Bayesian Neural Network Classification of EEG Correlates of Memory

Saber Bahranifard
Rohan Deshpande
David DiStefano

December 19, 2019

Abstract

In this paper, we train a Bayesian Neural Network classifier on EEG data to try and predict memorization based on brain state. We hypothesize that classification accuracy may be improved by increasing the network depth, and that a BNN will be able to out-perform non-deep machine learning methods. We provide analysis of each of these hypotheses by comparing networks with varying depth and layer size to a Support Vector Machine Classifier.

Results indicate that both SVM and the BNN models we propose are equally insufficient for classification of our EEG data set. While these models may express greater than chance classification accuracy, by further observing high sensitivity and low specificity, we suggest that these models are trained with high biased due to imbalance in the class ratio. Further exploration of these models must be considered using a larger data set in order to sufficiently conclude our hypotheses.

1 Motivation

The accurate classification of "brain states" through the use of scalp EEG (electroencephalogram) data is an area of strong interest in the field of brain-computer-interface research. Of particular interest is the classification of states of memory (i.e. whether a participant is in a "good" or "poor" brain state for encoding upcoming visual stimuli into memory).

In the present research, we attempt this classification using trials of EEG data from 7 participants that have been preprocessed and feature-extracted via spectral decomposition to yield a workable dataset. The feature space consists of 112 spectral power values across 8 scalp regions (computed as the average spectral power across several adjacent electrodes), 7 frequency bands covering from 1Hz to 42Hz, and 2 time windows of 500ms width with no overlap. Each trial of data has a corresponding label of subsequent memory of the trial's visual stimulus (remembered, "Hit"; or forgotten, "Miss").

Here we train a Bayesian neural network (BNN) classifier on each participant's dataset, testing on a secondary test set from each participant. The output of this classifier is designed to return a 1 or a 0, representing Hit or Miss of the test trial. The BNN will be parameterized to use a standard normal distribution as a prior for all weights and biases.

Our primary goal is to evaluate the performance of this classifier in terms of its classification accuracy as we vary the complexity of the model via the number of hidden layers. We hypothesize that average classification accuracy per participant will increase as the number of hidden layers increases up to limit where we expect classification accuracy to plateau and then drop. A more thorough breakdown of our hyperparameter search process can be found in the conclusion of Section 2.

As a means of comparison, we will additionally refer to results of a similarly trained support vector machine (SVM) that has been trained on the same data. The SVM has been commonly used for classification in the BCI literature for EEG data [1, 2, 3, 4]. The SVM will utilize a linear kernel, as is often used for two-class classification. We expect the BNN to consistently yield higher classification accuracy than the SVM.

The results from this experiment will provide preliminary evidence on how well the Bayesian neural network performs for classification of EEG data within the context of subsequent memory prediction. If we are correct in that classification accuracy may be maximized once an optimal network depth is established, this will inform future directions of architecture optimization where hidden layer sizes may then be subsequently optimized.

The use of the optimal network depth observed through training will also serve to provide an optimal classifier per participant to be used on the secondary datasets. This is a novel application of these results, where the classifier will be tested on a separately collected set of data from the same participants used to train the models.

2 Methods

The main model we used to classify EEG data is a Bayesian Neural Network. We compared this probabilistic model to to a non-probablistic model to report on the potential performance differences between these two approaches. We expected to see the probabilistic model outperform its non-probabilistic counterpart.

Our baseline model of choice was a soft-margin Support Vector Machine (SVM) for binary classification [5]. Comparing to SVM is appropriate, given that SVM has been used frequently in EEG classification literature and can be applied to EEG data set without too much effort [3]. Reported classification accuracies for SVM based models are only slightly better than a random classifier, but do not improve on these enough to be satisfactory (the best accuracy reported is 60.72%).

The optimization problem to solve for our baseline model is:

$$\begin{aligned} \min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \\ \text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \\ \zeta_i \geq 0, i = 1, ..., n \end{aligned}$$

where ζ_i are slack variables and their sum is an upper bound for training error. The parameter C is a regularizer to help control overfitting. The dual form is:

$$\begin{aligned} \min_{\alpha} \mathcal{L}(\alpha) = \min_{\alpha} \big[\sum_{i=1}^{N} \sum_{j=1}^{N} \frac{1}{2} \alpha_i^T y_i y_j K(x_i, x_j) \alpha_i - \sum_{i=1}^{n} \alpha_i \big] \\ \text{subject to } \sum_{i=1}^{N} y_i^T \alpha_i = 0 \\ 0 \leq \alpha_i \leq C, i = 1, ..., n \end{aligned}$$

in which parameter C puts an upper bound on the size of Lagrange multipliers, α_i, for support vectors. $K(x_i, x_j)$ represents Kernel functions.

In this project, we use builtin python SVM functions from the Scikit-Learn package to classify data[6]. Preliminary results for the SVM model are presented in the Appendix section. The hyperparameters we used to control our SVM's performance were as follows:

1. Regularization parameter, C: a brute-force algorithm along with Cross Validation is used to find optimal value for this parameter within a reasonable range
2. Kernel function, $K(x_i, x_j)$: Linear Kernel and RBF Kernel with hyperparameter γ, were tried to find the best match for EEG data
3. RBF Kernel hyperparameter, γ: a brute-force algorithm along with Cross Validation is used to find optimal value for this parameter within a reasonable range

The model we propose as an alternative to this non-probabilistic approach is a Neural Network (NN) trained using a Bayesian approach. The use of Feature-extracted EEG data with corresponding memorization class labels make this approach novel in two ways: (1) We will be using a fully Bayesian approach (i.e. Providing a model prior and making predictions via an approximated predictive posterior); and (2) We will model prediction uncertainty by making use of variational inference (VI) using the reparameterization trick.

Our Neural Network maps EEG features to binary outputs by the use of parameter θ, which represents all the weights and biases in the network:

$$y_i \sim p(y_i|x_i, \theta) : R^{112} \rightarrow (0, 1)$$

The likelihood distribution of this network takes the form of a Bernoulli distribution over the training labels given inputs and weights and biases:

$$p(y|x, \theta) = \prod_{i=1}^{N} p(y_i = 1|x_i, \theta)^{y_i}(1 - p(y_i = 1|x_i, \theta))^{(1-y_i)}$$

where the output probability of the NN with parameters θ is defined as:

$$p(y_i = 1|x_i, \theta) = \sigma(NN(x_i, \theta))$$

Uncertainty is incorporated into the model by defining a prior distribution over network parameters, θ. a simple Normal distribution is defined over each parameter:

$$p(\theta) = p(w, b) = \prod_{\ell=1}^{L} \prod_{j=1}^{J^\ell} \prod_{k=1}^{J^{\ell-1}} \mathcal{N}(w_{j,k}^\ell|0, 1) \cdot \prod_{\ell=1}^{L} \prod_{j=1}^{J^\ell} \mathcal{N}(b_j^\ell|0, 1)$$

in which L is the number of hidden layers and J is the number of hidden units per layer. Using the prior and likelihood distribution, we wish to find a posterior distribution over network parameters θ, $p(\theta|y, x)$, which can be used in turn to predict the output.

The exact posterior distribution, in general, is intractable because of the activation functions of the neural network. To approximate the posterior distribution, we make use of Variational Inference (VI), a powerful tool to locate the parameters of the posterior in presence of model uncertainty. The assumptions used to form the approximate posterior are: (1) Each scalar parameter θ_j is independent of other weights, and (2) each scalar parameter θ_j is a Normal random variable.

The goal is to approximate the posterior using a "simple" variational distribution q for each example n:

$$q(\theta \mid \lambda) = q(\theta|m, s) = \left[\prod_{\ell=1}^{L} \prod_{j=1}^{J^\ell} \prod_{k=1}^{J^{\ell-1}} \mathcal{N}\left(w_{j,k}^\ell|\tilde{m}_{j,k}^\ell, (e^{\tilde{s}_{j,k}^\ell})^2\right)\right] \cdot \left[\prod_{\ell=1}^{L} \prod_{j=1}^{J^\ell} \mathcal{N}\left(b_j^\ell|\bar{m}_j^\ell, (e^{\bar{s}_j^\ell})^2\right)\right]$$

Here, each scalar weight has a real valued **mean** parameter $m^{\ell}_{j,k} \in \mathbb{R}$ and a **log-standard-deviation** parameter $s^{\ell}_{j,k} \in \mathbb{R}$ (combined are called λ). The objective function of VI is the **Evidence Lower Bound**, or **ELBO**, defined as [7]:

$$\mathcal{L}(\lambda) = \mathbb{E}_{q(\theta|\lambda)}\Big[\log p(y|x,\theta) + \log p(\theta) - \log q(\theta|\lambda)\Big]$$

We seek to maximize the ELBO with respect to $\lambda = [m, s]$ by repeatedly updating λ by the estimated gradient of ELBO multiplied by the learning rate. This is equivalent to learning a set of m and s values which make the approximate posterior as close as possible to our true intractable posterior.

Unfortunately, the gradient of this expected function with respect to $q(\theta \mid \lambda)$, is also intractable. We thus make use of the **reparameterization trick** to compute a stochastic, unbiased estimate of the gradient of the objective function[8].

To implement the reparameterization trick, a simple distribution $p(\epsilon)$ is sampled. Then, a deterministic, differentiable transformer function $t(\epsilon, \lambda)$ transforms $p(\epsilon)$ into the variational posterior, $q(\theta|\lambda)$. For the case of this project, q is a normal distribution. A simple normal distribution $p(\epsilon)$ is transformable to $q(\theta|\lambda)$ with a linear transformer $t(\epsilon, \lambda)$:

$$\epsilon \sim p(\epsilon) = \mathcal{N}(0, I)$$

$$\theta = t(\epsilon, \lambda) = m + s \times \epsilon$$

With this reparametrization, the ELBO function transforms to:

$$\mathcal{L}(\lambda) = \mathbb{E}_{p(\epsilon)}\Big[\log p(y|x, t(\epsilon,\lambda)) + \log p(t(\epsilon,\lambda)) - \log q(t(\epsilon,\lambda) \mid \lambda)\Big] = \mathbb{E}_{p(\epsilon)}\Big[f(t(\epsilon,\lambda))\Big]$$

This way, the gradient of an expectation of a function is replaced by the expectation of the gradient of the function, which makes it feasible to use Monte Carlo Integration to calculate an unbiased estimator of the the gradient of the ELBO function:

$$\nabla_\lambda \mathbb{E}_{q(\theta|\lambda)}[f(\theta,\lambda))] = \mathbb{E}_{p(\epsilon)}[\nabla_\lambda f(t(\epsilon,\lambda))]$$

$$\mathbb{E}_{p(\epsilon)}[\nabla_\lambda f(t(\epsilon,\lambda))] = \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda f(t(\epsilon^{(s)},\lambda)), \quad \epsilon^{(s)} \sim p(\epsilon)$$

To update the parameter, we use of classical stochastic gradient descent optimizer:

$$\lambda = \lambda + \alpha \nabla_\lambda \mathcal{L}(\lambda)$$

with α being the learning rate or step size at which objective parameters are updated.

The hyperparameters that we seek to optimize using a brute force algorithm are:

1. Number of hidden layers, L: we test NNs with 0, 1, 2, and 3 hidden layers
2. Number of hidden units per layer, J: we obtain results for 5, 10, 20, and 50 units
3. Learning rate, α: throughout this experiment we kept this value fixed at $10^{-}4$
4. Number of MC samples: throughout this experiment we kept this value fixed at 50.

3 Experiments

Preprocessed EEG datasets (N=7 participants) are used to train and test models. Given the anatomical and neurophysiological differences across participants, we train a new model for each participant given their Session 1 data. The EEG data for Session 2 is being considered for future testing of each participant's model, but we must first explore various architectures for training on the Session 1 set to yield maximal training and validation test accuracy. Each participant has up to 280 trials of EEG data for Session 1 and up to 100 trials for Session 2. The number of trials that each participant has depends on the trial survival rate during the preprocessing phase of data cleaning. Table A1 shows the number of usable trials for each participant in Sessions 1 and 2, as well as the corresponding Hit Rate (number of hits out of the total trials).

All trial data has been feature-extracted from the original time-amplitude EEG signals, resulting in a 3-D feature space composed of dimensions: 1) scalp region, 2) frequency band, and 3) time window, and whose values are spectral power measured in dB. Feature extraction has been currently configured to extract 2 non-overlapping time windows within the 2.4 second epoch of EEG data. For each window, at each of the 8 scalp regions, there are 7 frequency bins generated, where the spectrogram function in MATLAB computes the corresponding spectral power. The reshaped size of the feature extracted trial data is thereby 112 x 1.

Prior to training Bayesian neural network classifier models, trials are randomized such that there is no inherent order in the splitting of data for cross-validation training. The use of 3-fold cross-validation naturally yields three models during training, each of which are tested on the held-out validation set, where the classification accuracies of each model are averaged to describe the generalized model.

As a diagnostic for the evaluation of the model, we first train on a toy dataset. For this, we've generated 200 data points which each have two features. Half of these points are labeled as Positive and the other half as Negative. These class labels correspond to the Hit and Miss labels with respect to our binary memory state classification. Figure A1 shows a scatter plot of these data points with the horizontal axis being the value of the first feature, and the vertical axis being the value of the second feature. Notably, the feature space for these data points shows that there is visual separation of the two classes, albeit with several points caught in the intersection. As shown in Figure A2, upon training the BNN on this data, we see that the ELBO is quickly maximized and the L1 norm of the gradient steps minimized.

To demonstrate the performance of each model, we plot a model's specificity versus sensitivity as independent measures along the y and x axes, with the z-axis being the log of the product of the model's sensitivity and specificity. Figures A3 and A4 demonstrate these plots on the toy data for the SVM and BNN models respectively. Note that for this type of plot for any SVM model, there is only a single data point due to the deterministic nature of SVM, while there may be multiple data points for this type of plot for any BNN model. This is

due to the stochastic nature of the BNN, where the optimal q parameters from the final trained model are sampled 1000 times, thereby producing 1000 models, and then calculate the sensitivity and specificity of each model given the predictions on the held-out validation test set. Proper interpretation of these plots is guided by the fact that data points plotted towards the upper right corner (sensitivity=1.0, specificity=1.0) are models whose sampled parameters result in achieving 100 percent classification accuracy and whose log product is close to zero and plotted in red.

To evaluate the performance of BNN models on the EEG dataset, we identify several architectures for training: a no hidden layer model, a single layer hidden model with 10 units, a two layer hidden model with 50 and 20 units, and a three layer hidden model with 50, 20, and 20 units. The performance metrics for these architectures are presented in Table 1. We see that by increasing the network depth from no hidden layers to a single 10-unit layer, there is marginal increase in mean classification accuracy. Increasing further to a [50, 20] architecture, we observe a substantial increase in classification accuracy and mean sensitivity, but also a decrease in mean specificity. This higher sensitivity of 0.85 and lower specificity of 0.15 suggests that trials are often being predicted as Hits, regardless of the data. This tendency for models to be biased in their predictions can be a direct result of the imbalanced class-ratio of our data (i.e. more Hits than Misses) where during training, models learn to predict the most frequent class. This is not ideal, but it is a key explanation in understanding these results. By increasing the architecture size further to [50, 20, 20], we see that this biasing has potentially been reduced, where sensitivity and specificity have begun to balance out at the cost of reducing classification accuracy. Still yet, this architecture yields greater classification accuracy than models with no hidden layers as well as our single layer model. In Figure 1, we observe that the [50, 20, 20] model is converging well, where the ELBO is maximizing towards zero, but still has further yet to go. The L1 norm of the gradient appears to be jumping between zero and 50 due to the application of gradient clipping, where the clipping results in the jagged effect. In Figure 2, we see the placement of the SVM model in the sensitivity versus specificity space as it compares to the models sampled from the [50, 20, 20] architecture in Figure 3. Here, we see that there is no clear advantage to this architecture where we would hope for models to be placed in the upper right-hand corner of the plot, but instead the sampled models hover around the negatively sloped diagonal indicating chance-level predictions.

During experimentation, we began to troubleshoot additional reasons as to why classification accuracy hovered around chance. One potential, valid reason discussed was size of the feature space as too large relative to the number of trials. To reduce the feature space from 112 features while retaining crucial variability in the data, we applied principle components analysis (PCA) to the EEG data prior to training. This reduced the feature space to 41 features. To test any advantage of using PCA, we trained the same [50, 20, 20] architecture for comparison. In Table 1, we see that this method yielded 0.56 mean sensitivity and 0.43 mean specificity, indicating that there is little to no bias due to class-ratio imbalance, but this also indicates that the model is predicting correctly only at chance. Unfortunately, these results lead us to believe that classification may not be fully possible without more trials and more meaningful features.

ELBO and L1 norm of gradient during VI training of BNN with three hidden layers, [50, 20, 20] architecture

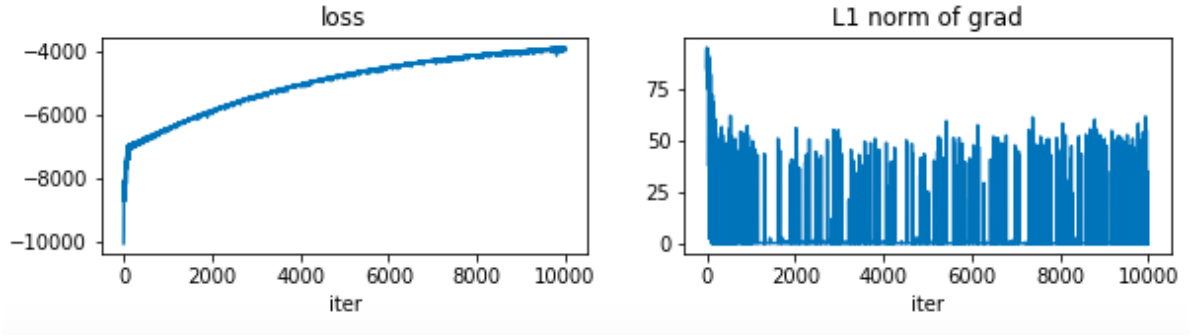

Figure 1: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

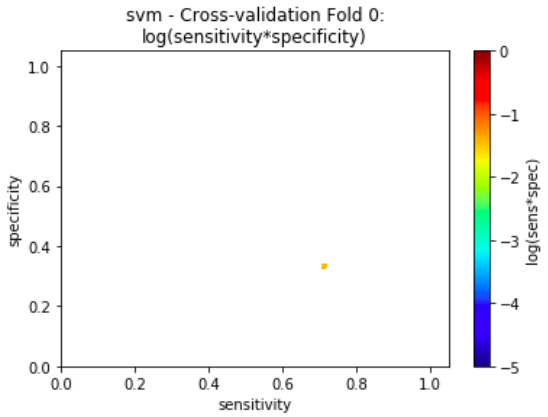

Figure 2: SVM model results in 0.74 mean sensitivity and 0.26 mean specificity

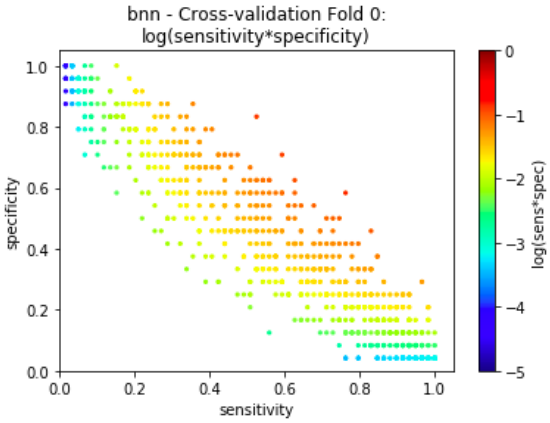

Figure 3: BNN model with [50,20,20] architecture classifies around chance accuracy with widespread sensitivity and specificity

Model / Architecture	**Classification Accuracy**	**Sensitivity**	**Specificity**
SVM	60%	0.74	0.26
[]	54%	0.63	0.35
[10]	55%	0.63	0.38
[50,20]	65%	0.85	0.15
[50,20,20]	58%	0.70	0.29
[50,20,20] (PCA)	52%	0.56	0.43

Table 1: Mean performance metrics for each architecture

4 Reflection and Outlook

4.1 Short-Term

Effective parallelization, which would solve our computational efficiency problem and greatly increase our ability to address other priorities, is our first short-term priority. Though our code does not have many loops, we suspect some of the imported functions we are relying on (e.g. `numpy`'s vectorize) disguise `for` loops in their backend.

Using some empirical feature extraction techniques on the EEG data set, we reduced the data dimensions from >24,000 features to 112. However, preliminary PCA results suggest that the feature space could be cut down even further (41 features with data variance > 80%). Our next short-term priority, would be a controlled experiment using PCA and perhaps VAE methods, to see if we can reduce overfitting and speed up the training procedure.

Moreover, we surmise that data imbalance is the reason both baseline and main models overfit. Thus, we started investigating some data balancing techniques including Bootstrap oversampling. Preliminary results are not encouraging, but we need to do more research.

4.2 Long-Term

Given a longer development and investigation period, we would focus more efforts on our analysis of the effects of network architecture on classification accuracy. Our analysis of this subject was limited by our model's lack of efficiency, and by a large class imbalance in our data. With more time, we'd prefer to create a controlled experiment varying both layer size and count in order to draw stronger conclusions about the optimal architecture.

Another long-term improvement would be our data division. Currently, our data is divided strictly by participant and by day: we have two days' worth of data for each of our seven participants. We have further divided this by using the first day for each participant as our training/validation set, and the second day as our testing set. Given more time, we would prefer to run experiments on different splits and make conclusions about the best method.

4.3 Takeaways

After working with Bayesian Deep Learning methods for a semester, our main conclusion is that they are not magic solutions guaranteed to outperform alternatives. Rather, BDL methods can produce great results when paired with a lot of dedicated setup, tuning, and analysis. The complexity of these methods should not be underestimated, and they require a great deal of adjustment. Ultimately, given the problem at hand, a simpler non-deep method may perform every bit as well. When using BDL methods, it's key to ensure that it makes sense given the domain at hand.

References

[1] E. Noh, G. Herzmann, T. Curran, and V. R. de Sa, "Using single-trial eeg to predict and analyze subsequent memory," *NeuroImage*, vol. 84, pp. 712–723, 2014.

[2] B. Richhariya and M. Tanveer, "Eeg signal classification using universum support vector machine," *Expert Systems with Applications*, vol. 106, pp. 169–182, 2018.

[3] X. Sun, C. Qian, Z. Chen, Z. Wu, B. Luo, and G. Pan, "Remembered or forgotten?—an eeg-based computational prediction approach," *PloS one*, vol. 11, no. 12, p. e0167497, 2016.

[4] Z. Wei, C. Wu, X. Wang, A. Supratak, P. Wang, and Y. Guo, "Using support vector machine on eeg for advertisement impact assessment," *Frontiers in neuroscience*, vol. 12, p. 76, 2018.

[5] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[7] R. Ranganath, S. Gerrish, and D. Blei, "Black box variational inference," in *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

A Extended Classification Results

	Session 1		Session 2	
Subject	Total Usable Trials	Hit Rate	Total Usable Trials	Hit Rate
1	248	70.97%	98	95.92%
2	251	39.44%	97	48.45%
3	288	59.03%	28	92.86%
4	269	58.74%	99	71.72%
5	284	63.03%	99	86.87%
6	279	62.01%	88	68.18%
7	247	31.98%	99	78.79%
Mean	266.57	55.03%	86.86	77.54%
St. Dev	17.77	13.97%	26.25	16.47%

Table A1: Summary of Participant Dataset Statistics

Diagnostic Results on Toy Data

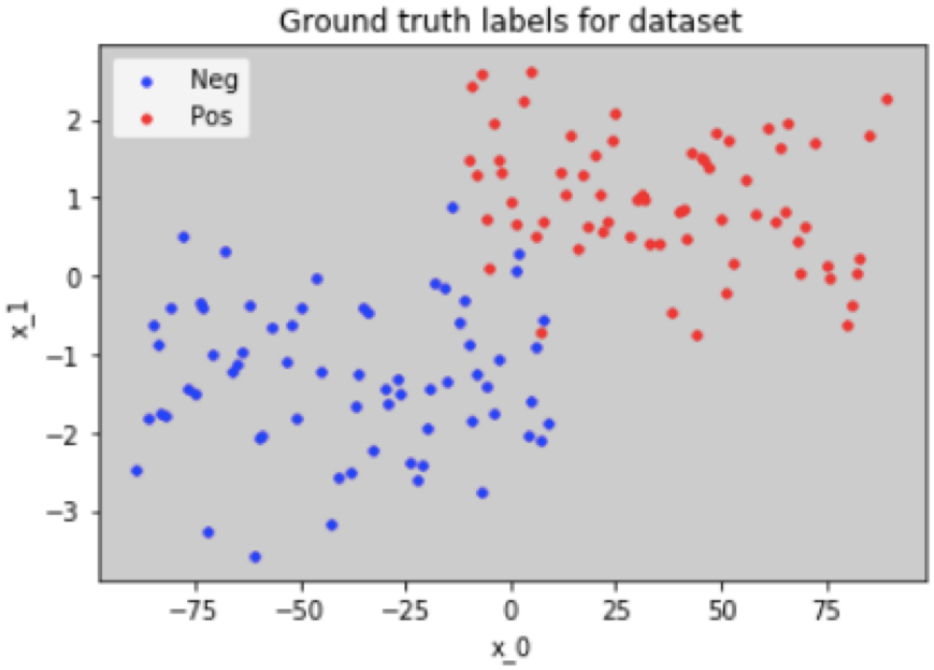

Figure A1: Scatter plot of semi-randomly generated toy data. 200 data points (Trials) with 2 features, evenly split between 100 Hits (positives) and 100 Misses (negatives).

ELBO and L1 norm of gradient during VI training on Toy Data

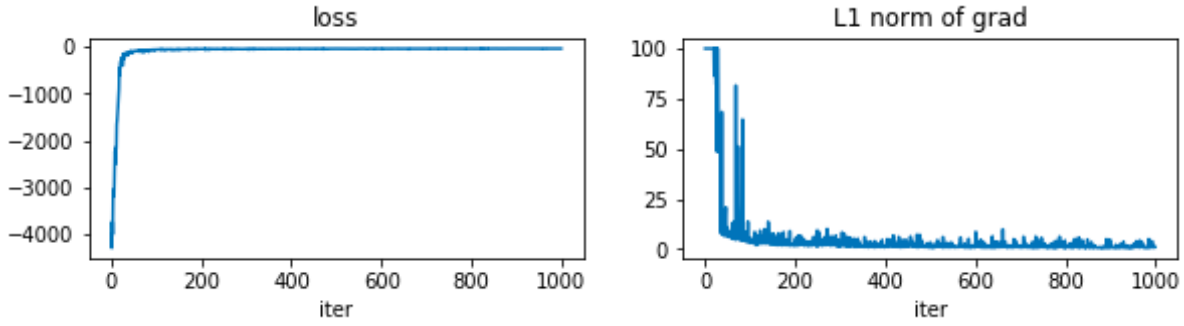

Figure A2: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

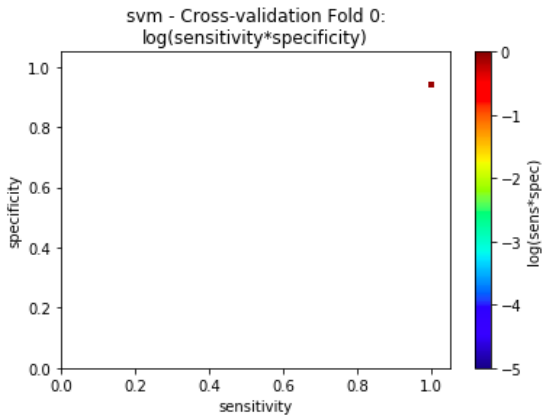

Figure A3: SVM model classifies toy data with high classification accuracy, sensitivity, and specificity

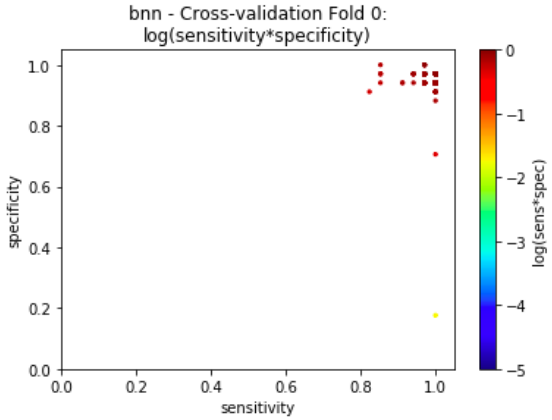

Figure A4: Models sampled from the trained parameters consistently yield high classification accuracy, sensitivity, and specificity

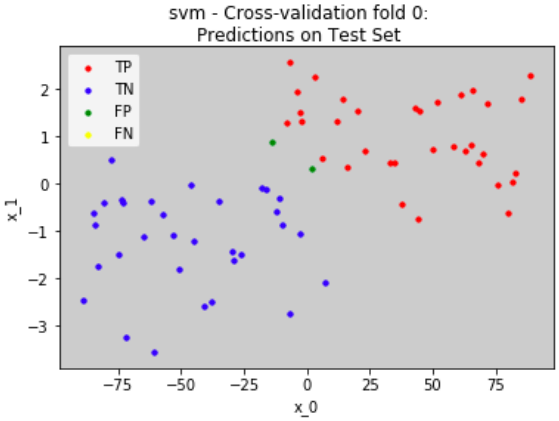

Figure A5: Predictions on the held-out validation test set. All but two data points are correctly classified.

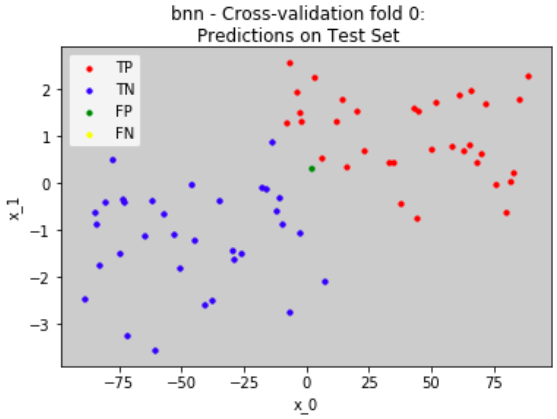

Figure A6: Predictions on the held-out validation test set using one of the sampled models. All but one of the data points are correctly classified.

ELBO and L1 norm of gradient during VI training of BNN with no hidden layers

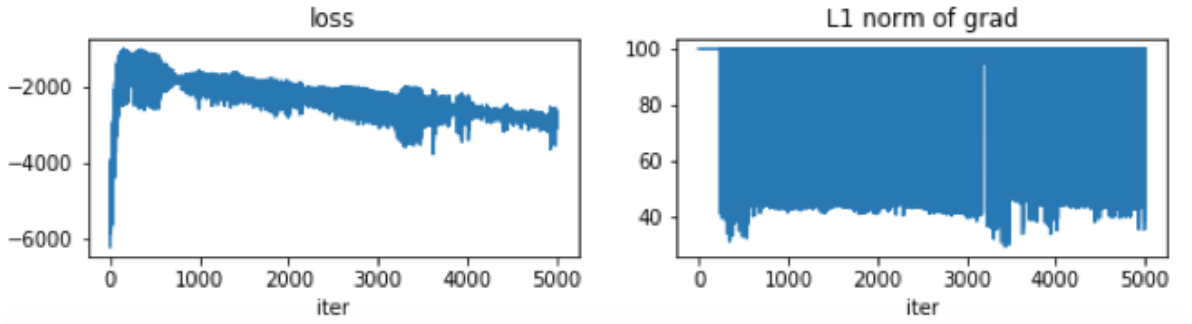

Figure A7: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

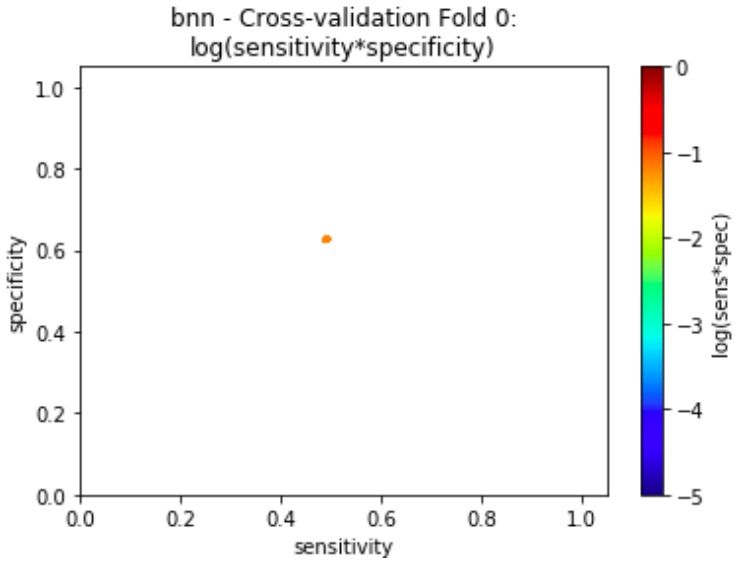

Figure A8: Models sampled from the trained parameters of the no hidden layer architecture show consistent sensitivity and specificity, both around 0.5, thereby maintaining an at chance classification ratio.

ELBO and L1 norm of gradient during VI training of BNN with single hidden layer, [10] architecture

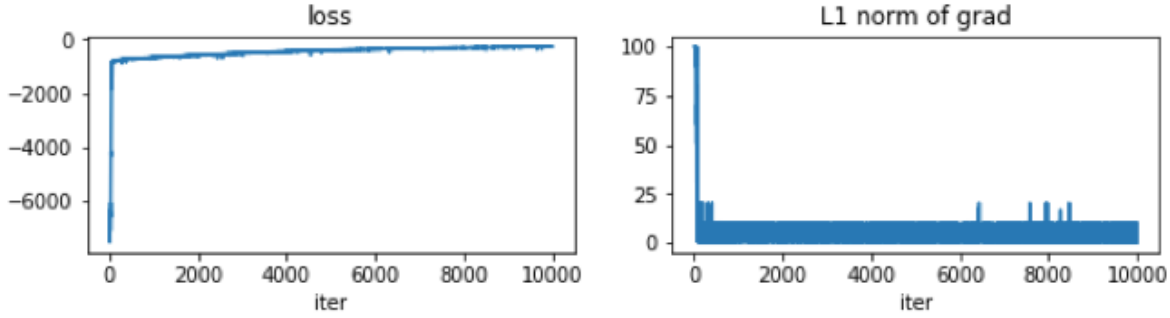

Figure A9: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

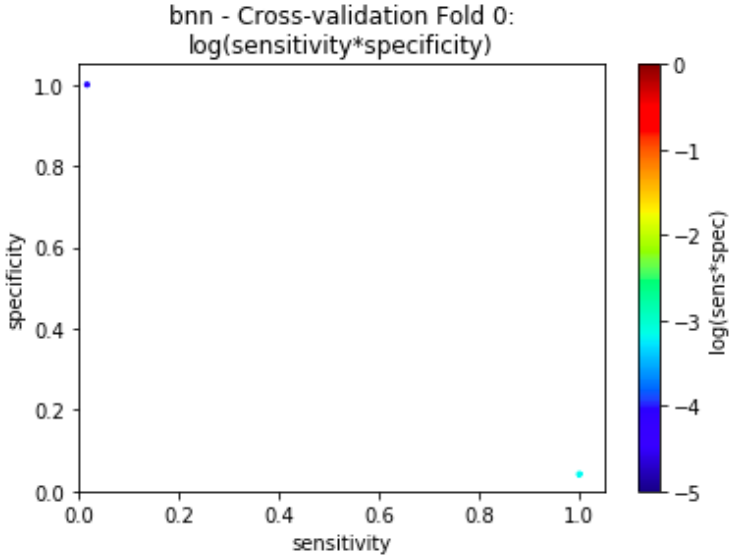

Figure A10: Models sampled from the trained parameters of the [10] architecture for cross-validation fold 0 show an outlier case where sensitivity is very low and specificity is very high, resulting in predicting all Misses.

ELBO and L1 norm of gradient during VI training of BNN with two hidden layers, [50, 20] architecture

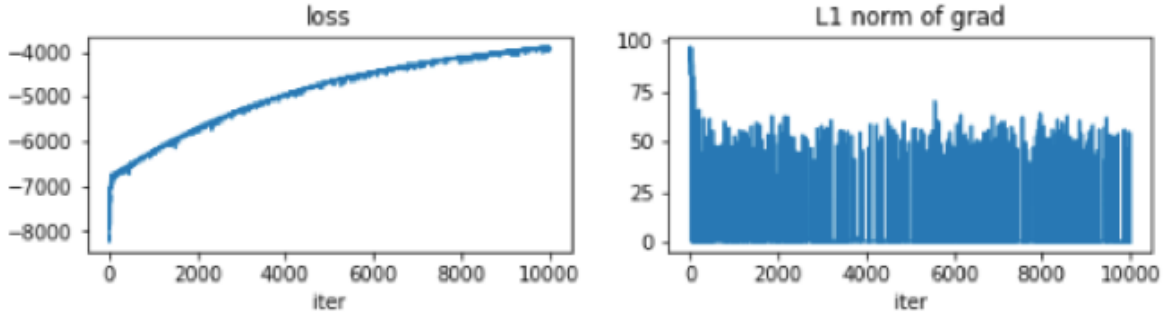

Figure A11: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

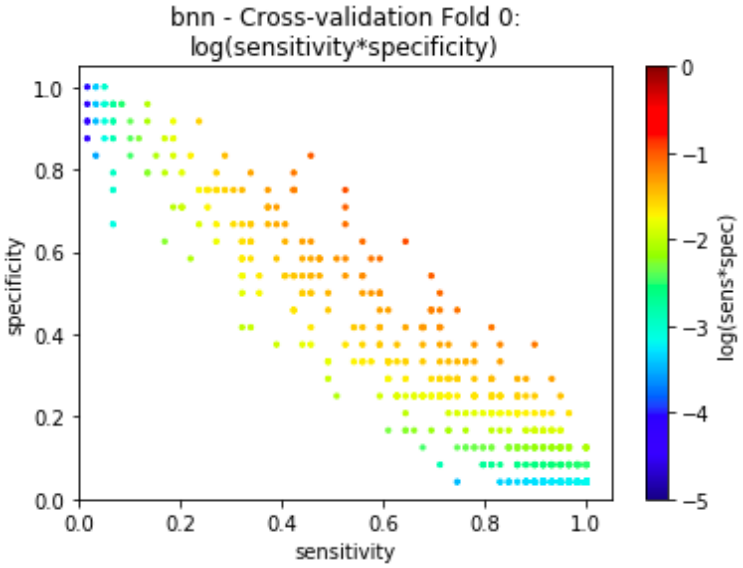

Figure A12: Models sampled from the trained parameters of the [50,20] architecture show dispersed ratios of sensitivity and specificity, but maintain an above chance classification ratio.

ELBO and L1 norm of gradient during VI training of BNN with three hidden layers, [50, 20, 20] architecture, on EEG data post-PCA

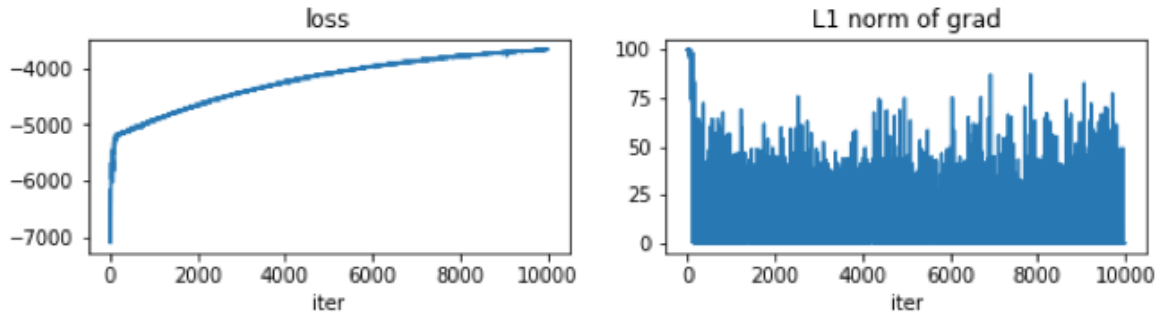

Figure A13: (Left) Maximizing of the evidence lower bound during across iterations of variational inference updating the parameters of the network. (Right) Minimizing of the L1 norm of the gradient steps across VI iterations.

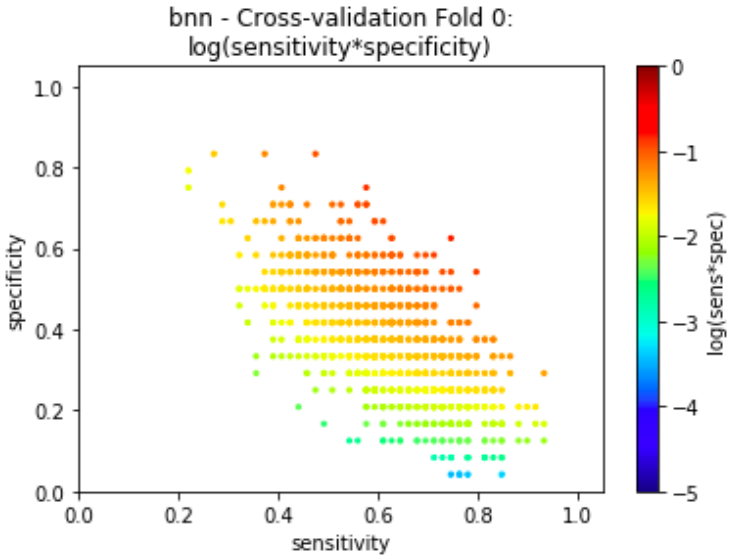

Figure A14: Models sampled from the trained parameters of the [50,20,20] architecture on post-PCA data show a constraint within a smaller sensitivity versus specificity space. The pattern of slightly higher mean sensitivity and slightly lower mean specificity suggests biasing towards Hits is still apparent.

B Receiver Operating Characteristic Curve

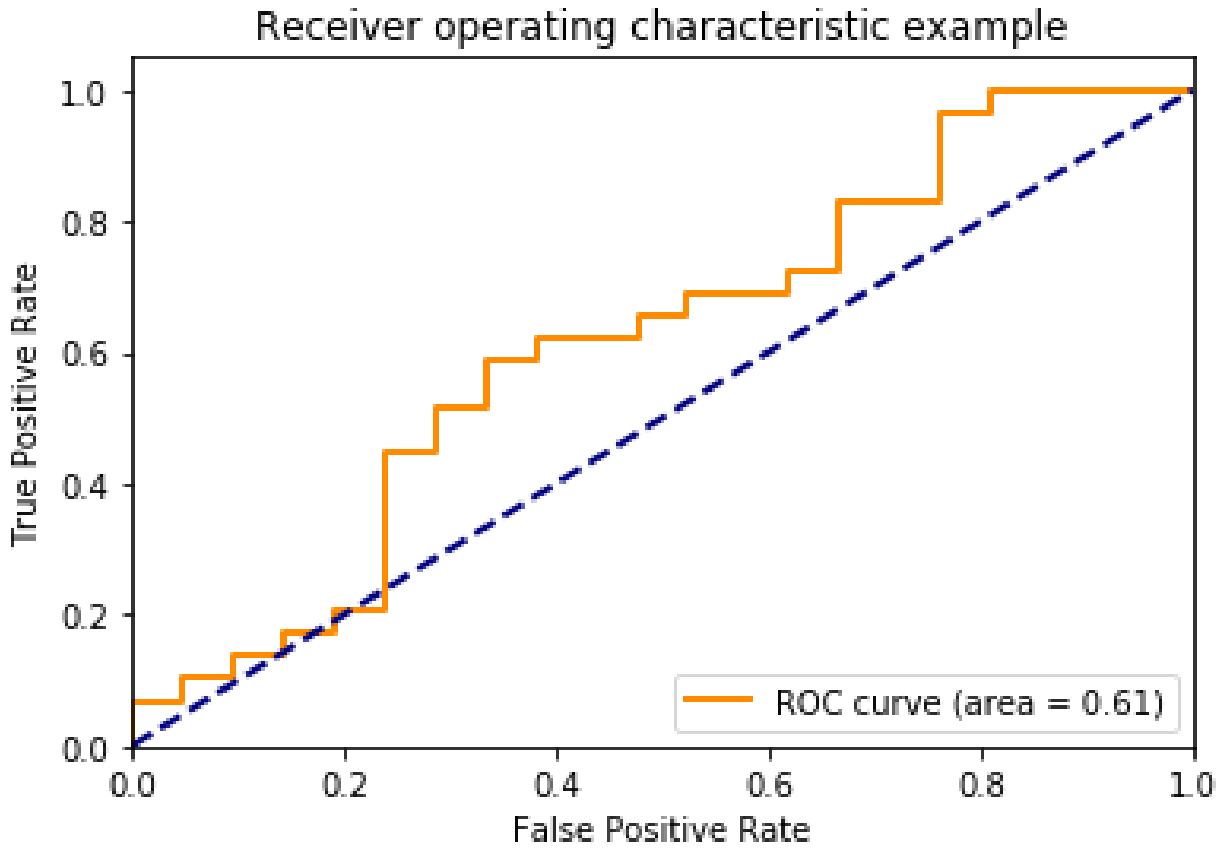

Figure B1: The plot of ROC curve for the SVM classification on the EEG data set. The underlying SVM model is using RBF Kernel with $\gamma = 0.1$ and $C = 1$.

As mentioned in Section 2, results for the baseline SVM model, used to classify the same data sets, are obtained by simply sweeping model parameters to find optimized model. Both Linear and RBF Kernel were tested with different regularization values. Also, γ parameter of RBF kernel was swept among limited number of options to find the best value. After all these experiments done, the best results achieved were reported for RBF Kernel with $C = 1$ and $\gamma = 0.1$. After all, the test accuracy achieved was 58% (for train it was 100%)and area under ROC curve is still very poor (61%).