university of groningen / faculty of science and engineering / artificial intelligence

Locating sources of uncertainty in an EEG classification task

Bachelor's Project Thesis

Martijn Wobbes, s4339312, m.s.wobbes@student.rug.nl,
Supervisor: I.P. de Jong

Abstract: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer interdum est non mi dictum, convallis finibus lectus venenatis. Suspendisse sed molestie lectus, in aliquam sem. Nulla nec porttitor quam. Curabitur neque tellus, facilisis a vestibulum ut, consectetur vitae mi. Duis auctor arcu eros, vel pretium ex ultricies ac. In eleifend, est ut porttitor semper, mauris risus ultricies turpis, at lobortis nibh magna sit amet nulla. Nam sit amet semper odio. Etiam rhoncus faucibus libero et dapibus. Nam nec eros ligula. Curabitur ut consequat tortor. Pellentesque eu mauris id tellus dictum congue. Quisque ex nulla, fermentum lacinia pretium quis, maximus a libero. Cras vehicula, quam vel convallis tincidunt, lacus purus venenatis ante, in pharetra nisi enim tincidunt lacus. Aliquam eu convallis ex. Praesent ullamcorper sit amet leo lobortis lobortis. Pellentesque id velit est. Curabitur ornare pulvinar dolor, ut porttitor dolor ornare non. Duis tempor euismod lacus. Aenean in metus scelerisque, sodales.

1 Introduction

Because of their high performance, deep neural networks have become the state-of-the-art method for various machine learning tasks. They are prevalent in computer vision, speech recognition, natural language processing, and the biomedical industry. However, because of their excellent performance, the outputs of these networks are often blindly trusted and presumed to be accurate, which is not always valid as the networks are generally overconfident. Generally, a network has no way of communicating how certain the model is with its predictions. Proper quantification of these uncertainties is becoming increasingly important in the practical field of machine learning. Uncertainty quantification gives the user a better insight into a model's prediction.

1.1 Uncertainty in machine learning

The uncertainty of a model consists of two types: aleatoric and epistemic uncertainty. Aleatoric uncertainty is the noise inherent in the observations, such as sensor noise. Feeding more data to the model will not help resolve this uncertainty. Epistemic uncertainty is the uncertainty in the model parameters itself. Feeding more data into a model can help resolve this type of uncertainty.

We can split aleatoric uncertainty further into two types. Homoscedastic aleatoric uncertainty is constant over all inputs. Heteroscedastic aleatoric uncertainty depends on the given input and differs between episodes. For example, homoscedastic aleatoric uncertainty can be sensor noise, constant across all episodes. In contrast, the heteroscedastic aleatoric uncertainty can only be a disconnected or faulty sensor in a few episodes.

1.2 Uncertainty quantification

Researchers have proposed various methods for the quantification of these uncertainties.

One of the most studied methods for modeling uncertainties is Bayesian Deep Learning Methods (Kendall & Gal, 2017; Chen et al., 2020). These models represent the weights as probability distributions rather than fixed values. The weights are treated as random values with prior distributions. After observing the data, the prior distributions are updated using Bayes' rule to obtain a posterior. The posterior can be used to make predictions, predicting a probability distribution rather than a

fixed estimate. The width of this final distribution indicates the uncertainty of the model.

Nix & Weigend (1994) suggested a different approach, initially designed for regression tasks. Their strategy was to use a non-bayesian network with two output heads. One predicts the target mean μ, and one predicts the variance σ with the variance indicating the uncertainty of the model. The model will learn to predict the variance indirectly with the loss function, with erroneous predictions getting punished less if the predicted variance is high. Using a sampling softmax and a different loss function can also be used for classification (Kendall & Gal, 2017).

A different method of capturing predictive uncertainty involved deep ensembles (Lakshminarayanan et al., 2017). The core idea behind deep ensembles is to have multiple models. With each trained independently on the same dataset. The variance between the different models can then be used to estimate the uncertainty. The variance reflects the level of disagreement between the models, which can be used to measure the overall uncertainty.

Monte Carlo Dropout is another method for predicting uncertainty (Gal & Ghahramani, 2016). This method sets random weights to zero, canceling out their contribution. Multiple passes will be done with random dropouts, resulting in an output distribution with which it is possible to calculate the uncertainty.

These methods can quantify uncertainty, each with its strengths and weaknesses. These methods can all answer the question of how uncertain a model is. However, they need to answer where this uncertainty comes from. Steps towards answering these questions can be made by introducing Explainable AI methods to the model. One approach is to integrate Shapley Values into the network, which approximates the contribution of features to the outcome (Merrick & Taly, 2020).

1.3 Error-related potentials

The medical field is one of the scientific fields that can benefit the most from uncertainty quantification. Generally, tasks such as medical diagnosis are of great importance and significance. However, data collection methods in this field, such as an electroencephalogram (EEG) or electrocardiogram (ECG), are very prone to noise. In the case of EEGs, artifacts, such as signals from blinking or malfunctioning electrodes, can easily corrupt EEG data.

BCIs are systems that use the brain's electrical activity to extract the intention from a user and translate it into computer instructions. These systems are helpful in fields such as prosthetics, where the user's intent can be translated into the movements of said prosthetic. Recently, error-related potentials (ErrP) have become a topic of interest. An ErrP signal is a response to error perception after feedback. These signals spark interest for BCIs since they indicate an erroneous extraction of intention, in which case the system can reconsider the translated action. Alternatively, the erroneous episode can be used as a training example to improve future performance.

Previous research has shown promising results in predicting ErrP signals (Correia et al., 2021). Specialized machine learning architectures designed explicitly for EEG signals have been designed and can be used for these tasks (Lawhern et al., 2018). However, these models all lack a form of uncertainty quantification.

1.4 Proposed method

In this paper, I propose a multi-headed Convolutional Neural Network classifying EEG signals on ErrP signals. The proposed model has two output nodes. One outputting the classification label and one outputting an estimate of the uncertainty. Furthermore, Shaply Values will be integrated into the model to indicate which features contribute the most to the uncertainty estimate, explaining the sources of uncertainty. The model will be trained and tested on the Monitoring Error-Related Potential dataset (Chavarriaga & Millán, 2010).

2 Methods

2.1 Dataset

The data used for this paper originates from the 'Monitoring error-related potentials' dataset, which was created by INSERT REF et al. This dataset is publically available on the BCNI Horizon 2020 project website. In this dataset, users were placed in front of a screen where they had to observe a

moving cursor. The working area of the screen consisted of 20 locations along the horizontal plane. A coloured square would appear either on the cursor's left or right and indicate the target to which the cursor should move. The cursor would move along the horizontal axis towards the target at each trial. Once the target has been reached, the cursor will stay in place, and a new target will appear along the horizontal plane, no more than three positions away from the cursor.

During the experiments, users were asked to solely monitor the agent's performance, knowing the goal of reaching the target. They had no control over the cursor. At each trial, there was a 20% chance for the cursor to move in the opposite direction relative to the target, which contradicts the agent's goal. These trials are labelled as 'Error-related potentials.'

Six users participated in this experiment, each performing two separate recording sessions. Each session consisted of 10 blocks, of 3 minutes each. Each trial consisted of approximately 50 trials. The EEG signals were recorded at a sampling rate of 512hz.

Since the users have no control over the cursor, the task is purely mental. This results in fewer signals originating from the motor complex being present in the EEG. However, the lack of physical movements also makes it easier for participants to lose focus and start mind wandering. This would inevitably result in more unwanted signals creeping into the EEG.

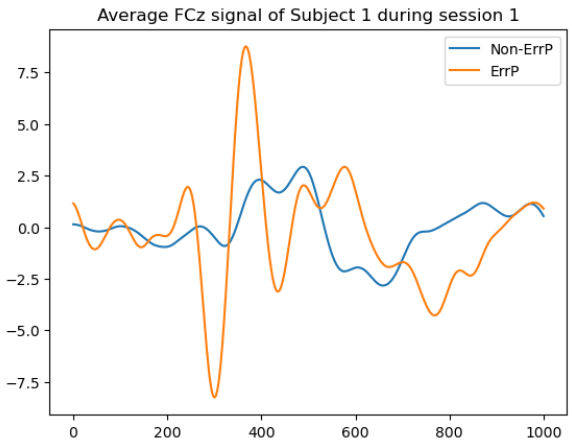

Figure 2.1: Graph of the average non-ErrP and ErrP signal in the FcZ channel. These signals originate from subject 1 during session.

2.2 Preprocessing

2.2.1 Filtering

EEG data is very prone to noise originating the environment. One such example is the frequencies originating from the power-line, which are 50 hz. To reduce this noise the raw data is being fed through a butterworth filter with a low-pass of 1hz, and a high pass of 10hz.

2.2.2 Epoching

The used dataset contained six different labels for trials. Two of those labels indicate that the cursor moved towards the target, which will be labeled as a 'non error-related potential'. Two other labels indicate that the cursor moved in the opposite direction compared to the target. These are labeled as 'error-related potentials'.These two labels will furtheron be referred to as `ErrP` and `non-ErrP` signals. The two remaining raw labels are ignored.

Figure 2.1 shows that the largest difference between `ErrP` and `non-ErrP` signals occur between 200 and 500 milliseconds after the feedback presentation. For this reason, the window size will be made 600ms, to capture these differences between signals. Note that this graph only shows the FcZ channel, but this channel is the most prominent for ErrP signals.

This resulted in the input EEG data being a matrix of size 64 x 308. Where the number of rows are the 64 channels of the EEG, and the number of columns are the length of the windows, 600ms of 512hz sampling rate.

The dicision of feeding al 64 channels into the model was made because of two reasons. First, previous experiments INSERT REF have shown that feeding all channels, as opposed to pre-selecting electrodes, resulted in consistently better accuracy. Second, the main aim of this research is to study uncertainty. Having all channels could present more interesting results with regard to the amount and origin of uncertainty, as opposed to only using a select amount of channels.

2.2.3 Balancing

Due to the nature of the experiment of the used dataset, the data is inherently unbalanced with a 1/5 ratio. Only 20% of the trials are `ErrP` trials. Rebalancing is necessary to prevent the model from predicting all input episodes as `non-ErrP`. To achieve this, in the training set, the under-represented class `ErrP` will be over-sampled until the two classes are represented equally.

To stick as close as possible to the real-world application of these classifiers, the dataset split for the training, validation and testing of the model was carefully considered. One participant is put aside for the testing set, and the remaining five participants will be used for training. Furthermore, 20 trials are randomly sampled from the training set to be used as a validation set. This results in approximately 66.7% of the data being used for training, 16.7% for validation, and 16.7% for testing, with the latter being a participant the model has not yet seen in training and validation.

2.3 Model

2.3.1 Model architecture

The main body of the model architecture consists of a model called EEGNet, a compact convolutional network tailored explicitly for BCI EEG classification INSERT REF. This model consists of two distinct sequential blocks.

The first block consists of two convolutional steps in sequence. The first layer applies F_1 convolutional filters of size (1, 64), which capture the EEG signal at different band-pass frequencies. Setting F_1 to half the sampling rate allows for capturing frequency information at 2Hz and above. Next, a `Depthwise Convolution` is applied to learn a spatial filter. After each convolutional layer, batch normalization is applied. Next, an exponential linear unit (ELU) is applied, followed by a Dropout layer to help regularize the model. Lastly, an average pooling layer is used to reduce the sampling rate to a quarter of the original sampling rate.

The second block consists of a Separable Convolution, which is a Depthwise Convolution followed by F_2 Pointwise Convolutions. These convolutions help reduce the number of parameters to fit and explicitly decouple the relationship across feature maps. This operation separates the learning of how to summarize individual feature maps in time from how to combine these feature maps optimally. An Average Pooling Layer follows this block to reduce the free parameters in the model.

In the original model, inputs are passed through the two blocks sequentially, followed by a flattened layer. This is followed by a linear layer, which gives the logits of the model's prediction. This output layer is where our model differs from the original mode. Rather than having only one layer returning the classification logits, our model uses two output layers called `heads`. The first head is a linear layer returning the logits representing the `mean` of the prediction. The second head is a linear layer followed by a softplus. This head returns logits representing the `variance` of the model. The softplus is necessary to make the variance positive.

2.3.2 Sampling softmax

The goal is to capture heteroscedastic aleatoric uncertainty in the model. Currently, our model predicts two tensors. One contains the prediction of the mean of both classes. The other contains the prediction of the variance of both classes. To achieve this capture of heteroscedastic aleatoric uncertainty, a gaussian distribution is placed over the unaries.

$$\hat{x}|w \sim \mathcal{N}(f^w, (\sigma^w)^2) \tag{2.1}$$

$$\hat{p} = Softmax(\hat{x}) \tag{2.2}$$

Here, f^w is the output of mean head with parameters w. σ^w is the output of the variance head with parameters w. The prediction consists of f^w, which is corrupted with Gaussian noise with a variance of σ^w. The corrupted vector is then squashed with the Softmax function to obtain a vector of probabilities.

Ideally, an analytical integral should be taken from this Gaussian Distribution. However, no such solution exists to achieve this. Therefore, an approximation of the integral has to be made. This is achieved through Monte Carlo integration. Here we sample from the aforementioned normal distribution, and apply the softmax function to each sample. Using these samples, we can calculate the mean and variance.

2.3.3 Loss function

This raises a question. How can we find the optimal model parameters θ, which results in the most accurate predictions of the mean, while simulatinously predicting a variance capturing the aleatoric uncertainty? The loss function should allow the model to reduce the received loss by predicting a high variance on incorrect predictions. However, it is undesirable if the model predicts high variance on all input sequences, and thus needs to be punished for predicting high uncertainty on correct predictions. One may ask how this can be achieved. Previous research by INSERT REF found such method to achieve this desired behavior. This method is called β-NLL.

$$\mathcal{L}_{\beta-NLL} := \sum_{i=1}^{n} \left(\lfloor \hat{\sigma}^{2\beta}(X) \rfloor NLL \right) \tag{2.3}$$

Here, the $\lfloor . \rfloor$ indicates the `stop gradient` operator. This makes $\hat{\sigma}^{2\beta}(X)$ act as a learning rate, which is dependant on the variance prediction. If β is set to a value of 0, the loss function behaves like a normal NLL loss. If the value is set to 0.5, interesting behaviour is observed. Now, all data points are weighted down by $\frac{1}{\sigma}$ (inverse standard deviation instead of inverse variance). Experiments conducted by INSERT REF, showed that $\beta = 0.5$ achieved the best balance between accuracy and log-likelihood.

To find the optimal parameters θ, the negative log-likelihood (NLL) is used:

$$NLL := \frac{1}{2} \log \hat{\sigma}^2(X) + \frac{\mathcal{L}_\theta}{\hat{\sigma}^2(X)} \tag{2.4}$$

This loss measures the decrepancy between the predicted distribution and the target distribution, assuming a Gaussian distribution with mean μ and variance σ^2. Here, the left term $\frac{1}{2} \log \hat{\sigma}^2(X)$ acts as a normalization factor for the NLL loss, which ensures the model learns to predict the best possible mean and variance parameters for the given data. In the right term, $\mathcal{L}_\theta$, is the loss function used nested in the NLL loss. This loss is divided by $\hat{\sigma}^2(X)$. By dividing thiss loss by the variance, we ensure that the NLL loss is sensitive to both the accuracy of the predicted mean μ, and the uncertainty estimation encoded in the predicted variance σ^2. A higher uncertainty estimation, results in a higher dividing factor, and thus a lower loss. The first term ensures the model doesn't always predict a high σ^2.

In the original paper, the task at hand was a regression task, and the used loss function $\mathcal{L}_\theta$ was the mean squared error loss (MSE). In our case, the task at hand is a classification task. For this, we will use the Binary Cross Entropy loss (BCE):

$$\mathcal{L}_\theta := -(Y \log \hat{\mu}(X) + (1-Y) \log(1-\hat{\mu}(X))) \tag{2.5}$$

The BCE loss measures the difference between the predicted probability distribution of the binary output variable, and the true probability distribution of said variable. The first term in this function, $Y \log \hat{\mu}(X)$, punishes the model when it predicts a low probability $\mu(X)$ for the positive class. Similarely, the second term punished the model when it predicts a high probability for the negative class. Together, these two terms ensure that BCE loss penalizes the model for making incorrect predictions.

The combination of these three elements, allow β-NLL to ensure that the model learns to predict an optimal mean μ, and a variance σ^2, capturing the aleatoric uncertainty of the model.

2.4 Explainable AI

To be able to understand what our model is doing, and explain where the uncertainty is originating from, we need to introduce a method of Explainable AI to our model. For this, a method based on shapley values will be used.

Shapley values are a concept which originate from the field of cooperative game theory. They provide a way of allocating a value generated by a group of players to each individual player. This strategy has been widely adapted to the field of machine learning. Here, they attribute the contribution of each individual input feature to the final prediction of the model. This resulting attribution method is called SHAP (SHapley Addaptive exPlanations)

The key idea behind SHAP values is to decompose the model's output into the contribution of each input feature, while also taking into account their relations with the other features. Exact calculations are practically impossible to calculate for larger inputs. Instead, the SHAP values are calculated by approximation. This algorithm involves

sampling a subset of the features, and computing the SHAP values for each sample. The final values are then averaged over all samples to obtain an estimate of the feature importance.

The advantage of using the SHAP approach, is its flexibility and generality. They can be applied to a wide range of models, including classification tasks. Moreover, they provide a rich and interpretable representation of the model's behaviour. Another advantage of SHAP values, is that we can differentiate between the two heads. Using SHAP, we can build understanding of how each feature contributed solely to the mean prediction. And how they contributed solely to the variance prediction. The latter being of extreme interest in explaining the origins of the uncertainty.

2.5 Experiment design

References

Chavarriaga, R., & Millán, J. d. R. (2010). Learning from eeg error-related potentials in noninvasive brain-computer interfaces. *IEEE transactions on neural systems and rehabilitation engineering*, *18*(4), 381–388.

Chen, W., Zhang, B., & Lu, M. (2020). Uncertainty quantification for multilabel text classification. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *10*(6), e1384.

Correia, J. R., Sanches, J. M., & Mainardi, L. (2021). Error perception classification in brain-computer interfaces using cnn. In *2021 43rd annual international conference of the ieee engineering in medicine & biology society (embc)* (pp. 204–207).

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, *30*.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., & Lance, B. J. (2018). Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, *15*(5), 056013.

Merrick, L., & Taly, A. (2020). The explanation game: Explaining machine learning models using shapley values. In *Machine learning and knowledge extraction: 4th ifip tc 5, tc 12, wg 8.4, wg 8.9, wg 12.9 international cross-domain conference, cd-make 2020, dublin, ireland, august 25–28, 2020, proceedings 4* (pp. 17–38).

Nix, D. A., & Weigend, A. S. (1994). Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (icnn'94)* (Vol. 1, pp. 55–60).