arXiv:1806.01875v1 [eess.SP] 5 Jun 2018

EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals

Kay Gregor Hartmann [1] **Robin Tibor Schirrmeister** [1] **Tonio Ball** [1]

Abstract

Generative adversarial networks (GANs) are recently highly successful in generative applications involving images and start being applied to time series data. Here we describe EEG-GAN as a framework to generate electroencephalographic (EEG) brain signals. We introduce a modification to the improved training of Wasserstein GANs to stabilize training and investigate a range of architectural choices critical for time series generation (most notably up- and down-sampling). For evaluation we consider and compare different metrics such as Inception score, Frechet inception distance and sliced Wasserstein distance, together showing that our EEG-GAN framework generated naturalistic EEG examples. It thus opens up a range of new generative application scenarios in the neuroscientific and neurological context, such as data augmentation in brain-computer interfacing tasks, EEG super-sampling, or restoration of corrupted data segments. The possibility to generate signals of a certain class and/or with specific properties may also open a new avenue for research into the underlying structure of brain signals.

1. Introduction

While large parts of machine learning deal with the decoding of information from real-world data such as in classification tasks, there is also the recently very active field of how to generate such real-world data through implicit generative models. Generating artificial data could for example be used for data augmentation by producing natural looking samples that are not included in the original data set and thereby artificially increase training data with unseen samples. Additionally, the possibility to produce natural looking samples with certain properties, and the investigation of the models creating them, can be a useful tool for understanding the original data distribution used for training the GAN

One recently proposed framework for the generation of artificial data are generative adversarial networks (Goodfellow et al., 2014) which showed groundbreaking results for the generation of artificial images. Originally, vanilla GANs suffered heavily from training instability and were restricted to low resolution images. A lot of advancement in regard to stability and the quality of the generated images has been made with different regularization methods (Mao et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Kodali et al., 2017) and by progressively increasing the image resolution during training (Karras et al., 2017). GANs also allow the intentional manipulation of specific properties in generated samples (Radford et al., 2015) and therefore could prove to be a useful tool in understanding the original data distribution used for training the GAN.

GANs have mainly been developed and applied to the generation of images and only a few studies investigating time series were conducted; recently they showed promising results for the generation of artificial audio (Donahue et al., 2018). The generation of artificial EEG signals would have applications in many different areas dealing with decoding and understanding brain signals, but to our best knowledge no research regarding the generation of raw EEG signals with GANs has been published at this time.

In this work, we apply the GAN framework to the generation of artificial EEG signals. Though the generation of time-series data is often approached with autoregressive models (e.g. WaveGAN by van den Oord et al. (2016)), we deliberately chose regular convolutional neural networks - on the one hand because most of GAN studies use the DCGAN (Radford et al., 2015) architecture which is based on CNNs, on the other hand because the local and hierarchical structure of CNNs may allow for better interpretability (Sturm et al., 2016; Kindermans et al., 2017; Schirrmeister et al., 2017; Hartmann et al., 2018) that is particularly important for brain singals in a neuroscientific or clinical context. To generate naturalistic samples of EEG data, we propose an improvement to the Wasserstein GAN training showing increased training stability. Furthermore, we compare different evaluation metrics and discuss methodological and

[1]Translational Neurotechnology Lab, Medical Center, University of Freiburg, Freiburg i.Br., Germany. Correspondence to: Kay Hartmann <kg.hartma@gmail.com>.

architectural choices of the network that delivered the best results in this study.

2. Methods

2.1. GAN background and improvement

The GAN framework consists of two opposing networks trying to outplay each other (Goodfellow et al., 2014). The first network, the discriminator, is trained to distinguish between real and fake input data. The second network, the generator, takes a latent noise variable z as input and tries to generate fake samples that are not recognized as fake by the discriminator. This results in a minimax game in which the generator is forced by the discriminator to produce ever better samples.

One big drawback of GANs is the notorious instability of the discriminator during training. The discriminator might collapse into only recognizing few and narrow modes of the input distribution as real, which drives the generator to produce only a limited amount of different outputs. Mode collapse is very problematic for training GANs and is the subject of various works (Mao et al., 2016; Arjovsky et al., 2017; Gulrajani et al., 2017; Kodali et al., 2017).

Wasserstein GANs and their improved version proposed by Arjovsky et al. (2017) show promising advances for training stability. The original GAN framework tries to minimize the Jensen-Shannon (JS) divergence between the real data distribution $\mathbb{P}_r$ and fake data distribution $\mathbb{P}_\theta$ (Goodfellow et al., 2014). If the discriminator is trained to optimality this may lead to the problem of vanishing gradients for the generator (Arjovsky et al., 2017). Arjovsky et al. (2017) proposed to to minimize the Wasserstein distance between the distributions instead of the JS-divergence . This leads the discriminator (now called critic) to maximize the difference

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

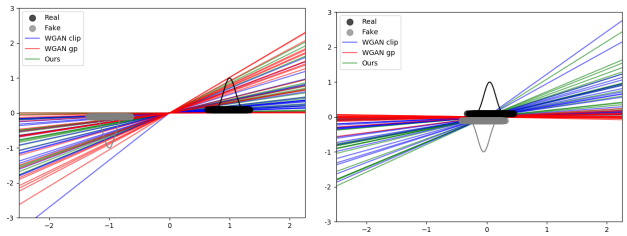
and the generator to maximize $E_{x_f \sim \mathbb{P}_\theta}[D(x_f)]$. They showed that the critic provides a useful gradient for the generator everywhere if $D(x)$ is K-Lipschitz. In their original paper, they enforced Lipschitz continuity by clipping the weights of the discriminator to an interval $[-c, c]$ (WGAN-clip), but later derived a more elegant solution by adding a gradient penalty term

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

with $\mathbb{P}_{\hat{x}}$ containing the points lying on a straight line between real and generated samples, to the critic loss (Gulrajani et al., 2017).

The choice of λ is crucial when training WGANs with gradient penalty (WGAN-GP). If λ is chosen too high, the penalty term can easily dominate the distance term. The other way around, if λ is chosen too small, the lipschitz continuity is not sufficiently enforced. We noticed that a good choice for λ heavily depends on the distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$. If the distance is high, λ has to be chosen accordingly high. If they are close, λ has to be chosen accordingly low. However, during training the generator learns to approximate $\mathbb{P}_r$. This leads to a decrease of the distance between $\mathbb{P}_r$ and $\mathbb{P}_\theta$, whereas λ stays constant.

Figure 1a shows WGAN-clip and WGAN-GP critics trained to distinguish between two normal distributions. Parameters c and λ were chosen such that the critics provide a useful gradient and $||\nabla_{\hat{x}} D(\hat{x})||_2 <= 1$ for most critics . Figure 1b again shows critics with the same parameter setting trained to distinguish between two distributions, but now with decreased distance between them. This simulates the approximation of $\mathbb{P}_\theta$ to $\mathbb{P}_r$ by the generator. WGAN-GP critics noticeably collapse into vanishing gradients and in several cases displaying $E_{x_f \sim \mathbb{P}_\theta}[D(x_f)] > E_{x_r \sim \mathbb{P}_r}[D(x_r)]$. WGAN with weight clipping remains stable, as its only objective is to maximize the critic difference regularized only by limiting network weights. Limiting weights, however, leads to an undesired convergence of network parameters to those limits (Gulrajani et al., 2017).



(a) **(b)**

Figure 1. WGAN critics trained to optimality for two distributions being **(a)** distant and **(b)** near.

Therefore, we propose an improvement to WGAN-GP by gradually relaxing the gradient constraint. Instead of only weighting the penalty term with λ, we also scale it by the current critic difference $\tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta)$. Thereby, the penalty term is only heavily enforced if the first objective of the critic to distinguish between $\mathbb{P}_r$ and $\mathbb{P}_\theta$ is met and additionally λ is scaled down for decreasing distribution distances. Additionally, we will not use the two-sided penalty $P_2(\mathbb{P}_{\hat{x}})$ recommended by Gulrajani et al. (2017), but the one-sided penalty

$$P_1(\mathbb{P}_{\hat{x}}) = \lambda \cdot E_{\hat{x} \sim \mathbb{P}_{\hat{x}}}[\max(0, ||\nabla_{\hat{x}} D(\hat{x})||_2 - 1)^2]). \quad (3)$$

They did not state a specific reason to choose the two-sided penalty over the one-sided penalty, but preferred it from empirical results. The resulting loss function for the critic

then becomes:

$$L_c = -\tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta) + \max(0, \tilde{W}(\mathbb{P}_r, \mathbb{P}_\theta)) \cdot P_1(\mathbb{P}_{\hat{x}}). \quad (4)$$

Critics trained with this loss exhibit stable gradients for distributions with decreasing distance (Figure 1).

2.2. Training and architecture choices

We trained our networks according to the setup described in Karras et al. (2017) They showed that the generated image quality can be enhanced by training the network progressively with increasing resolution. Accordingly, we start at a resolution of 24 time samples and increase the resolution by factor 2 over 6 steps to arrive at 768 samples. Factor 2 introduced the least frequency artifacts and led to the best results. We included additional techniques from Karras et al. (2017) such as minibatch standard deviation, equalized learning rate and pixel normalization. Instead of their proposed additional penalty term $\epsilon \cdot E_{x_r \sim \mathbb{P}_r}[D(x_r)^2]$ to keep the critic from drifting too far away from 0, we instead use $\epsilon \cdot (E_{x_r \sim \mathbb{P}_r}[D(x_r)] + E_{x_f \sim \mathbb{P}_\theta}[D(x_f)])^2$ with $\epsilon = 0.001$ to keep the critic centered at 0.

In opposition to Karras et al. (2017), we do not train the critic and generator equally, but train the critic until optimality first (by 5 critic iterations, as originally proposed by Arjovsky et al. (2017)). We set $\lambda = 10$, as originally recommended by Gulrajani et al. (2017) Each resolution stage is trained for 2000 epochs (which equals to 876.000 signal showings), with an additional 2000 epochs for fading in each stage. The networks are trained using the ADAM optimizer (Kingma & Ba, 2014) with $lr = 0.001$, $\beta_1 = 0$ and $\beta_2 = 0.99$. Latent variables z for the generator are sampled from $\mathcal{N}(0, 1)$.

Our network architecture can be seen in Table 1. Each upsampling block in the generator consists of an upsampling layer followed by 2 convolution layers of size 9. Similarly, each critic block consists of 2 convolution layers followed by 1 downsampling layer. For downsampling we used average pooling and strided convolutions with a size and stride of 2. We use leaky ReLUs in the critic and generator to avoid sparse gradients.

For upsampling we compared nearest-neighbor upsampling and linear and cubic interpolation. As stated by Donahue et al. (2018), upsampling always introduces aliasing frequency artifacts. Whereas they argue that those artifacts may be necessary to produce fine-grained details, we believe that it unnecessarily complicates training for the generator, at least in the case of EEG signals. Nearest-neighbor upsampling introduces strong high-frequency artifacts, while linear or cubic interpolation lead to much weaker artifacts (Figure 2). We argue this is favorable, because we do not want the generator to filter out artifacts after upsampling, but expect the generation of additional high-frequency features.

Table 1. Network architecture

(a) Generator

Layer	Act./Norm.	Output shape
Latent vector	-	200 x 1
Linear	LReLU	50 x 12
Upsample	-	50 x 24
Conv 9	LReLU/PN	50 x 24
Conv 9	LReLU/PN	50 x 24
Upsample	-	50 x 48
Conv 9	LReLU/PN	50 x 48
Conv 9	LReLU/PN	50 x 48
Upsample	-	50 x 96
Conv 9	LReLU/PN	50 x 96
Conv 9	LReLU/PN	50 x 96
Upsample	-	50 x 192
Conv 9	LReLU/PN	50 x 192
Conv 9	LReLU/PN	50 x 192
Upsample	-	50 x 384
Conv 9	LReLU/PN	50 x 384
Conv 9	LReLU/PN	50 x 384
Upsample	-	50 x 768
Conv 9	LReLU/PN	50 x 768
Conv 9	LReLU/PN	50 x 768
Conv 1	-	1 x 768

(b) Critic

Layer	Act.	Output shape
Input signal	-	1 x 768
Conv 1	LReLU	50 x 768
Conv 9	LReLU	50 x 768
Conv 9	LReLU	50 x 768
Downsample	-	50 x 384
Conv 9	LReLU	50 x 384
Conv 9	LReLU	50 x 384
Downsample	-	50 x 192
Conv 9	LReLU	50 x 192
Conv 9	LReLU	50 x 192
Downsample	-	50 x 96
Conv 9	LReLU	50 x 96
Conv 9	LReLU	50 x 96
Downsample	-	50 x 48
Conv 9	LReLU	50 x 48
Conv 9	LReLU	50 x 48
Downsample	-	50 x 24
Conv 9	LReLU	50 x 24
Conv 9	LReLU	50 x 24
Downsample	-	50 x 12
Linear	-	1 x 1

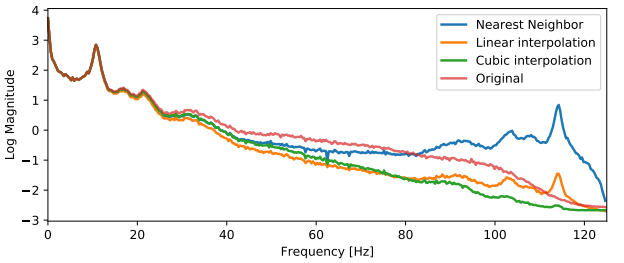


Figure 2. Mean frequency spectra of EEG signals after upsampling visualizing aliasing artifacts in high frequencies. The original signals were first downsampled by average pooling and then upsampled by different upsampling methods.

2.3. Evaluation metrics

2.3.1. Inception score

Evaluation of the distribution of the samples generated by of the generator is an ongoing challenge. Visual inspection of generated samples can be helpful to identify obvious failures and mode collapse, but fail to give any quantitative information about the variance of generated samples and how similar they are to the training data. A common approach to give information about the quality of the trained generator is to use the inception score (IS) (Salimans et al., 2016). To calculate the inception score, a classifier is trained on the training data and used to determine the entropy of the conditional label distribution of generated samples, which should be low, and the entropy of their marginals, which should be high. Though the inception score was shown to be well-correlated with author annotations, it fails to give useful information about the quality of the generator. The inception score is highly sensitive to noise and is not able to detect mode collapse, as it solely relies on the final probabilities of the classifier. We used a pretrained Deep4

model (Schirrmeister et al., 2017) as a replacement for the inception model for EEG data.

2.3.2. Frechet inception distance

The Frechet inception distance (FID) (Heusel et al., 2017) aims to give a better evaluation of the quality of generated samples and is a proper distance. Similarly to the inception score, the FID also uses a trained classifier. But instead of simply evaluating the distribution of class probabilities of generated samples, the FID compares the values in the embedding layer (i.e., the layer before the final classification layer) for real and generated samples. The Frechet distance is used to calculate the Wasserstein2 distance between the distributions of values in the embedding layer for real and fake samples, under the assumption that they follow a multivariate Gaussian distribution. The FID has been shown by Heusel et al. to be consistent with human judgment, and, in contrast to the inception score, more robust to noise, giving information about the quality of the generated samples and to be sensitive to mode collapse. However, it is also not able to detect overfitting of the generator to training samples.

2.3.3. Euclidean distance

The euclidean distance can be used to evaluate how similar generated samples are to the training data. By comparing the distances between generated and real samples, we can investigate of the generator simply replicates samples from the training set or produces something unseen, a property often not evaluated for generative models, but especially important for us, since we have a lower amount of training samples compared to very large image datasets. Optimally, the distribution of minimal distances between real and fake samples (ED_{min}) should be equivalent to the distribution of minimal distances between only real samples with others than themselves.

2.3.4. Sliced Wasserstein distance

The Wasserstein distance describes the cost of transforming one distribution to another using under a given cost function (see (Peyré & Cuturi, 2018) for a more detailed explanation and an overview). The sliced Wasserstein distance (SWD) is an approximation to the Wasserstein distance using 1d projections. It approximates the Wasserstein distance by computing Wasserstein distances between all 1d-projections (slices) of the two distributions. This has the advantage of a closed-form solution and corresponding fast computation for the 1d-case. In practice, the sliced Wasserstein distance is itself approximated by using a finite set of random 1d-projections (Rabin et al., 2012). A low sliced Wasserstein distance indicates that the two distributions are similar in their appearance and variation of samples.

3. Data

The EEG signals we will use for training stem from a simple motor task in which the subjects were instructed to either rest or move the left hand. The signals were recorded with a 128-electrode EEG system and downsampled to 250 Hz. The subject showed characteristic spectral information for left hand movement at electrode channel FCC4h in the alpha, beta and high gamma range. Here, we will only use channel FCC4h for training the GAN. The dataset was scaled to $[-1, 1]$ by subtracting the mean and then dividing by the maximum absolute value. Overall the dataset contains 438 signals, from which 286 will be used as training data for the inception classifier, 72 as validation and 80 as test set. All 438 signals will be used for training the GAN.

#	Model	IS	FID	ED_{min}	SWD
1	AVG-NN	1.361	**9.523**	**-0.056**	*0.102*
2	CONV-NN	1.297	16.755	-0.121	0.084
3	CONV-LIN	**1.363**	11.854	-0.252	0.086
4	CONV-CUB	*1.292*	*33.765*	*-0.375*	**0.078**
5	WGAN-GP CONV-CUB	1.281	120.854	+0.034	0.309
	Real	1.555	0.	4.653	0.
	Noise	1.049	614.782	+1.061	0.155

Table 2. Results for GANs with different architectures. AVG denotes average pooling, CONV strided convolution as downsampling. NN denotes nearest-neighbor upsampling, LIN linear and CUB cubic interpolation. All models except WGAN-GP were trained with our method. WGAN-GP collapsed during training. Best scores are marked by bold, worst scores by italic fonts.

4. Results

4.1. Distance results

Table 2 shows metric results for different architectures trained with our method (1-4) and the original WGAN-GP (5). AVG denotes downsampling by average pooling, CONV by strided convolution. For upsampling, NN denotes nearest neighbor, LIN linear interpolation and CUB cubic interpolation. The test accuracy of the inception classifier used for calculating the inception score and Frechet inception distance was 91.25%. Scores for real data and noise (sampled from a normal distribution with mean and variance of the real data) are listed for comparison.

From visual inspection and the FID and sliced Wasserstein distance, it was clear that the WGAN-GP model collapsed, though we have to note that we neither performed any hyperparameter search nor conducted multiple runs to find a working model. However, we neither did this for the models trained with our method and, though they varied in performance, none of them collapsed.The IS gave no strong evidence for the collapse of the model.

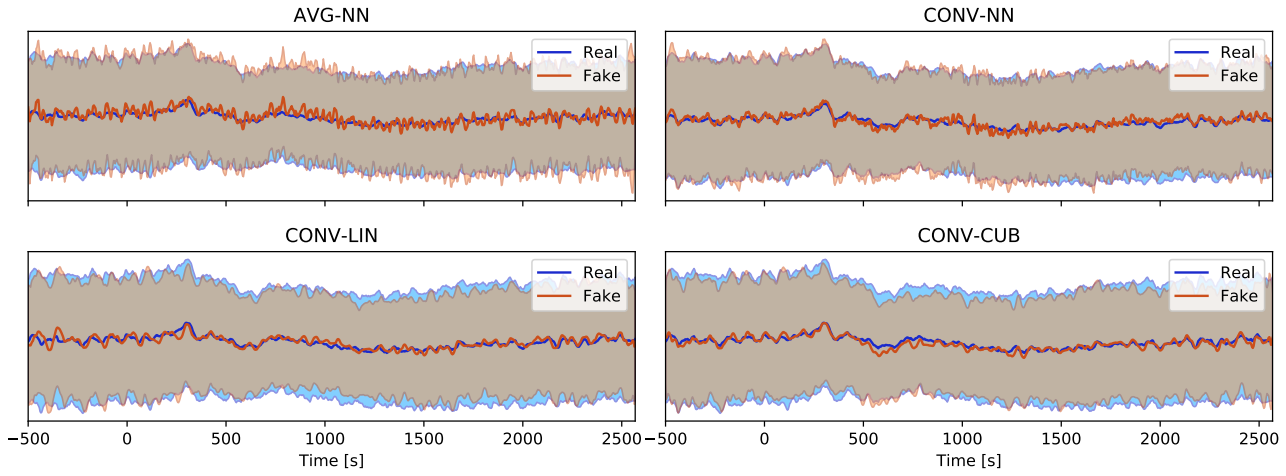


Figure 3. Comparison of the distribution of values at each time point between real signals and signals created by different architectures.

For the models trained with our method, different architectures performed best for different metrics. CONV-LIN performed best for IS with AVG-NN being a close second. AVG-NN did perform best and CONV-CUB clearly worst for FID. EDmin was closest to the real EDmin for AVG-NN, again with CONV-CUB being the worst. For SWD however, CONV-CUB was clearly the best and AVG-NN clearly the worst performing model. Overall, CONV-CUB was the worst performing model for all metrics except SWD.

4.2. Visual inspection

4.2.1. Time samples

Figure 3 shows the mean and standard deviation from signals created by the 4 architectures trained by our model compared to the real signals. The similarity of samples at each time point increases from architecture 1 to 4.. Whereas AVG-NN shows a clear deviation of the generated sample distributions from real data, CONV-CUB shows a very good fit.

4.2.2. Frequency spectra

Similarly, Figure 4 shows the comparison of frequency-resolved spectral power distributions. AVG-NN and CONV-NN show deviations from the real spectrum even in low frequencies, whereas CONV-LIN and CONV-CUB again show a good fit. It can be argued that CONV-LIN better fits low frequencies, whereas CONV-LIN better fits high frequencies. No model managed to properly fit frequencies higher than 100 Hz (which however have very low power).

4.2.3. Generated samples

Figure 5 shows random samples generated by AVG-NN and CONV-CUB. Both models appear to generate visually

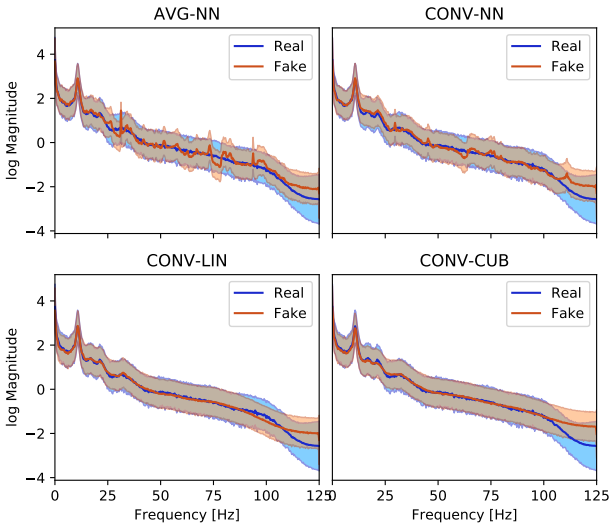
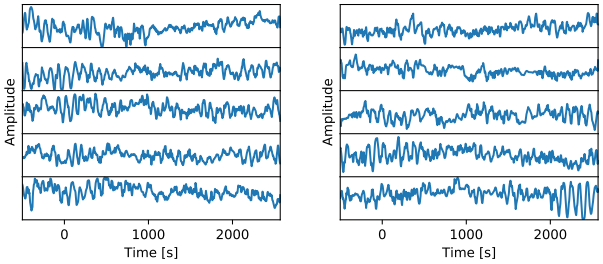


Figure 4. Comparison of the distribution of frequency spectra real signals and signals created by different architectures.

sound signals. A notable difference between the signals is that CONV-CUB decently often creates signals containing only weakly oscillating sequences. Those sequences were highly indicative for fake signals when surveyed by visual comparison to real signals.

4.3. Class-specific properties

To investigate class specific properties of signals generated by CONV-CUB, we used the inception classifier to determine signals that were classified to belong to either class with >90% accuracy. Similarly, we used the same approach to determine real signals that exhibited a probability of >90% for either class. A comparison of the respective frequency spectra is shown in Figure 6. Generated signals classified as left hand movement match the decrease of al-



(a) **(b)**

Figure 5. Signals created by the **(a)** AVG-NN and **(b)** CONV-CUB models.

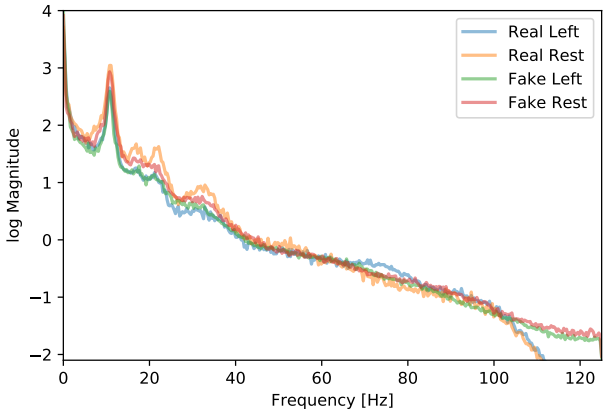


Figure 6. Comparison of mean frequency spectra of real and generated signals that were either classified as either resting activity or left hand movement.

pha and beta activity in the real signals up to around 40 Hz. Generated left hand signals though did not express the increase of high gamma activity present in real left hand signals.

5. Discussion & Conclusion

With this work, we showed that it is possible to generate artificial EEG signals with a generative adversarial network. With a modification of the improved WGAN training (Gulrajani et al., 2017), we were able to progressively train a GAN to produce artificial signals in a stable fashion that strongly resemble single-channel real EEG signals in the time and frequency domain.

We evaluated the up- and down-sides of several choices for up- and downsampling by comparing them with various metrics. In our case, the inception score (IS) (Salimans et al., 2016) did not give meaningful information about the quality of signals generated by a model. Additionally we observed that models with the lowest Frechet inception distances (FID) (Heusel et al., 2017) did not necessarily produce signals with spatial and spectral properties similar to the real input samples. The model that produced the most naturally distributed signals according to spatial and spectral properties was assigned the worst FID. Therefore optimization of GANs used for EEG towards good IS or FID could lead to the production of signal distributions wrongly believed to be similar to real data. The model expressing the most natural looking spatial and spectral distributions had the best sliced Wasserstein distance (SWD). A low Euclidean distance suggests a preference of the generator towards specific training samples, though in our case it was not as low that it indicates the simple reproduction of training samples. Overall, no single metric gave sufficient information about the quality of a model, but the combination of FID, SWD and ED gave a good idea about its possible overall properties. Therefore we do not recommend any single metric, but encourage the use of several metrics with different advantages and disadvantages. encourage the use of several metrics with different advantages and disadvantages.

6. Outlook

With the first step for the generation of artificial EEG signals done, there are now many open possibilities for further investigations. Of course, the next step would be to not only generate single channel signals, but to model complete multi-channel EEG recordings. For this it will be important to further understand the impact of different design choices such as convolution size and up- and down-sampling techniques. In our experiments we noted a strong influence of convolutional size onto which frequency ranges are correctly expressed by the generator. Furthermore, we are currently applying our models to a large sample of EEG recordings from different subjects and will evaluate the quality of produced signals by an ensemble of medical experts.

In summary, EEG-GANs open up the possibility for new applications, not only limited to data augmentation, but e.g. also spatial or temporal super-sampling (Corley & Huang, 2018) or restoration of corrupted signals. The possibility to generate signals of a certain class and/or with specific properties may also open a new avenue for research into the underlying structure of brain signals.

References

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein GAN. jan 2017. URL http://arxiv.org/abs/1701.07875.

Corley, I. A. and Huang, Y. Deep EEG super-resolution: Upsampling EEG spatial resolution with Generative Adversarial Networks. In *2018 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, pp. 100–103. IEEE, mar 2018. ISBN 978-1-5386-2405-0. doi: 10.1109/BHI.2018.8333379. URL http://ieeexplore.ieee.org/document/8333379/.

Donahue, C., McAuley, J., and Puckette, M. Synthesizing Audio with Generative Adversarial Networks. feb 2018. URL http://arxiv.org/abs/1802.04208.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Networks. jun 2014. URL http://arxiv.org/abs/1406.2661.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved Training of Wasserstein GANs. mar 2017. URL http://arxiv.org/abs/1704.00028.

Hartmann, K. G., Schirrmeister, R. T., and Ball, T. Hierarchical internal representation of spectral features in deep convolutional networks trained for EEG decoding. In *2018 6th International Conference on Brain-Computer Interface (BCI)*, pp. 1–6. IEEE, jan 2018. ISBN 978-1-5386-2574-3. doi: 10.1109/IWW-BCI.2018.8311493. URL http://ieeexplore.ieee.org/document/8311493/.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. jun 2017. URL http://arxiv.org/abs/1706.08500.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive Growing of GANs for Improved Quality, Stability, and Variation. oct 2017. URL http://arxiv.org/abs/1710.10196.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: PatternNet and PatternAttribution. may 2017. URL http://arxiv.org/abs/1705.05598.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. dec 2014. URL http://arxiv.org/abs/1412.6980.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. On Convergence and Stability of GANs. may 2017. URL http://arxiv.org/abs/1705.07215.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P. Least Squares Generative Adversarial Networks. nov 2016. URL http://arxiv.org/abs/1611.04076.

Peyré, G. and Cuturi, M. Computational Optimal Transport. mar 2018. URL http://arxiv.org/abs/1803.00567.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein Barycenter and Its Application to Texture Mixing. pp. 435–446. Springer, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-24785-9_37. URL http://link.springer.com/10.1007/978-3-642-24785-9{_}37.

Radford, A., Metz, L., and Chintala, S. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. nov 2015. URL http://arxiv.org/abs/1511.06434.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved Techniques for Training GANs. jun 2016. URL http://arxiv.org/abs/1606.03498.

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., and Ball, T. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 38(11):5391–5420, nov 2017. ISSN 10659471. doi: 10.1002/hbm.23730. URL http://doi.wiley.com/10.1002/hbm.23730.

Sturm, I., Lapuschkin, S., Samek, W., and Müller, K.-R. Interpretable deep neural networks for single-trial EEG classification. *Journal of Neuroscience Methods*, 274:141–145, dec 2016. ISSN 01650270. doi: 10.1016/j.jneumeth.2016.10.008. URL http://www.ncbi.nlm.nih.gov/pubmed/27746229http://linkinghub.elsevier.com/retrieve/pii/S0165027016302333.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. sep 2016. URL http://arxiv.org/abs/1609.03499.