Using adversarial autoencoders to infer actions from the peripheral nervous system

Jos van der Westhuizen[1,2], **Tris Edwards**[1], **Raphael Schmetterling**[1,2], **Robert Tinn**[1,2]
jos@cbas.global, tris@cbas.global, raph_s@cbas.global, rob_t@cbas.global

Oliver Armitage [1], **Joan Lasenby**[2], **Emil Hewage**[1]
oliver@cbas.global, jl221@cam.ac.uk, emil@cbas.global

Cambridge Bio-Augmentation Systems[1]
Cambridge University[2]

Abstract

Advances in neural interface technology are giving rise to a new challenge in computational neuroscience. Chronic implants are able to record the activity of neural populations over several months or even years, creating datasets that are difficult to analyze using current neuroinformatic techniques due to their size and crude labels. Here we propose a practical solution, an adversarial autoencoder comprised of LSTMs to generate a label-like latent representation. For this novel data paradigm, we introduce a dataset of local field potential signals over a 6-week recording from a porcine specimen with natural animal mobilization. Our initial implementation yields a classification accuracy of 83.3%, demonstrating the suitability for neuro-prosthesis applications.

1 Introduction

The peripheral nervous system (PNS) holds a plethora of information on physiological processes conveyed from the central nervous system to other systems in the body. Recent improvements in neural interfaces enable long-term recordings of the PNS, generating datasets too large and containing features too nuanced for manual human analysis. In this paper, we propose an adversarial autoencoder to infer actions from such signals in the presence of limited crude labels. The data are multi-channel signals recorded from the tibial nerve of a healthy porcine specimen.

This problem fits under the umbrella of learning representations, more specifically semi-supervised sequence-to-sequence models. Other generalizations of this problem have been solved in seminal work by Makhzani et al. (2015); Srivastava et al. (2015); Hinton and Salakhutdinov (2006). This scenario is pertinent in the medical field where datasets tend to be large and require expert labeling, warranting the use of unsupervised learning as a practical approach to feature learning (Längkvist et al., 2014). Being able to infer the actions of an agent from peripheral neural signals using semi-supervised learning could mitigate the issue of scant or difficult-to-generate labels and as a result, provide useful control signals for neurological human-machine interfaces with potential benefits for amputees.

This is also the introduction of a potentially very rich new data paradigm for the neuro-prosthesis community. Advances in neural interface technology have allowed the collection of peripheral data in a chronic setting, over long periods of time and in natural mobilization settings. However, compared to the shorter and restricted laboratory studies, the new datasets are expensive to label.

The proposed solution comprises a sequence-to-sequence model to encode a given snippet of the neurological signals into a fixed-size representation. The encoder and decoder are long short-term

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

memory recurrent neural networks (Hochreiter and Schmidhuber, 1997). Inspired by the work in Makhzani et al. (2015), we augment the sequence-to-sequence model with adversarial discriminator networks that regularize the generated latent representations. The aim is to capture the latent factors that best explain the data and label the data accordingly. Autoencoders such as sequence-to-sequence models are well known for being able to extract meaningful latent representations (Murphy, 2012). To the best of our knowledge, this work-in-progress paper is the first analysis of peripheral neurological signals with a sequence-to-sequence model.

2 Model architecture

As in Bowman et al. (2015) we make use of a single layer long short-term memory (LSTM) recurrent neural network as an encoder and decoder in an autoencoder fashion. As illustrated in Figure 1 the encoder, denoted as $q(z, y|x)$, generates two fixed size latent representations z and y of an arbitrary length sequence x. The decoder network then utilizes both the z and y representations to reconstruct the original input signal. We assume that the data analysed by the model are generated by a latent class variable y that comes from a categorical distribution as well as a latent style variable z that comes from a Gaussian distribution:

$$p(y) = Cat(y) \qquad\qquad p(z) = \mathcal{N}(z|0, I)$$

For each step in the decoder, the y section of the LSTM memory is replaced by the original y, where the rest of the vector is left to change over time. This places more emphasis on generating an informative y representation for the decoder to utilize at each step. To simultaneously make learning stable and the model robust, we alternate the input to the decoder at each training iteration to either be the true input x, or the output from the previous step in the LSTM. Moreover, we found that reversing the output when decoding (Srivastava et al., 2015) improves training by allowing the model to start off with low-range correlations.

In order to ensure that the y representation is label-like, a discriminator network is employed to introduce an additional loss term. This follows a generative adversarial network approach (Goodfellow et al., 2014; Makhzani et al., 2015) with the generator being an encoder recurrent neural network. The discriminator learns to distinguish between samples from a categorical distribution (random one-hot vectors) and the y representation generated by the encoder. This encourages the y representation towards a categorical distribution from which we can infer actions.

A second discriminator network is employed to encourage the z representation to be Gaussian distributed. This discriminator has to distinguish between generated representations z and samples from a Gaussian distribution $\mathcal{N}(z|0, I)$. Hence by concatenating a selected y representation, $\tilde{y}$, and a $\tilde{z}$ sampled from a Gaussian distribution, one is able to generate neural-like signals from the decoder. Additionally, this allows generation of mixed categories in y.

Batch normalization (Ioffe and Szegedy, 2015) is applied to the input and the two ReLu (Nair and Hinton, 2010) activated hidden layers with 50 and 20 units respectively. This is followed by minibatch discrimination (Salimans et al., 2016) before being linearly mapped into a scalar value. The structure of both discriminators is the same. Batch normalization, minibatch discrimination, and the Wasserstein generative adversarial network (Arjovsky et al., 2017) were found essential to prevent mode collapse during training.

The model is trained in 3 separate stages. First, the encoder and decoder are trained with the mean squared error loss of the input reconstruction. Second, the discriminator function f learns the difference between labels y generated from the generator function g and categorical samples y'. For N data points with the input denoted as x, the y discriminator loss is

$$L_{D_y} = \frac{1}{N}\sum_{n=1}^{N}(-f(y'_n) + f(g(x_n))) \tag{1}$$

where each y'_n is sampled at random from a categorical distribution. Effectively, the discriminator is trained to produce negative values when the input is generated and positive values when the input is sampled from a categorical distribution. Third, the generator (encoder) is trained to generate a y representation that is one-hot-like by 'fooling' the discriminator. The following loss function

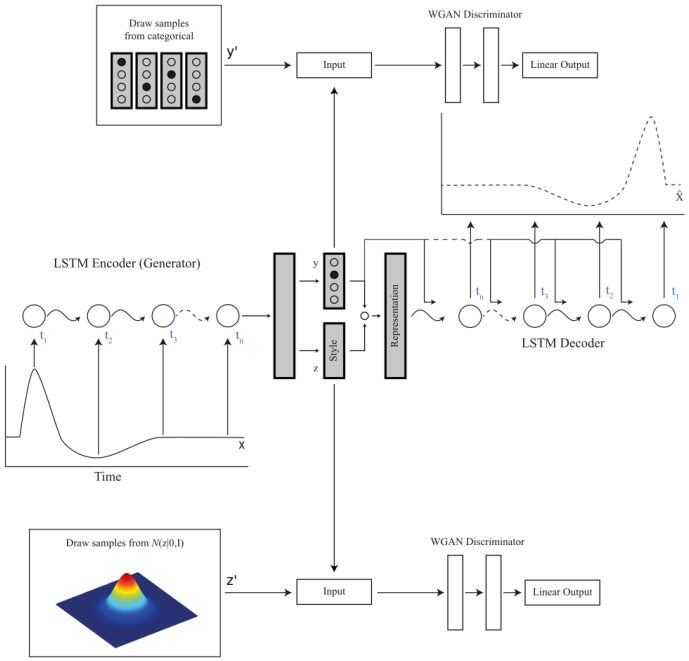

Figure 1: The sequence-to-sequence model regularized by 2 discriminative neural networks. The top discriminator encourages the y representation to be one-hot-like (categorical). The bottom discriminator encourages the z representation to be Gaussian distributed. In the decoder, the original y representation replaces the corresponding elements of the LSTM memory at each time step.

encourages the y encoder to generate a y such that the now fixed function f yields positive values,

$$L_{G_y} = -\frac{1}{N}\sum_{n=1}^{N} f(g(x_n)) \tag{2}$$

The discriminator and generator training updates for the z representation are the same as those detailed above for y, with the exception of replacing the categorical y' with samples z' from a Gaussian distribution. Both z and y generators are updated every third training iteration.

3 Experimental setup

In order to confirm that the model works as intended, we evaluated it on 2 well-understood datasets. The first is a synthetic dataset with 4 classes (sine-, cosine-, saw-tooth-, and square-waves). Here 150,000 samples were generated with unit amplitudes and random periods between 5 and 40 time steps. All the waveforms had a length of 50 time steps. 30,000 samples were held out for testing. The second dataset was a low-resolution version of MNIST (LeCun, 1998) rescaled to range from 0 to 1. Here the MNIST images were resized from a size of 28×28 pixels to 7×7. The images were processed in scanline order (Cooijmans et al., 2016), the shorter sequences making learning easier.

The experimental neural data was collected continuously during natural mobilization of a porcine specimen over a 6-week period. Fifteen channels of local field potential (LFP) data were gathered from the tibial nerve at a sampling rate of 30 kHz using an IT2 PNS implant (Cambridge Bio-Augmentation Systems, Cambridge, UK). The data used in this study is from a 2-hour window 3 days post surgical implantation. Field potentials were identified by means of a 26 µV threshold.

Two variations of the neural data were constructed for investigation. The first dataset consisted of the raw signals on all 15 channels for 50 time steps after a local field potential (LFP) was detected on any channel. A total of 250,911 LFPs were detected over all channels in the recorded period and the data were rescaled to range from 0 to 1. The second dataset consisted of the spike counts on each channel

using a 100ms time window. The counts were similarly rescaled and were then sliced into segments of 50 consecutive counts, resulting in a total of 6,840 data points. The spike counts reduced some of the noise present in the raw data and spanned longer periods. For both datasets above, a single data point had a sequence length of 50 and 15 variables.

Two four-minute video segments of the porcine specimen were recorded during the LFP recordings used for training. Through a retrospective analysis of the video, we were able to identify five distinct motion classes during the time of the video. The actions were: walking forwards, standing, shuffling, reversing, and turning. With the video synchronized to the recorded time series, we were able to label segments of the raw signal according to the identified actions with a granularity of 0.1s. Of the total number of data points in the raw dataset and the count dataset, 3003 and 74 were labeled respectively. The labeled data allowed a quantitative evaluation of the y representation by using the classification accuracy as a proxy. The labeled data were removed from the datasets and not used during training.

For each of the 4 datasets, a validation set was constructed by randomly splitting the training data with an 80:20 (training:validation) ratio. The best model was selected based on the lowest reconstruction error achieved on the validation set over the course of training. Adam (Kingma and Ba, 2014) with a learning rate of 0.001 and no learning rate decay was used for optimization. We denote the number of elements in representations z and y to be S_z and S_y respectively. In order to prevent overfitting on the smaller neural-count dataset, we set $S_y = 20$ and $S_z = 44$. For the synthetic, MNIST, and neural-raw datasets, we chose $S_y = 30$ and $S_z = 98$. Larger y-representations were chosen based on the work in Makhzani et al. (2015) showing that this results in more accurate classifications.

4 Related work

Our work is closely related to the thorough study of Pandarinath et al. (2017), who made use of a variational sequence-to-sequence model to analyze, with unprecedented accuracies, the neurological signals from electrodes implanted in the motor cortex. Our focus is on peripheral neurological signals instead of signals measured in the cortex: one can argue that the former exhibits less interference from other neurological signals, and is certainly a more convenient site for human-machine interfacing for amputees. We also explicitly cluster the signals using the adversarial autoencoder structure.

The model designed as part of this work is similar to that proposed for unsupervised learning in Makhzani et al. (2015). Our model employs recurrent neural networks instead of convolutional neural networks. Kingma et al. (2014) proposed a theoretically grounded technique for doing unsupervised training. The proposed technique was surpassed by the Gumbel-softmax method (Jang et al., 2016), however the adversarial approach Makhzani et al. (2015) remained superior. Our sequence-to-sequence model is based on Bowman et al. (2015). The difference lies in our model using the adversarial autoencoder approach with WGANs for semi-supervised labeling.

For a thorough review of unsupervised feature learning with machine learning, we refer the reader to the fascinating work by Längkvist et al. (2014). The review also gives an overview of all the unsupervised learning applied to signals such as electrocardiograms and electroencephalographs. Serruya et al. (2002) have used motor cortex neurons in primates to control a mouse on a screen. Utilizing the signals in peripheral nerves could replace current electromyography (EMG) systems with improved resolution. EMG have yielded recognition above 95% when combined with neural networks and hidden Markov models (Ahsan et al., 2009) and recognition rates above 90% in a study by Ahsan et al. (2009). To our knowledge, no research has been done to date on a chronic recording of peripheral neurological signals with adversarial sequence-to-sequence networks.

5 Results

Owing to the generated y representation not being fully one-hot encoded, we plot the generated representations for the test sets of the MNIST dataset and the neural-raw dataset in Figure 2 using t-distributed Stochastic Neighbour Embedding (t-SNE) (van der Maaten and Hinton, 2008). From the figure, it is clear that the different classes are much better separated for the MNIST dataset. As expected for the neural-raw dataset, there is more inter-class overlap in the t-SNE plot. Here each data point spans a time of only 0.00167s and thus there could be several data points that are not related to the action we labeled it as. It should also be noted that the clusters in the t-SNE plots are not necessarily the eventual clusters of labels derived from the maximum values in each y representation.

To ensure that the data points are not simply clustered with respect to the time of recording, we overlay the t-SNE plot with a heatmap of the recording time. From this heatmap, it is evident that clusters contain data points from various points in time.

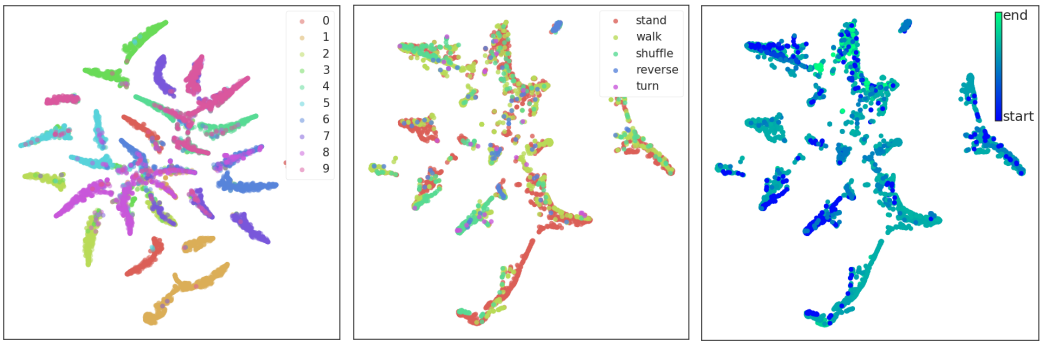

Figure 2: t-SNE plots of the y representation. *Left*: the low-resolution MNIST dataset. *Middle*: The neural-raw dataset colored by class. *Right*: The neural-raw dataset colored by time of recording.

We evaluated classification accuracy as follows: for all dimensions of y find the set $X_i \subset X$ of data points that have maximum probability $p(y|x),\ x \in X$ in dimension i; weight the true class $c \in C_i^{N \times k}$ (where k is the number of true classes) of $x \in X_i$ by the probability $p(y = i|x)$; assign to y_i the maximum class of the average weighted true classes C_i, $\arg\max_k \frac{1}{N} \sum_{n=0}^{N} p(y = i|x_n)c_n$ by means of a hashmap. The accuracy is then computed based on the labels assigned to each data point.

The classification accuracies obtained with the aforementioned datasets are shown in Table 1. The accuracies reported are the averages over five independent runs. We also tabulate the squared error achieved on the test sets to show the efficacy of the data reconstruction achieved by the model. High accuracies were achieved for both the synthetic and MNIST datasets, which confirms that the model works as intended. For the MNIST data, we did not expect similar accuracies to Makhzani et al. (2015) because we use a low-resolution version making the digits harder to distinguish. A higher classification accuracy was achieved on the neural-count dataset compared to the neural-raw dataset. The neural-count dataset reduces the noise present in the raw signal and allows analysis over a longer section of time. Moreover, a single local field potential as measured in the neural-raw dataset (0.00167s) does not necessarily contain information about an action, but a set of them could. The results obtained exceeded expectations given the coarse labeling procedure.

Table 1: Model performance on test data

Dataset	Accuracy [%]	Reconstruction error
Synthetic	89 ± 1.9	0.001316
Low-resolution MNIST	83.5 ± 2.1	0.000177
Neural-raw	63 ± 1.5	0.012113
Neural-count	83.3 ± 1.4	0.006509

Nearly perfect reconstruction was achieved on the MNIST and synthetic dataset. However, the neural data was harder to model and as a result, the reconstructions are smoothed versions of the original signal as shown in Appendix A. Both reconstructions with and without teacher forcing (Goodfellow et al., 2016) are illustrated.

6 Discussion and conclusion

This study, part of a larger on-going project, has proved the utility of the novel, long-term dataset. By means of an adversarial autoencoder, we were able to successfully characterise the dynamics of peripheral neurological signals. We believe this could be of great benefit to amputees by allowing control of smart prosthetic devices through the nerves in a limb. The chosen approach allows the use of a continuous y representation as input to the decoder, which could provide the benefit of allowing combinations of actions as a signal to a prosthetic device. Although the data already span a long

period of time, future work will include validating that the model is able to generalize over much longer periods. One of the main goals is to stitch together datasets collected from different patients as in Pandarinath et al. (2017) to make a large part of the model agnostic to the specific patient. In further work we also aim to compare the model with other architectures.

References

Ahsan, M. R., Ibrahimy, M. I., Khalifa, O. O., and others (2009). EMG signal classification for human computer interaction: A review. *European Journal of Scientific Research*, 33(3):480–501.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv:1701.07875 [cs, stat]*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating Sentences from a Continuous Space. *arXiv:1511.06349 [cs]*.

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., and Courville, A. (2016). Recurrent Batch Normalization. *arXiv:1603.09025 [cs]*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*.

Jang, E., Gu, S., and Poole, B. (2016). Categorical Reparameterization with Gumbel-Softmax. *arXiv:1611.01144 [cs, stat]*.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised Learning with Deep Generative Models. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3581–3589. Curran Associates, Inc.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

LeCun, Y. A. (1998). The MNIST Database of Handwritten Digits. *http://yann.lecun.com/exdb/mnist/*.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial Autoencoders. *arXiv:1511.05644 [cs]*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Pandarinath, C., O'Shea, D. J., Collins, J., Jozefowicz, R., Stavisky, S. D., Kao, J. C., Trautmann, E. M., Kaufman, M. T., Ryu, S. I., Hochberg, L. R., Henderson, J. M., Shenoy, K. V., Abbott, L. F., and Sussillo, D. (2017). Inferring single-trial neural population dynamics using sequential auto-encoders. *bioRxiv*, page 152884.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved Techniques for Training GANs. *arXiv:1606.03498 [cs]*.

Serruya, M. D., Hatsopoulos, N. G., Paninski, L., Fellows, M. R., and Donoghue, J. P. (2002). Brain-machine interface: Instant neural control of a movement signal. *Nature*, 416(6877):141–142.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

A Appendix: Example reconstructions

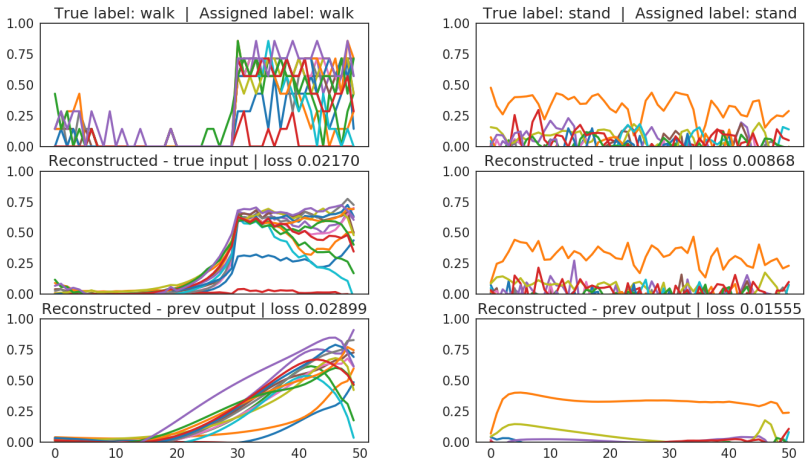

Figure 3: Example reconstructions for the neural-count dataset (left) and the neural-raw data (right). The colors represent the different channels of the signal. The top plot is the original input x, the middle is the reconstructed input where the correct input was fed to the decoder at each step. The bottom graphs are the reconstructions where the output from the previous step is used as input to the next.