A Semi-Supervised Wasserstein Generative Adversarial Network for Classifying Driving Fatigue from EEG signals

Sharaj Panwar[1], Paul Rad[2], John Quarles[3], Edward Golob[4], Yufei Huang[1]
[1]Department of Electrical and Computer Engineering, [2]Department of Information Systems, [3]Department of Computer Science, [4]Department of Psychology, University of Texas at San Antonio, TX 78249

***Abstract*— The goal of this project is to use deep learning to predict the cognitive states of drivers from electroencephalograph (EEG) signals. To address the challenge posed by limited labeled training samples, a semi-supervised Wasserstein Generative Adversarial Networks (sWGAN) is proposed. The proposed sWGAN includes a classifier with the shared architecture with the discriminator in GAN and its loss function enables the augmentation of limited training samples with generated EEG samples during training, thus resulting in improved classification performance. The several modeling challenges including frequency artifacts and training instability are also considered. The test results on predicting the alert and drowsy states from a simulated driving experiment demonstrate improved prediction performance and training stability over the baseline semi-supervised GAN and a convolutional neural network model.**

Keywords—Wasserstein Generative adversarial network, Deconvolution with bi-linear weigh initialization, EEG, Driving data, convolutional neural network (CNN).

I. INTRODUCTION

Predicting whether a driver is in an alert or drowsy state based on electroencephalograph (EEG) signals has been an active area of research [1, 2]. Predicting driver state is extremely challenging because driving requires multitasking, which in turn is supported by complex neural activities. Advanced machine learning and deep learning approaches may be particularly useful for solving this problem. However, collecting large, labeled training samples of driver cognitive states, and for EEG-related tasks in general, is often not feasible due to complex experimental setups, discomfort from prolonged wearing of EEG sensory, and difficulties in inducing and correctly labeling fatigue states. Therefore, an approach that can augment limited samples with authentic simulated samples is attractive.

Generative Adversarial Networks (GANs) are deep generative models that can learn to generate samples from the data distribution[3]. GANs have been recently applied in a variety of applications such as images and time series data such as EEG[4]. The basic GAN framework consists of a generator and a discriminator, both of which are deep neural networks. The generator tries to fool the discriminator by producing realistic fake samples, and the discriminator tries to distinguish between the real and fake samples[3]. The generator and discriminator are optimized together according to a two-player game. The GAN framework has been used to solve many problems such as domain adaption[3-4], photo editing [7] and semi-supervised learning [6-11], which are enabled by architectural changes and modified loss functions.

The semi-supervised GAN is an extension of GAN to include a classifier that predicts the class labels of the data. Intuitively, this approach augments the training samples with GAN generated samples. Once trained, no sample generation is needed and the classifier exploits the superior ability of GAN in learning data representations to achieve improved performance. However, existing semi-supervised GAN (SGAN) model training does not attempt to use the label identify of generated samples to influence the training of the classifier[14]. Implementing the GAN framework on EEG signals is challenging due to training instability and frequency artifacts in the generator [4]. Artifacts can severely impair GAN convergence if the discriminator learns a poor solution and rejects generated samples based on these artifacts. This would collapse the semi-supervised GAN training and create a poor quality learning data representation, which in turn would result in a poor classification performance.

To address these issues, we propose an EEG specific semi-supervised Wasserstein GAN (sWGAN) architecture where the generator has two-step upsampling that combines bi-cubical interpolation upsampling and a deconvolution layer with Bi-linear weight initialization. This approach helps prevent frequency artifacts and stabilizes GAN training. We also propose an improved loss function that attempts to predict the labels of generated samples to be used for classifier training, which therefore improves the classifier performance. The proposed sWGAN framework is trained on two sets of XB-Driving EEG data, one including data from a single channel and the second consisting of data from 64 channels. In both datasets the trained semi-supervised GAN classifier improved prediction of driver fatigue in comparison to convolutional neural network models.

II. METHODS

A. Mathematical Background on Generative Adversarial Networks

The basic GAN framework consists of two opposing networks trying to outplay each other[3]. Given the real data distribution $\mathbb{P}_r$ and the generated data distribution $\mathbb{P}_\theta$, the first network, the discriminator D, is trained to distinguish between real x_r and fake x_f data. The second network, the generator G, takes a latent noise variable z from a random distribution $\mathbb{P}_z$ as input and tries to generate fake samples x_f to fool the D. This results in a minimax game, in which the G is forced by the D to produce ever better samples. However, the vanilla GANs suffers heavily from training instability and is restricted to low resolution images. Much advancement in regards to stability and the quality of the generated images has been made with different regularization methods [15]. The Wasserstein GAN (WGAN) with a gradient penalty loss function has been shown to effectively improve training stability and convergence. The Wasserstein GAN minimizes the Wasserstein distance between the real and fake data distributions

$$\widetilde{W}(\mathbb{P}_r \mathbb{P}_\theta) = E_{x_r \sim \mathbb{P}_r}[D(x_r)] - E_{x_f \sim \mathbb{P}_\theta}[D(x_f)] \qquad (1)$$

where the generator maximizes $E_{x_f \sim \mathbb{P}_\theta}[D(x_f)]$, whereas the discriminator minimizes $\widetilde{W}$. The WGAN requires $D(x)$ to be K-Lipschitz [16] and this can be achieved by clipping the

weights of the discriminator to an interval [−c, c] (WGAN clip). A more elegant solution is achieved by adding a gradient penalty term to the WGAN loss (1)

$$P_2(\mathbb{P}_{\hat{x}}) = \lambda \cdot E_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2] \quad (2)$$

where λ is a hyper parameter controlling the trade-off between the WGAN loss (1) and gradient penalty (2), $\hat{x}$ denotes the data points lying on a straight line between $\mathbb{P}_r$ and $\mathbb{P}_\theta$.

The GAN framework can be extended for semi-supervised learning, which includes an additional classification task as shown in [14]. To achieve this, the classifier and discriminator share lower layers. This results in two sets of outputs; one discriminator branch and one classifier branch. In addition, the classifier also adds one additional class y_f for fake samples. Adding a classifier branch does not change the basic GAN framework and discriminator still plays a vital role in the GAN training. The discriminator branch computes the minimax loss also called the GAN loss i.e. likelihood of real $E[D(x)]$ and fake samples $E[\mathrm{D}(\mathrm{G}(\mathrm{z}))]$. The classification branch computes the likelihood of classes y_r for the real samples in the data in addition to a GAN loss class [14]. The classifier branch will still have the GAN loss term while predicting fake classes since we do not have labels to assign for predicted classes y_f. The classifier losses for real and fake samples can be expressed as

$$\min_{y_r} H[p(y_r|x_r),\ C], \quad \text{with } x_r \sim \mathbb{P}_x \quad (3)$$

$$\min_{y_f} H[p(y_f|x_f),\ C], \quad \text{with } x_f \sim \mathbb{P}_\theta \quad (4)$$

Adding an extra GAN loss term helps the training process, however, the absence of target fake labels weaken this method. One reason is that the loss for predicted fake labels y_f does not contribute to the total loss used for optimization. Secondly, this part of lothe ss adversely affects training when the generated samples gets close to real samples. Since the total loss for generated samples is still only the GAN loss while for the real samples it is the addition of GAN loss and loss calculated in predicting real classes y_r there is an unbalanced mini-max optimization that prevents the algorithm from reaching its full potential.

In this work we propose an improvement to the above semi-supervised GAN (SGAN) frameworks to efficiently classify EEG data. The fake class labels y_f for the generated samples ($x_f \sim \mathbb{P}_\theta$) are now predicted from the model itself as $C(G(z))$, which is optimized simultaneously with the real data samples ($x_r \sim \mathbb{P}_r$) training. This enables us to remove the GAN loss class from the classifier's loss branch. The architecture of the proposed semi-supervised Wasserstein GAN (sWGAN) is shown in Figure 1(a). Now the classifier loss will try to decrease the entropy of real samples, as demonstrated in (5) and Figure 1(b), and generated samples as shown in (6) and Figure 1(c).

$$\min_{y_r} H[p(y_r|x_r,\ C)], \quad \text{with } x_r \sim \mathbb{P}_x \quad (5)$$

$$\min_{C(G(z))} H\big[p\big(C(G(z))|x_f,\ C\big)\big], \quad \text{with } z \sim \mathbb{P}_z \ \&\ x_f \sim \mathbb{P}_\theta \quad (6)$$

Since the WGAN loss with gradient penalty has proven to be an efficient loss function for GANs, we use this for our discriminator loss branch.

B. Generator architecture for EEG time-series data

The GAN generator takes a random noise vector as input and generates samples by gradually upsampling the input dimension. The upsampling method plays an important role in determining the quality of generated samples, which in turn will impact overall GAN training. Deconvolution, also known as transposed convolution or fractionally strided convolution, has been effectively used for increasing the resolution in image related tasks and can also be used here for upsampling in the GAN generator [17]. In the deconvolution operation, based on the upsampling factor, input values of zero are inserted between consecutive input values, and a trainable kernel is applied to perform a convolution operation (Figure 2). The number of zeros to be inserted are calculated by subtracting 1 from the upsampling factor. However, when the kernel stride of deconvolution is smaller than the kernel size there is an uneven overlap in the output regions [18]. The uneven overlap positions cause the checkerboard pattern in the resulting image.

Ideally, the deep learning model should learn the weights carefully for unevenly overlapping positions so that checkerboard pattern at the output can addressed. However, it is a balancing act to learn the weights for both even and uneven overlap positions. The problem can be partially addressed by choosing a kernel size that is divisible by deconvolution stride, but evenly overlapping regions too can still learn kernels that cause similar artifacts. A significant presence of such patterns in time series signals produce huge frequency artifacts, which can force the discriminator to learn a trivial solution to reject generated samples based on these

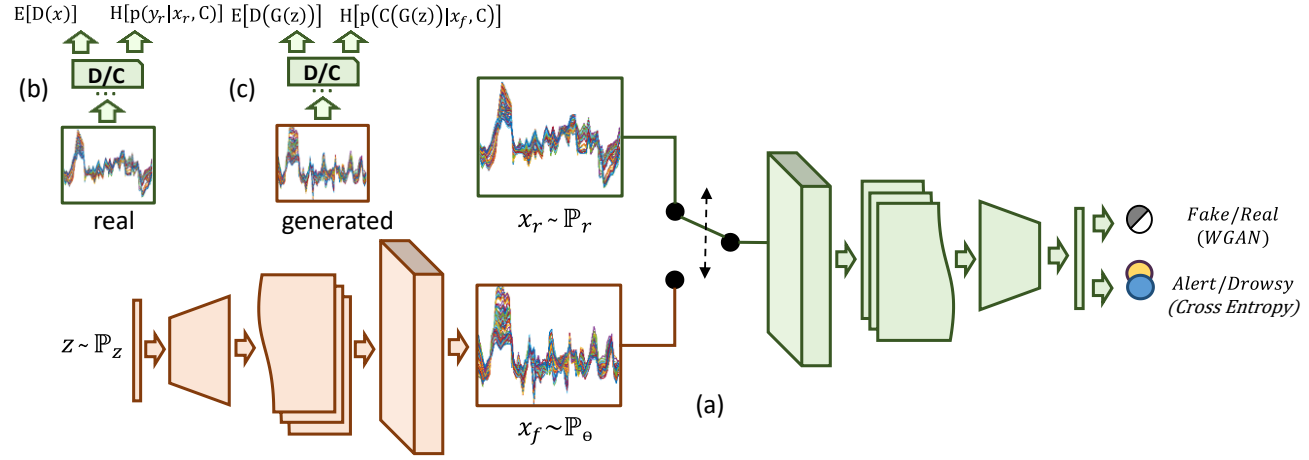

Figure 1 a): The illustration of the proposed sWGAN framework. It has two outputs at the discriminator architecture, a classifier and a discriminator. The discriminator discriminates real from generated data based on the WGAN loss, and the classifier predicts the class labels based on the cross entropy loss. b) The illustration of discriminator and classifier loss for real samples. c) The illustration of discriminator and classifier loss for generated samples.

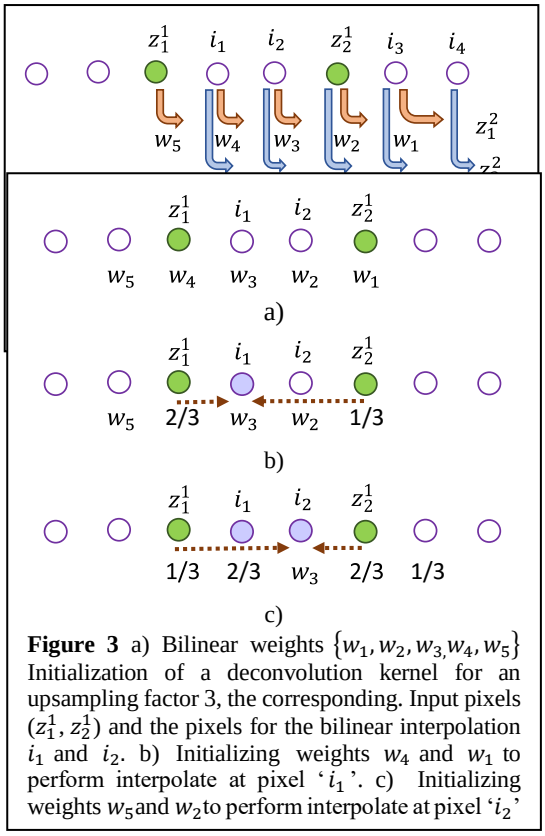

Figure 3 a) Bilinear weights $\{w_1, w_2, w_3, w_4, w_5\}$ Initialization of a deconvolution kernel for an upsampling factor 3, the corresponding. Input pixels (z_1^1, z_2^1) and the pixels for the bilinear interpolation i_1 and i_2. b) Initializing weights w_4 and w_1 to perform interpolate at pixel 'i_1'. c) Initializing weights $w_5$and $w_2$to perform interpolate at pixel 'i_2'

artifacts and collapse the entire training process. One natural approach to avoid these artifacts is to unfold the deconvolution process by using an up-sampling layer through bilinear interpolation, bi-cubical interpolation, or nearest neighbor interpolation, followed by a convolution kernel [19].

The upsampling task is a key feature of GAN architecture for the generation of time series EEG data bcause it can reduce artifacts and improve discriminator stability. Based on our experience, the interpolation upsampling methods including the nearest neighbor and bi-cubical interpolation can generate EEG signals with good shapes but fail to capture the signal amplitude. The classifier of such a semi-supervised GAN model also yields both poor classifier performance and longer converge times. Conversely, the deconvolution captures the amplitude but produces large artifacts in the generated signals.

To address these issues, we used a deconvolution layer with bilinear weights initialization. The idea of initializing the deconvolution kernel is to encourage a bilinear interpolation. The bilinear weights initialization for one dimensional upsampling assumes an upsampling factor of 3 and a deconvolution kernel with weights w_1, w_2, w_3, w_4, and w_5 as depicted in Figure 3. The weight calculation for interpolation of $i_1$is performed by sliding the center weight w_3 on pixel i_1. The bilinear weights w_1 and $w_4$can now be calculated based on the contribution of 'z_1^1' and 'z_2^1' to interpolate i_1 . Hence the weight w_4 comes out to be $2/3$ and w_1 to be $1/3$. Similarly, by sliding the center weight w_3 at i_2 , we can calculate the contribution of 'z_1^1' and 'z_2^1' for interpolating i_2. Finally, we can assign a weight value 1 to the center weight w_3. This way we can still preserve the advantages of the bi-linear interpolation by avoiding artifacts and have a trainable kernel to help learn the amplitude of real samples, thus increasing classification performance.

C. Proposed Semi-supervised GAN (sWGAN) architecture.

In the proposed sWGAN, the upsampling process uses a two-step upsampling by the factor of 2 in the generator with combinations of fully connected layers and convolution layers. In first step upsampling, a bi-cubical interpolation followed by a convolution layer is used and then a deconvolution with bilinear weights initialization is applied in the second step. The kernel size of deconvolution is kept (1×4) for experimenting on one channel EEG data and (4×4) for all 64 channels while the stride is kept 2 . The deconvolution kernel in both the cases is kept divisible by the stride in order to avoid checkerboard patterns. Bi-cubical interpolation has been shown to perform better compared to other sampling methods in first step. This upsampling arrangement gives the best performance in reducing artifacts and improving GAN training and, eventually, classifier performance.

The detailed sWGAN model architectures for a single-channel and 64-channel input EEG data are shown in Table 1 and 2, respectively. The only difference between the 64-channel and 1-channel architectures is that for 6 chanels the deep learning operations such as upsampling, convolution, and activations are performed in both EEG channel and time step dimensions, while operations for a single channel are performed only in the time dimension (the kernel dimension for EEG channels is kept 1). Because the sWGAN architecture for 64 channels and 1 channel are similar in terms of model layers and upsampling methods, only the 64 channel architecture is described here. However, the component tables for both the architecture are provided.

The generator of the 64-channel sWGAN takes input as a sample from a120-dimensional i.i.d normal distribution $(0, 1)$ followed by a fully connected layer of 1024 neurons and Leaky Relu activation function. Another block of 2048 neurons is used through this time batch normalization is performed before applying the activation function. The output of these blokes is reshaped to $(128 \times 16 \times 16)$. A Bi-cubical interpolation upsampling with batch normalization and Leaky Relu is done in both EEG channel and time dimensions to get an increased resolution of $(128 \times 32 \times 32)$. This output is fed to a 2D convolution block with 64 kernels of size (3×3) with batch normalization and activation for the second stage of upsampling which uses the deconvolution layer of (4×4) kernel with bilinear weight initialization instead. Finally, a convolution layer with of (3×3) with one kernel is applied at the end of generator architecture to generate a dimension consistent to real signals. Discriminator architecture is very similar to any convolutional neural network (CNN) classifier which takes an EEG input of dimension (64×64). An additional Gaussian noise at the beginning of discriminator architecture helps improve training stability and learning.

Generator		**Discriminator**	
Layer	**Output Shape**	**Layer**	**Output Shape**
Input layer	120	Input layer	1, 64, 1
Fully connect.	1024	Gaussian noise	1, 64, 1
LeakyReLU	1024	Conv. 2D	1, 64, 64
Fully connect.	2048	LeakyReLU	1, 64, 64
Batch Norm.	2048	Conv. 2D	1, 32, 128
LeakyReLU	2048	LeakyReLU	1, 32, 128
Reshape	1, 16, 128	Conv. 2D	1, 16, 128
Upsampling	1, 32, 128	LeakyReLU	1, 16, 128
Batch Norm.	1, 32, 128	Flatten	2048
LeakyReLU	1, 32, 128	Fully connect.	1024
Conv. 2D	1, 32, 64	LeakyReLU	1024
Batch Norm.	1, 32, 64	Validity (r/f)	1
LeakyReLU	1, 32, 64	Classes	2

Deconv. with Bilinear weight	1, 64, 128	**Generator Parameters:** Total: 2,285,761
Batch Norm.	1, 64, 128	Trainable: 4,736
LeakyReLU	1, 64, 128	**Discriminator Parameters:**
Conv. 2D	1, 64, 1	Total: 2,173,441
Output layer	1, 64, 1	Trainable: 2,173,441

Table 1: Proposed Architecture for 1-channel EEG input

Generator		**Discriminator**	
Layer	**Output Shape**	**Layer**	**Output Shape**
Input layer	120	Input layer	64, 64, 1
Fully connect.	1024	Gaussian noise	64, 64, 1
LeakyReLU	1024	Conv. 2D	64, 64, 64
Fully connect.	32768	LeakyReLU	64, 64, 64
Batch Norm.	32768	Conv. 2D	32, 32, 128
LeakyReLU	32768	LeakyReLU	32, 32, 128
Reshape	16, 16, 128	Conv. 2D	16, 16, 128
Upsampling	32, 32, 128	LeakyReLU	16, 16, 128
Batch Norm.	32, 32, 128	Fully connect.	32768
LeakyReLU	32, 32, 128	Fully connect.	1024
Conv. 2D	32, 32, 64	LeakyReLU	1024
Batch Norm.	32, 32, 64	Validity (r/f)	1
LeakyReLU	32, 32, 64	Classes	2
Deconv. with Bilinear weight	64, 64, 128	**Generator Parameters:** Total: 34,049,601	
Batch Norm.	64, 64, 128	Trainable: 33,983,425	
LeakyReLU	64, 64, 128	**Discriminator Parameters:**	
Conv. 2D	64, 64, 1	Total: 33,777,536	
Output layer	64, 64, 1	Trainable: 33,777,536	

Table 2: Proposed Architecture for 64-channel EEG input

Three blocks of 2D convolution with Leaky Relu are added before flattening the input to finally feed into a fully connected block with 1024 neurons. Finally we take two sets of outputs, one output is a logit with a single neuron also called as validity (real/fake) for WGAN loss which tries to reduce the earth moving distance between logits of real samples and generated samples and second a two-neuron output to model the class probability for classification task [Figure 1a]. The training ratio of 1: 5 is kept for generator and discriminator update to extract maximum performance.

III. DATASET AND PREPROCESSING

A. *XB-Driving Dataset*

The XB-Driving EEG datasets were collected form a simulated driving experiment, where 100 young adult subjects participated in a virtual reality driving task. Each task included either 6, 10-min sessions, 2, 30-min sessions, or a single 45-min session, where both steering and speed must be controlled by the drivers. Every 8 to 12 seconds, a perturbation to the right or left of the driving lane was introduced to push the car out of the driving lane. The driver then had to rapidly steer the car back to the driving lane. The driver's reaction time (RT) was measured to gauge the driver's fatigue state. RT is defined as the time between the onset of the lane perturbation and the instance that the subject began to steer the car back to the driving lane. The mean RT (±SD) of all drivers 1.48 ± 0.30 sec. Two states were defined based on RT, where and the driver was considered as in the alert condition when

$$RT \leq \text{Mean(RTs)-2Std(RTs)=1.48-2(0.30)=0.88s} \quad (1)$$

or in the drowsy condition when

$$RT \geq \text{Mean(RTs)+2Std(RTs)= 1.48+2(0.30)=2.08s.} \quad (2)$$

BioSemi EEG systems with 64 channels and a 256 Hz sampling rate were used for EEG recording. The EEG signals were first band-pass filtered with a bandwidth of 0.1-55 Hz to remove the DC and electrical noise and then down-sampled to 128 Hz. Since the RT-defined alert/drowsy state reflects the fatigue condition of the driver right before perturbation, 1s EEG epochs of dimension 64×128 were extracted from one second before each perturbation and further down-sampled to 64Hz. We labeled each epoch as either drowsy or alert state based on the corresponding RT-defined fatigue condition. We balanced the data by randomly dropping extra drowsy samples to get 752 samples in each of the states (alert, drowsy.

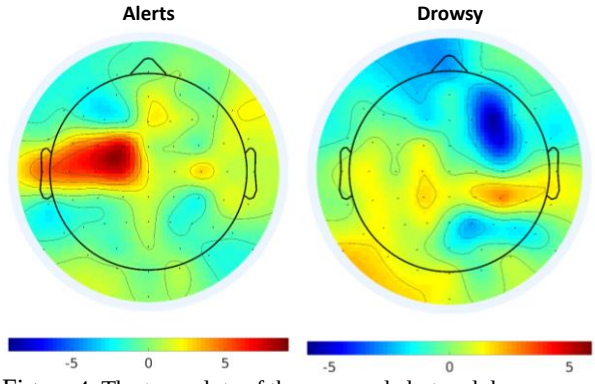

Figure 4: The topo plots of the averaged alert and drowsy

The topographic plots of the averaged alert and drowsy samples for all 64 channels are shown in Figure 4. We can clearly see activation in the left temporal regions under the alert state and deactivation in the right frontal region under the drowsy state. The data are divided into training and test sets of 1000 and 500 samples, respectively.

IV. EXPERIMENTS, RESULTS AND DISCUSSION

Performance of the proposed sWGAN was first assessed on a single channel EEG samples, and then on 64-channel data. The single channel of interest was selected using sequential forward selection criteria and a Support Vector Machine (SVM) with a polynomial kernel. The channel AF7 was selected, which gave 53.82% classification AUC performance,

The experiments are divided into three parts. We started our experiments with training a baseline Convolutional Neural Network (CNN) classifier to classify between alert and drowsy state. We also trained the existing semi-supervised GAN (SGAN) framework as another [14]. We, however, don't use their architecture, which is tuned for MNIST data. Our proposed EEG specific architecture was used in the SGAN network instead. Our proposed sWGAN architecture was then trained 20 times, and the performance statistics are summarized. These experiments are performed for both 1-channel and 64-channel data. Area under the ROC curve (AUC) was used to measure the performance.

A. *Performance of the baseline CNN Classifier*

We trained the baseline CNN classifiers for 1-channel and 64 channel data separately. The hyper parameters of both the classifiers are tuned using hyperas. The 1-channel CNN classifier consists of 3 convolution blocks with 64 kernels each, a kernel size of (1×3), a batch normalization, Relu activation function, max pooling with pool size (1,2), stride (1×2), and a dropout layer as shown in Table 3.

Layer	**Details**
Convolution	Kernel size (1x3)
Batch Normalization	channel axis
Activation	Relu
Max Pooling	Pool size (1, 2), Stride = (1, 2)
Drop-out	0.1

Classifier AUC Evaluation

'AF7' Channel	Baseline CNN	Baseline SGAN	Proposed sWGAN
Mean	56.85	60.27	63.17
STD	2.0228	2.2693	0.9103
64 Channels	Baseline CNN	Baseline SGAN	Proposed sWGAN
Mean	63.318	62.84	66.49
STD	1.4784	3.2415	1.3790

Table 4: Classification results baseline methods and proposed sWGAN

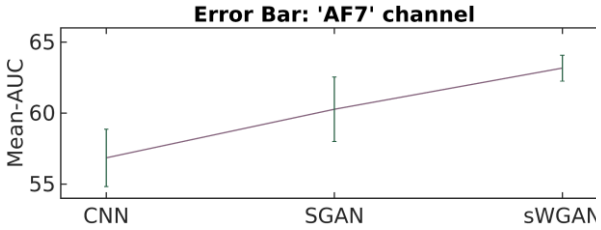

Figure 5: Error bar of classification AUC of baseline methods and proposed sWGAN for 64 channels

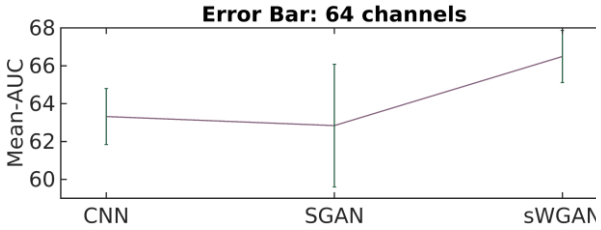

Figure 6: Error bar of classification AUC of baseline methods and proposed sWGAN for single channel

Table 3: Convolution Block

The output of the convolution layers is flattened before being passed on to the fully connected layer and the sigmoid activation. The model is trained to reduce binary cross entropy loss with an ADAM optimizer. A similar architecture is used for the classifier trained on 64 channel. The differences are in the convolution and max pooling kernel size, where a 2D convolution the kernel of size (3x3) and max pooling kernel is kept (2, 2) with a stride of (2, 2).

As shown in Table 4, the average CNN classifier AUC for 1-channel data was 56.85% and for 64 channel was 63.31%

B. Performance of baseline Semi-supervised GAN (SGAN)

The baseline SGAN was also trained for 1-channel and 64-channel data. The GAN training suffered from instability; when modeling was performed multiple times the same EEG input yielded highly different results. Similarly, due to the adversarial framework, it is not certain when the optimal classification performance will occur. Thus, it is useful to create checkpoints of model weights to be able to use the best GAN classifier afterward. We also calculate the GAN classifier AUC on test data after each training epoch. This AUC can also be used as training guidance.

The baseline SGAN achieved a mean AUC of 60.27% on 1-channel data and 62.84% on 64 channel data (Table 4). The average AUC of SGAN exceeds the mean AUC of CNN for 1 channel (56.85 %), but is slightly worse for 64 channels (63.31%). This suggests the inability of the current framework to model multi-channel distribution. Also, as depicted in the error bar plot of classification AUC for 1-channel (Figure 5), the standard deviation of 20 trials of SGAN in 1- channel case is 2.27 as compared to 2.02 in that of CNN. On the other hand, the standard deviation for 64-channel SGAN is 3.24, which is > 1.47 of 64-channel CNN and 1- channels SGAN (Figure 6). This intuitively suggests that the existing SGAN framework struggles to model a complicated distribution with a possibly increased number of modes within the data.

C. Proposed semi-supervised WGAN (sWGAN) training

The proposed sWGAN was also trained on both 1-channel and 64-channel data. Since the novelty of this method over the baseline SGAN is that in case of SGAN the loss for predicted fake labels does not contribute to the total loss used for optimization. The proposed sWGAN attempts to predict the labels of generated samples to be used for classifier training, which therefore improves the classifier performance. The proposed sWGAN outperformed both the baseline CNN and the existing SGAN framework in terms of classification AUC performance. As shown in Table 4, the average AUC on 64 channel data is 66.49% for sWGAN as compared to 62.84% in case of SGAN and 63.312% for baseline CNN. The increased AUC suggests that the improved loss function helps the sWGAN to learn the distribution better by taking advantage of generated data labels. Similarly, for 1-channel sWGAN the average AUC is 63.17% which is better than 62.27% average AUC of SGAN and 56.85% average AUC of baseline CNN, depicted in Table 4. Notably, the difference in the margin of mean AUCs for 1-channel sWGAN and 1-channel SGAN is < the 64-channel case. This bolsters the argument that with the increased complication and modes in the data with the proposed sWGAN is a promising approach.

Another aspect of our approach is the stability and consistency in the learning process. The sWGAN stabilizes adversarial training with the help of generated samples and labels. Hence, the classifier performance is more consistent, as can be seen by a much lower standard deviation of 1.38 for the 64 channel as compared to 3.24 of SGAN in Figure 6. The standard deviation in the 1-channel case is 0.9103 for sWGAN which is again < 2.2693 for that of SGAN (Figure 5). These lower standard deviations suggest that the sWGAN approach is less affected with the initial states and also learns the data modes better than the existing SGAN framework.

V. CONCLUSION

A semi-supervised WGAN (sWGAN) is proposed to classify EEG data collected during a driving simulator experiment. To overcome the creation of artifacts by the generator, we proposed a combination of interpolation and deconvolution with bilinear weights initialization for upsampling in the generator. The proposed architecture used the fake labels, which are predicted by the classifier trained on real data, for the gradient calculation on fake data samples for optimization of classifier part. The discriminator part, on the other hand, uses the Wasserstein loss with gradient penalty. We empirically show that this framework had promising results in terms of increased stability and improved classification performance. The semi-supervised learning of GANS for EEG data opens up the possibility of extending this framework to unsupervised representation by taking advantage of unlabeled data. Such an approach could enhance AI studies of EEG due to the scarcity of good EEG data.

ACKNOWLEDGMENT

The research was partially supported by a grant from Intel Corporation and also sponsored by the Army Research Laboratory under Cooperative Agreement Number W911NF-

10-2-0022. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implies, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for the Government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] M. Hajinoroozi, Z. Mao, T.-P. Jung, C.-T. Lin, and Y. Huang, "EEG-based prediction of driver's cognitive performance by deep convolutional neural network," Signal Process. Image Commun., vol. 47, pp. 549–555, Sep. 2016.

[2] Chun-Hsiang Chuang, Li-Wei Ko, Yuan-Pin Lin, Tzyy-Ping Jung, and Chin-Teng Lin, "Independent Component Ensemble of EEG for Brain–Computer Interface," IEEE Trans. Neural Syst. Rehabil. Eng., vol. 22, no. 2, pp. 230–238, Mar. 2014.

[3] I. Goodfellow et al., "Generative Adversarial Nets." pp. 2672–2680, 2014.

[4] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, "EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals," p. 7, Jun. 2018.

[5] J.-Y. Zhu, T. Park, P. Isola, A. A. Efros, and B. A. Research, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks Monet Photos," 2018.

[6] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial Discriminative Domain Adaptation."

[7] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "NEURAL PHOTO EDITING WITH INTROSPECTIVE AD-VERSARIAL NETWORKS."

[8] J. T. Springenberg, "Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks," Nov. 2015.

[9] T. Miyato, A. M. Dai, and I. Goodfellow, "ADVERSARIAL TRAINING METHODS FOR SEMI-SUPERVISED TEXT CLASSIFICATION," 2017.

[10] A. Kumar, P. Sattigeri, and P. T. Fletcher, "Semi-supervised Learning with GANs: Manifold Invariance with Improved Inference."

[11] J. T. Springenberg, "UNSUPERVISED AND SEMI-SUPERVISED LEARNING WITH CATEGORICAL GENERATIVE ADVERSARIAL NETWORKS."

[12] Y. Tu, Y. Lin, J. Wang, and J.-U. Kim, "Semi-Supervised Learning with Generative Adversarial Networks on Digital Signal Modulation Classification," CMC, vol. 55, no. 2, pp. 243–254, 2018.

[13] E. Denton, S. Gross, and R. Fergus, "SEMI-SUPERVISED LEARNING WITH CONTEXT-CONDITIONAL GENERATIVE ADVERSARIAL NETWORKS."

[14] A. Odena, "Semi-Supervised Learning with Generative Adversarial Networks," Jun. 2016.

[15] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, Improved Training of Wasserstein GANs. 2017, pp. 5767–5777.

[16] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Nov. 2015.

[18] J. G.-C. P. for S. CS231N and U. 2014, "Conditional generative adversarial nets for convolutional face generation," pdfs.semanticscholar.org, vol. 2, 2014.

[19] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 2, pp. 295–307, Feb. 2016.