operations performed by the model in one forward pass. Measurement units are Mega (10^6).
- **Inference Time**: Average time taken (in seconds) to score one night of EEG data. We assume a night consists of 9 hours and amounts to 1,080 EEG recordings (each of 30 seconds). This is measured on a Pixel 2 smartphone.
- **Energy Consumption**: Average energy consumed by a model (in Joules) to score one night of EEG data on a Pixel 2 smartphone. To measure consumed energy, we implement an infinite inference loop over EEG recordings until the battery level drops from 100% down to 85%. For each unit percent drop (i.e., 15 levels), we log the number of iterations N_i performed by the model. Given that a standard Pixel 2 battery can deliver 2700 mAh at 3.85 Volts, we use the following conversion to estimate energy consumed E (in Joules) for a unit percent drop in battery level $E = \frac{2700}{1000} \times 3600 \times 3.85$. The total energy for inferencing over an entire night of EEG recordings is then calculated as $\frac{E}{N_i} \times 1080$ where N_i is the number of inferences made in the unit battery drop interval. We average this for every unit battery percentage

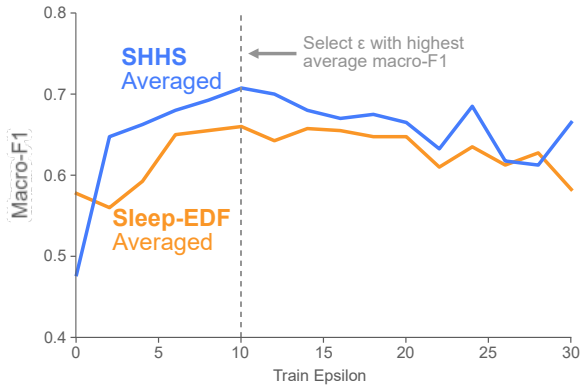

Figure 3: Line search results for ϵ on Sleep-EDF and SHHS datasets. We select ϵ=10, since it provides the best average macro-F1 score on both datasets.

drop from 100% to 85% (i.e., 15 intervals) to calculate the average energy consumption

4.4 Hyperparameter Selection

Optimal hyper-parameter selection is crucial for obtaining good performance with both baseline and REST models. We systematically conduct a series of line and grid searches to determine ideal values of ϵ, c_∂, λ_o and λ_∂ using the validation sets.

Selecting ϵ This parameter controls the perturbation strength of adversarial training in Equation 2. Correctly setting this parameter is critical since a small ϵ value will have no effect on noise robustness, while too high a value will lead to poor benign accuracy. We follow standard procedure and determine the optimal ϵ on a per-dataset basis [27], conducting a line search across $\epsilon \in$ [0,30] in steps of 2. For each value of ϵ we measure benign and adversarial validation macro-F1 score, where adversarial macro-F1 is an average of three strength levels: low (ϵ=2), medium (ϵ=6) and high (ϵ=12). We then select the ϵ with highest macro-F1 score averaged across the benign and adversarial macro-F1. Line search results are shown in Figure 3; we select ϵ = 10 for both dataset since it's the value with highest average macro-F1.

Selecting c_∂ This parameter controls the noise perturbation strength of Gaussian training in Equation 8. Similar to ϵ, we determine c_∂ on a per-dataset basis, conducting a line search across c_∂ values: 0.1 (low), 0.2 (medium) and 0.3 (high). Based on results from Table 2, we select c_∂=0.2 for both datasets since it provides the best average macro-F1 score while minimizing the drop in benign accuracy.

Selecting λ_o and λ_∂ These parameters determine the strength of spectral and sparsity regularization in Equation 7. We determine the best value for λ_o and λ_∂ through a grid search across the following parameter values $\lambda_o = [0.001, 0.003, 0.005]$ and $\lambda_\partial = [1E-04, 1E-05]$. Based on results from Table 3, we select $\lambda_o = 0.003$ and $\lambda_\partial = 1E-05$. Since these are model dependent parameters, we calculate them once on the Sleep-EDF dataset and re-use them for SHHS.

			Guassian F1			
	c_∂	Benign F1	Low	Med	High	Average F1
EDF	0.1	0.75	0.76	0.7	0.5	0.68
	0.2	0.7	0.72	0.75	0.64	**0.70**
	0.3	0.67	0.68	0.71	0.75	0.7025
SHHS	0.1	0.69	0.74	0.45	0.21	0.52
	0.2	0.68	0.69	0.68	0.43	**0.62**
	0.3	0.55	0.57	0.65	0.74	0.63

Table 2: Line search results for identifying optimal c_∂ on Sleep-EDF and SHHS datasets. Macro-F1 is abbreviated F1 in table; average macro-F1 is the mean of all macro-F1 scores. We select c_∂=0.2 for both datasets as it represents a good trade-off between benign and Gaussian macro-F1.

			Adversarial F1			
λ_o	λ_∂	Benign F1	Low	Med	High	Avg. F1
0.001	1E-04	0.73	0.66	0.65	0.61	0.66
0.003	1E-04	0.72	0.64	0.63	0.59	0.65
0.005	1E-04	0.72	0.65	0.64	0.62	0.66
0.001	1E-05	0.73	0.66	0.65	0.62	0.67
0.003	1E-05	0.73	0.67	0.66	0.62	**0.67**
0.005	1E-05	0.73	0.64	0.64	0.62	0.66

Table 3: Grid search results for λ_o and λ_∂ on Sleep-EDF dataset. Macro-F1 is abbreviated as F1 in table; average macro-F1 is the mean of all macro-F1 scores. We select λ_o and λ_∂ with highest average macro-F1 score.

4.5 Noise Robustness

To evaluate noise robustness, we ask the following questions—(1) what is the impact of REST on model accuracy with and without noise in the data? and (2) how does REST training compare to baseline methods of benign training, Gaussian training and noise filtering? In answering these questions, we analyze noise robustness of models at three scales: (i) meta-level macro-F1 scores; (ii) meso-level confusion matrix heatmaps; and (iii) granular-level single-patient hypnograms.

I. Meta analysis: Macro-F1 Scores In Table 4, we present a high-level overview of model performance through macro-F1 scores on three types and strength levels of noise corruption. The Macro-F1 scores and standard deviation are reported by averaging over three runs for each model and noise level. We identify multiple key insights as described below:

(1) **REST Outperforms Across All Types of Noise** As demonstrated by the higher macro-F1 scores, REST outperforms all baseline methods in the presence of noise. In addition, REST has a low standard deviation, indicating model performance is not dependent on weight initialization.

Data	Method	Compress	No noise	Adversarial			Gaussian			Shot		
				Low	Med	High	Low	Med	High	Low	Med	High
Sleep-EDF	Sors [32]	✗	0.67 ± 0.02	0.57 ± 0.02	0.51 ± 0.04	0.19 ± 0.06	0.66 ± 0.03	0.60 ± 0.03	0.39 ± 0.08	0.58 ± 0.04	0.42 ± 0.08	0.11 ± 0.03
	Liu [26]	✓	**0.69** ± 0.02	0.52 ± 0.07	0.41 ± 0.07	0.09 ± 0.02	0.67 ± 0.02	0.53 ± 0.02	0.28 ± 0.04	0.52 ± 0.03	0.31 ± 0.04	0.06 ± 0.01
	Blanco [7]	✓	0.68 ± 0.01	0.51 ± 0.06	0.40 ± 0.06	0.09 ± 0.02	0.65 ± 0.02	0.54 ± 0.04	0.31 ± 0.10	0.53 ± 0.04	0.34 ± 0.09	0.08 ± 0.02
	Ford [15]	✓	0.64 ± 0.01	0.59 ± 0.01	0.60 ± 0.02	0.31 ± 0.08	0.65 ± 0.01	0.67 ± 0.02	0.57 ± 0.03	0.67 ± 0.02	0.60 ± 0.02	0.10 ± 0.01
	REST (A)	✓	0.66 ± 0.02	0.64 ± 0.02	0.64 ± 0.02	0.61 ± 0.02	0.66 ± 0.02	0.67 ± 0.01	0.66 ± 0.01	0.67 ± 0.01	0.66 ± 0.01	**0.42** ± 0.06
	REST (A+S)	✓	**0.69** ± 0.01	**0.67** ± 0.02	**0.66** ± 0.01	**0.61** ± 0.03	**0.69** ± 0.01	**0.68** ± 0.01	**0.67** ± 0.02	**0.68** ± 0.01	**0.67** ± 0.02	**0.42** ± 0.08
SHHS	Sors [32]	✗	**0.78** ± 0.01	0.62 ± 0.03	0.46 ± 0.03	0.33 ± 0.00	0.64 ± 0.03	0.43 ± 0.02	0.35 ± 0.04	0.69 ± 0.02	0.59 ± 0.03	0.45 ± 0.01
	Liu [26]	✓	0.77 ± 0.01	0.61 ± 0.02	0.49 ± 0.04	0.34 ± 0.03	0.66 ± 0.05	0.45 ± 0.05	0.34 ± 0.04	0.70 ± 0.04	0.62 ± 0.04	0.47 ± 0.05
	Blanco [7]	✓	0.77 ± 0.01	0.60 ± 0.03	0.47 ± 0.04	0.33 ± 0.02	0.64 ± 0.07	0.43 ± 0.05	0.34 ± 0.04	0.67 ± 0.06	0.59 ± 0.05	0.46 ± 0.04
	Ford [15]	✓	0.62 ± 0.02	0.59 ± 0.01	0.62 ± 0.00	0.59 ± 0.05	0.66 ± 0.00	0.75 ± 0.04	0.47 ± 0.10	0.65 ± 0.00	0.68 ± 0.01	0.74 ± 0.04
	REST (A)	✓	0.70 ± 0.01	0.68 ± 0.00	0.70 ± 0.01	0.67 ± 0.01	0.72 ± 0.01	0.76 ± 0.01	0.58 ± 0.03	0.72 ± 0.01	0.74 ± 0.01	0.76 ± 0.01
	REST (A+S)	✓	0.72 ± 0.01	**0.69** ± 0.01	**0.70** ± 0.01	**0.69** ± 0.02	**0.74** ± 0.01	**0.77** ± 0.01	**0.62** ± 0.03	**0.73** ± 0.01	**0.75** ± 0.01	**0.78** ± 0.00

Table 4: Meta Analysis: Comparison of macro-F1 scores achieved by each model. The models are evaluated on Sleep-EDF and SHHS datasets with three types and strengths of noise corruption. We bold the compressed model with the best performance (averaged over 3 runs) and report the standard deviation of each model next to the macro-F1 score. REST performs better in *all* noise test measurements.

(2) **Spectral Regularization Improves Performance** REST ($A+S$) consistently improves upon REST (A), indicating the usefulness of spectral regularization towards enhancing noise robustness by constraining the Lipschitz constant.

(3) **SHHS Performance Better Than Sleep-EDF** Performance is generally better on the SHHS dataset compared to Sleep-EDF. One possible explanation is due to the SHHS dataset being less noisy in comparison to the Sleep-EDF dataset. This stems from the fact that the SHHS study was performed in the hospital setting while Sleep-EDF was undertaken in the home setting.

(4) **Benign & Adversarial Accuracy Trade-off** Contrary to the traditional trade-off between benign and adversarial accuracy, REST performance matches Liu in the no noise setting on sleep-EDF. This is likely attributable to the noise in the Sleep-EDF dataset, which was collected in the home setting. On the SHHS dataset, the Liu model outperforms REST in the no noise setting, where data is captured in the less noise prone hospital setting. Due to this, REST models are best positioned for use in noisy environments (e.g., at home); while traditional models are more effective in controlled environments (e.g., sleep labs).

II. Meso Analysis: Per-class Performance We visualize and identify class-wise trends using confusion matrix heatmaps (Fig. 4). Each confusion matrix describes a model's performance for a given level of noise (or no noise). A model that is performing well should have a dark diagonal and light off-diagonal. We normalize the rows of each confusion matrix to accurately represent class predictions in an imbalanced dataset. When a matrix diagonal has a value of 1 (dark blue, or dark green) the model predicts every example correctly; the opposite occurs at 0 (white). Analyzing Figure 4, we identify the following key insights:

(1) **REST Performs Well Across All Classes** REST accurately predicts each sleep stage (W, N1, N2, N3, REM) across multiple types of noise (Fig. 4, bottom 3 rows), as evidenced by the dark diagonal. In comparison, each baseline method has considerable performance degradation (light diagonal) in the presence of noise. This is particularly evident on the Sleep-EDF dataset (left half) where data is collected in the noisier home environment.

(2) **N1 Class Difficult to Predict** When no noise is present (Fig. 4, top row), each method performs well as evidenced by the dark diagonal, except on the N1 sleep stage class. This performance drop is likely due to the limited number of N1 examples in the datasets (see Table 1).

(3) **Increased Misclassification Towards "Wake" Class** On the Sleep-EDF dataset, shot and adversarial noise cause the baseline models to mispredict classes as Wake. One possible explanation is that the models misinterpret the additive noise as evidence for the wake class which has characteristically large fluctuations.

III. Granular Analysis: Single-patient Hypnograms We want to more deeply understand how our REST models counteract noise at the hypnogram level. Therefore, we select a test set patient from the SHHS dataset, and generate and visualize the patient's overnight hypnograms using the Sors and REST models on three levels of Gaussian noise corruption (Figure 5). Each of these hypnograms is compared to a trained technicians hypnogram (expert scored in Fig. 5), representing the ground-truth. We inspect a few more test set patients using the above approach, and identify multiple key representative insights:

(1) **Noisy Environments Require Robust Models** As data noise increases, Sors performance degrades. This begins at the low noise level, further accelerates in the medium level and reaches

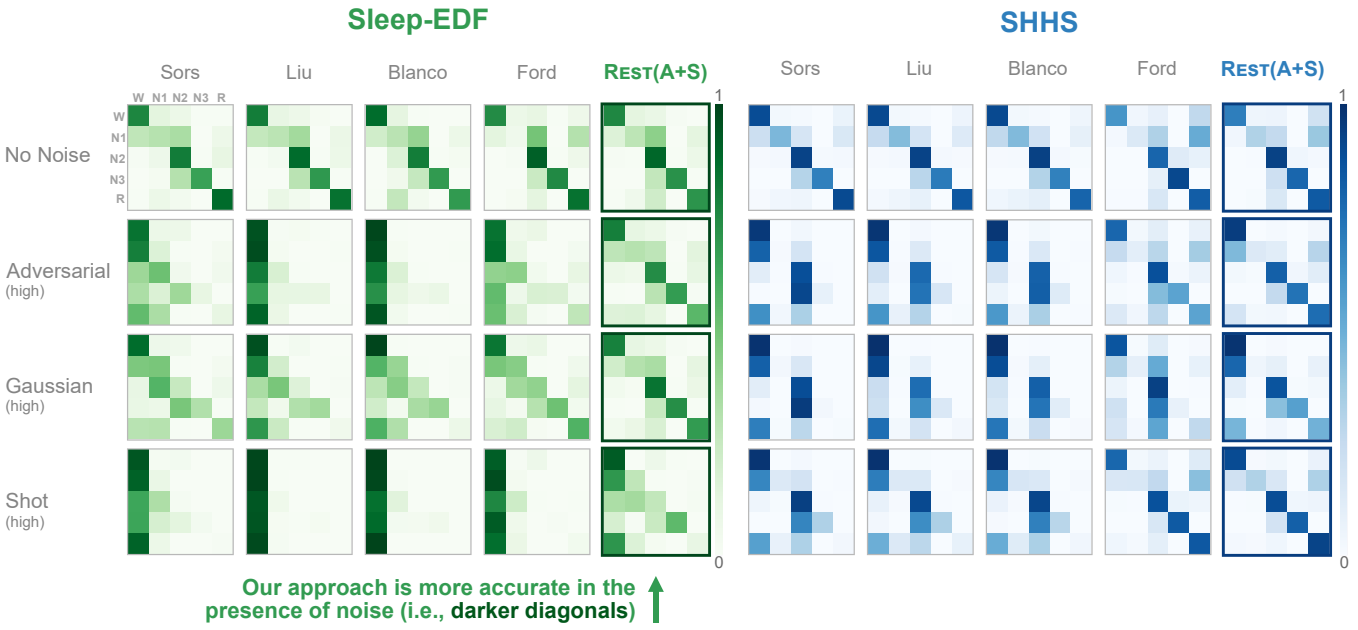

Figure 4: Meso Analysis: Class-wise comparison of model predictions. The models are evaluated over the SHHS test set perturbed with different noise types. In each confusion matrix, rows are ground-truth classes while columns are predicted classes. The intensity of a cell is obtained by normalizing the score with respect to the class membership. When a cell has a value of 1 (dark blue, or dark green) the model predicts every example correctly, the opposite occurs at 0 (white). A model that is performing well would have a dark diagonal and light off-diagonal. REST has the darkest cells along the diagonal on both datasets.

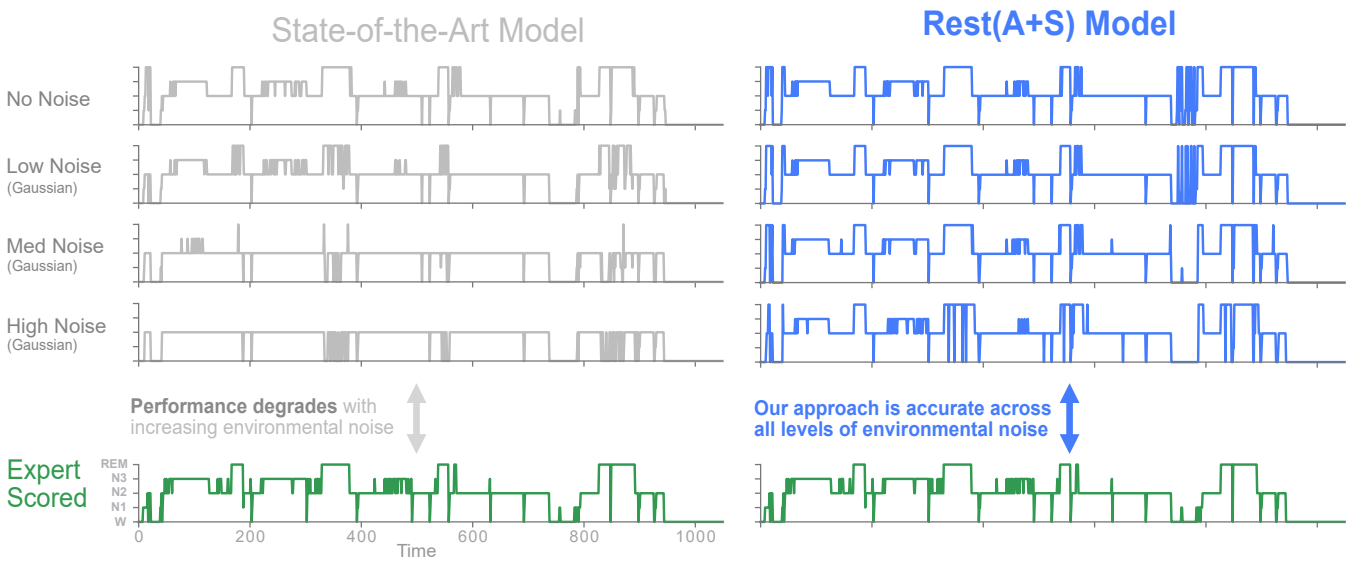

Figure 5: Granular Analysis: Comparison of the overnight hypnograms obtained for a patient in the SHHS test set. The hypnograms are generated using the Sors (left) and REST (right) models in the presence of increasing strengths of Gaussian noise. When no noise is present (top row), both models perform well, closely matching the ground truth (bottom row). However, with increasing noise, Sors performance rapidly degrades, while REST continues to generate accurate hypnograms.

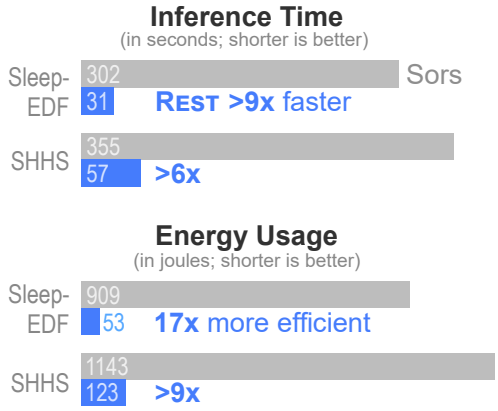

Figure 6: Time and energy consumption for scoring a single night of EEG recordings. REST(A+S) is significantly faster and more energy efficient than the state-of-the-art Sors model. Evaluations were done on a Pixel 2 smartphone.

Data	Model	Size (KB)	MFlops
Sleep-EDF	Sors [32]	8,896	1451
	Liu [26]	**440**	127
	Blanco [7]	440	127
	Ford [15]	448	144
	REST (A)	464	98
	REST (A+S)	449	**94**
SHHS	Sors [32]	8,996	1815
	Liu [26]	**464**	211
	Blanco [7]	464	211
	Ford [15]	478	170
	REST (A)	476	160
	REST (A+S)	496	**142**

Table 5: Comparison on model size and the FLOPS required to score a single night of EEG recordings. REST models are significantly smaller and comparable in size/compute to baselines.

nearly zero at the high level. In contrast, REST effectively handles all levels of noise, generating an accurate hypnogram at even the highest level.

(2) **Low Noise Environments Give Good Performance** In the no noise setting (top row) both the Sors and REST models generate accurate hypnograms, closely matching the contours of expert scoring (bottom).

4.6 Model Efficiency

We measure model efficiency along two dimensions—(1) *static metrics*: amount of memory required to store weights in memory and FLOPS; and (2) *dynamic metrics*: inference time and energy consumption. For dynamic measurements that depend on device hardware, we deploy each model to a Pixel 2 smartphone.

Analyzing Static Metrics: Memory & Flops Table 5 describes the size (in KB) and computational requirements (in MFlops) of each model. We identify the following key insights:

(1) **REST Models Require Fewest FLOPS** On both datasets, REST requires the least number of FLOPS.

(2) **REST Models are Small** REST models are also smaller (or comparable) to baseline compressed models while achieving significantly better noise robustness.

(3) **Model Efficiency and Noise Robustness** Combining the insights from Section 4.5 and the above, we observe that REST models have significantly better noise robustness while maintaining a competitive memory footprint. This suggests that robustness is more dependent on the the training process, rather than model capacity.

Analyzing Dynamic Metrics: Inference Time & Energy In Figure 6, we benchmark the inference time and energy consumption of a Sors and REST model deployed on a Pixel 2 smartphone using Tensorflow Lite. We identify the following insights:

(1) **REST Models Run Faster** When deployed, REST runs 9× and 6× faster than the uncompressed model on the two datasets.

(2) **REST Models are Energy Efficient** REST models also consume 17× and 9× less energy than an uncompressed model on the Sleep-EDF and SHHS datasets, respectively.

(3) **Enabling Sleep Staging for Edge Computing** The above benefits demonstrate that model compression effectively translates into faster inference and a reduction in energy consumption. These benefits are crucial for deploying on the edge.

5 CONCLUSION

We identified two key challenges in developing deep neural networks for sleep monitoring in the home environment—*robustness to noise* and *efficiency*. We proposed to solve these challenges through REST—a new method that simultaneously tackles both issues. For the sleep staging task over electroencephalogram (EEG), REST trains models that achieve up to 19× parameter reduction and 15× MFLOPS reduction with an increase of up to 0.36 in macro-F-1 score in the presence of noise. By deploying these models to a smartphone, we demonstrate that REST achieves up to 17× energy reduction and 9× faster inference.

6 ACKNOWLEDGMENTS

This work was in part supported by the NSF award IIS-1418511, CCF-1533768, IIS-1838042, CNS-1704701, IIS-1563816; GRFP (DGE-1650044); and the National Institute of Health award NIH R01 1R01NS107291-01 and R56HL138415.

REFERENCES

[1] Bruce M Altevogt, Harvey R Colten, et al. 2006. *Sleep disorders and sleep deprivation: an unmet public health problem.* National Academies Press.

[2] F. Andreotti, H. Phan, N. Cooray, C. Lo, M. T. M. Hu, and M. De Vos. 2018. Multichannel Sleep Stage Classification and Transfer Learning using Convolutional Neural Networks. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).* 171–174. https://doi.org/10.1109/EMBC.2018.8512214

[3] Z Beattie, A Pantelopoulos, A Ghoreyshi, Y Oyang, A Statan, and C Heneghan. 2017. 0068 ESTIMATION OF SLEEP STAGES USING CARDIAC AND ACCELEROMETER DATA FROM A WRIST-WORN DEVICE. *Sleep* 40, suppl_1 (April 2017), A26–A26.

[4] Richard B Berry, Rita Brooks, Charlene E Gamaldo, Susan M Harding, Carole L Marcus, Bradley V Vaughn, et al. 2012. The AASM manual for the scoring of sleep and associated events. *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine* 176 (2012).

[5] Vikrant Bhateja, Shabana Urooj, Rishendra Verma, and Rini Mehrotra. 2013. A novel approach for suppression of powerline interference and impulse noise in ECG signals. In *IMPACT-2013*. IEEE, 103–107.

[6] Siddharth Biswal, Joshua Kulas, Haoqi Sun, Balaji Goparaju, M. Brandon Westover, Matt T. Bianchi, and Jimeng Sun. 2017. SLEEPNET: Automated Sleep Staging System via Deep Learning. *CoRR* abs/1707.08262 (2017). arXiv:1707.08262 http://arxiv.org/abs/1707.08262

[7] S Blanco, S Kochen, OA Rosso, and P Salgado. 1997. Applying time-frequency analysis to seizure EEG activity. *IEEE Engineering in medicine and biology magazine* 16, 1 (1997), 64–71.

[8] Manuel Blanco-Velasco, Binwei Weng, and Kenneth E Barner. 2008. ECG signal denoising and baseline wander correction based on the empirical mode decomposition. *Computers in biology and medicine* 38, 1 (2008), 1–13.

[9] Stanislas Chambon, Mathieu N Galtier, Pierrick J Arnal, Gilles Wainrib, and Alexandre Gramfort. 2018. A deep learning architecture for temporal sleep stage classification using multivariate and multimodal time series. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 26, 4 (2018), 758–769.

[10] Kang-Ming Chang and Shing-Hong Liu. 2011. Gaussian noise filtering from ECG by Wiener filter and ensemble empirical mode decomposition. *Journal of Signal Processing Systems* 64, 2 (2011), 249–264.

[11] Yongjian Chen, Masatake Akutagawa, Takahiro Emoto, and Yohsuke Kinouchi. 2010. The removal of EMG in EEG by neural networks. *Physiological measurement* 31, 12 (2010), 1567.

[12] Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 854–863.

[13] Rahul Duggal, Cao Xiao, Richard Vuduc, and Jimeng Sun. 2019. CUP: Cluster Pruning for Compressing Deep Neural Networks. arXiv:arXiv:1911.08630

[14] Farzan Farnia, Jesse M Zhang, and David Tse. 2018. Generalizable Adversarial Training via Spectral Normalization. *arXiv preprint arXiv:1811.07457* (2018).

[15] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. 2019. Adversarial Examples Are a Natural Consequence of Test Error in Noise. *CoRR* abs/1901.10513 (2019). arXiv:1901.10513 http://arxiv.org/abs/1901.10513

[16] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation* 101, 23 (2000), e215–e220.

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[18] Yiwen Guo, Chao Zhang, Changshui Zhang, and Yurong Chen. 2018. Sparse dnns with improved adversarial robustness. In *Advances in neural information processing systems*. 242–251.

[19] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*. 1135–1143.

[20] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261* (2019).

[21] André Henriksen, Martin Haugen Mikalsen, Ashenafi Zebene Woldaregay, Miroslav Muzny, Gunnar Hartvigsen, Laila Arnesdatter Hopstock, and Sameline Grimsgaard. 2018. Using fitness trackers and smartwatches to measure physical activity in research: analysis of consumer wrist-worn wearables. *Journal of medical Internet research* 20, 3 (2018), e110.

[22] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[23] Vadim Lebedev and Victor Lempitsky. 2016. Fast convnets using group-wise brain damage. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2554–2564.

[24] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* (2016).

[25] Ji Lin, Chuang Gan, and Song Han. 2019. Defensive quantization: When efficiency meets robustness. *arXiv preprint arXiv:1904.08444* (2019).

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*. 2736–2744.

[27] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[28] American Academy of Sleep Medicine et al. 2016. Economic Burden of Undiagnosed Sleep Apnea in US is Nearly $150 Billion per Year. *Published on the American Academy of Sleep Medicine's official website, on August* 8 (2016).

[29] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos. 2019. Joint Classification and Prediction CNN Framework for Automatic Sleep Stage Classification. *IEEE Transactions on Biomedical Engineering* 66, 5 (May 2019), 1285–1296. https://doi.org/10.1109/TBME.2018.2872652

[30] Stuart F Quan, Barbara V Howard, Conrad Iber, James P Kiley, F Javier Nieto, George T O'Connor, David M Rapoport, Susan Redline, John Robbins, Jonathan M Samet, et al. 1997. The sleep heart health study: design, rationale, and methods. *Sleep* 20, 12 (1997), 1077–1085.

[31] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*. Springer, 525–542.

[32] Arnaud Sors, Stéphane Bonnet, Sébastien Mirek, Laurent Vercueil, and Jean-François Payen. 2018. A convolutional neural network for sleep stage scoring from raw single-channel EEG. *Biomedical Signal Processing and Control* 42 (2018), 107 – 114. https://doi.org/10.1016/j.bspc.2017.12.001

[33] Annette Sterr, James K Ebajemito, Kaare B Mikkelsen, Maria A Bonmati-Carrion, Nayantara Santhi, Ciro Della Monica, Lucinda Grainger, Giuseppe Atzori, Victoria Revell, Stefan Debener, et al. 2018. Sleep EEG derived from behind-the-ear electrodes (cEEGrid) compared to standard polysomnography: A proof of concept study. *Frontiers in human neuroscience* 12 (2018), 452.

[34] Akara Supratak, Hao Dong, Chao Wu, and Yike Guo. 2017. DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 25, 11 (2017), 1998–2008.

[35] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2018. Robustness may be at odds with accuracy. *stat* 1050 (2018), 11.

[36] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*. 2074–2082.

[37] Jian Xue, Jinyu Li, and Yifan Gong. 2013. Restructuring of deep neural network acoustic models with singular value decomposition.. In *Interspeech*. 2365–2369.

[38] Yuichi Yoshida and Takeru Miyato. 2017. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941* (2017).

[39] Mingmin Zhao, Shichao Yue, Dina Katabi, Tommi S Jaakkola, and Matt T Bianchi. 2017. Learning Sleep Stages from Radio Signals: A Conditional Adversarial Architecture. 70 (2017), 4100–4109.

[40] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. 2018. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*. 875–886.