The quantification and tracing of uncertainty in an Error-Related potential classifier

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, created by Chavarriaga & Millán (2010). This dataset is publically available on the BCNI Horizon 2020 project website. For this experiment, users were placed in front of a screen where they had to ob-

serve a moving cursor. The working area of the screen consisted of 20 locations along the horizontal plane. A coloured square would appear on the cursor's left or right side. This square indicates the target the cursor should move towards. At each trial, the cursor would move along the horizontal axis towards the target. 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 indicate a different outcome than the participant anticipated for an 'Error Related Potential signal.'

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

Since the participants have no control over the cursor, fewer signals originating from the motor complex are present in the EEG signals, reducing the number of unwanted signals. However, the lack of physical movements, and the need for constant attention, also makes it easier for participants to lose focus and start mind wandering, inevitably resulting in more unwanted signals creeping into the EEG.

2.2 Preprocessing

2.2.1 Filtering

EEG data is very prone to noise originating from its environment. One such example is the frequencies originating from the power line. To reduce this noise, the raw data is fed through a Butterworth filter with a low pass of 1Hz and a high pass of 10Hz.

2.2.2 Labeling

The used dataset contained six different labels for trials. Two labels indicate that the cursor moved
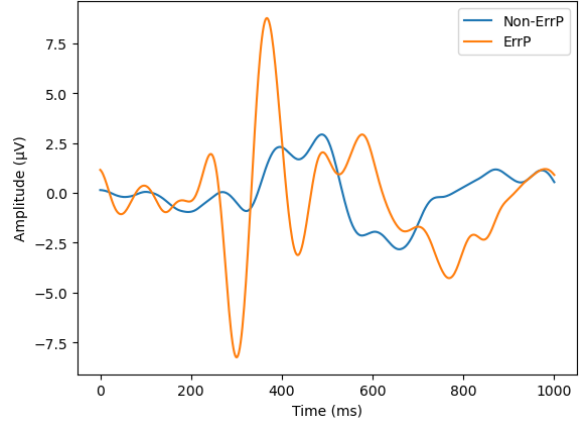

Figure 2.1: The average non-ErrP and ErrP signal on the FcZ channel. These signals originate from subject 1 during session 1.

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'. In the remainder of this paper, these two labels will be referred to as `ErrP` and `non-ErrP` signals. The two remaining raw labels are ignored.

2.2.3 Epoching

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

This window size results in the input EEG data being a matrix of size 64 x 308. The number of rows is the 64 channels of the EEG, and the number of columns is the length of the windows, 600ms of a 512Hz sampling rate.

The decision to feed all 64 channels into the model was made for two reasons. First, previous experiments have shown that feeding all channels instead of pre-selecting electrodes consistently resulted in higher accuracy (Correia et al., 2021). Second, the main aim of this research is to study uncertainty. Using all channels could present more

exciting results concerning the amount and origin of uncertainty instead of only using a select amount of channels.

2.2.4 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`. The under-represented class `ErrP` will be over-sampled in the training set until the two classes are represented equally.

The dataset split for the model's training, validation and testing was carefully considered to stick as close as possible to the real-world application of these classifiers. 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 (Lawhern et al., 2018). 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. These blocks are 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 layer. This head returns logits representing the `variance` of the model. The softplus layer 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 vectors. 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 method called `Sampling Softmax` is used (Kendall & Gal, 2017). With this method, we have to interpret the model as if it had a Gaussian distribution placed over it:

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

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

Here, f^w is the output of the 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 approximation 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 used architecture raises a question. How can we find the optimal model parameters θ, which results in the most accurate predictions of the mean while simultaneously 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 Seitzer et al. (2022) found such a method to achieve this desired behavior. This method is called β-NLL:

$$\mathcal{L}_{\beta-NLL} := \mathop{\mathbb{E}}_{X,Y} \left[\lfloor \hat{\sigma}^{2\beta}(X) \rfloor NLL \right] \quad (2.3)$$

Here, the $\lfloor . \rfloor$ indicates the `stop gradient` operator. This operator makes $\hat{\sigma}^{2\beta}(X)$ act as a learning rate, which is dependent on the variance prediction. If β is set to 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 by Seitzer et al. (2022) showed that $\beta = 0.5$ achieved the best balance between accuracy and log-likelihood.

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

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

This loss measures the discrepancy between the predicted and target distributions, 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 this loss by the variance, we ensure that the NLL loss is sensitive to 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 does not always predict a high σ^2.

In the original paper, the task 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):

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

The BCE loss measures the difference between the variable's predicted probability distribution and the variable's true probability distribution. 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. Similarly, the second term punishes the model when it predicts a high probability for the negative class. These two terms ensure that BCE loss penalizes the model for incorrect predictions.

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

2.4 Explainable AI

To understand what our model is doing and explain where the uncertainty originates 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 originating from the cooperative game theory field. They provide a way of allocating a value generated by a group of players to each player. This strategy has been widely adapted to the field of machine learning. They attribute each input feature's contribution to the model's final prediction. This resulting attribution method is called `SHAP` (SHapley Adaptive exPlanations)

The key idea behind SHAP values is to decompose the model's output into the contribution of

each input feature while also considering 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 estimate the feature importance.

The advantage of using the SHAP approach is its flexibility and generality. They can be applied to various 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 an understanding of how each feature contributed solely to the mean prediction and how they contributed solely to the variance prediction. The latter is of high interest in explaining the origins of the uncertainty.

2.5 Experiment design

This section describes the designs of the three experiments and the training methods used in these experiments.

2.5.1 Artificial noise

To properly test whether our suggested approach captures aleatoric uncertainty, we need a method of artificially introducing noise to our data to invoke this uncertainty. Optimally, in the bane of BCIs, we would like to add noise to the data, which frequently occurs on EEG data. However, artificially adding physiological artefacts, such as muscle movements, is beyond our reach and could be an exciting topic for further research. For this paper, we keep the artificial noise simple. The first method of artificial noise generation is adding Gaussian noise to our data. Each data point, within the desired range, will receive noise sampled from a Gaussian distribution of varying intensity. The second method of applying artificial noise is designed to simulate faulty or improperly connected sensors. This method will set a random amount of data points to zero within a specific range.

Figure 2.2 shows an example of artificially generated noise. The green signal is the original FcZ
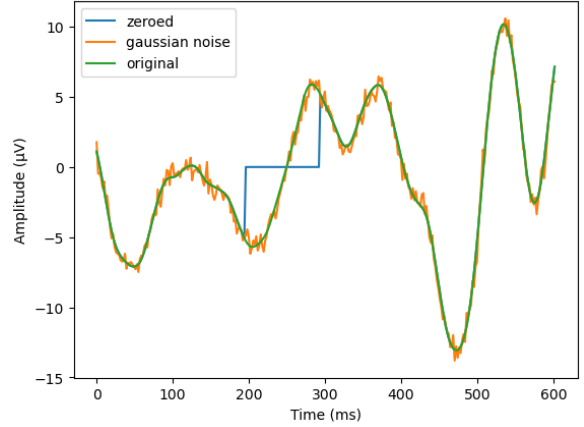

Figure 2.2: FcZ signal with artificially introduced Gaussian noise and localized zeroing.

signal, whereas the orange signal has artificially introduced Gaussian noise across the entire signal, with a standard deviation of 0.5. The blue signal shows an artificial local zeroing between 200 and 250ms.

2.5.2 Experimental setup

Each experiment uses the same hyperparameters optimal for EEGNet, suggested by Lawhern et al. (2018). Each training run will last for 50 epochs. This amount showed to be a good balance between performance and computational time. Each training session is averaged over a total of six runs. Each run uses a different participant as the test participant. The training was done using an NVIDIA GTX 1070 and took around 30 minutes per training session of six runs with 50 epochs each.

2.5.3 Effect of general noise

The first experiment adds noise to the entire input length, increasing the noise amount and intensity with every training session. It is expected that the addition of noise should result in a higher overall uncertainty compared to the clean data. The increase in the amount and intensity of the noise between training sessions is also expected to increase the predicted uncertainty between these sessions.

This experiment will accompany a test where labels are shuffled with increasing intensity. Here it

is also expected that the uncertainty increases linearly with the number of shuffled labels.

2.5.4 Effect of localized noise

The second experiment will test whether the location of the added noise affects the predicted uncertainty. Two training sessions will be completed here, each with a different input data section corrupted. Both sections have an equal length and intensity of noise. One section will be located on the area with the most contribution to input data, which will be located using the SHAP approximation on the mean head. The second location will be an area with the most minor contribution to the output. Adding noise to the high-contribution areas is expected to result in a more substantial increase in uncertainty than adding noise to the low-contribution areas.

2.5.5 Tracable noise

For the final experiment, we test whether we can trace the uncertainty through the SHAP approximation. Here, we add localized noise to some regions of the input data. Afterwards, the SHAP approximation is taken and compared to the SHAP approximation of the original input data. We expect that a shift in the SHAP values is noticeable. Ultimately, the area where we added noise should have higher uncertainty than the other areas and can be located by plotting the SHAP values.

3 Results

3.1 Base variance predictions

Figure 3.1 shows a histogram with the variance distribution on correct and incorrect predictions. From this histogram, one can observe that the model's classification is almost always correct when it is paired with a low variance prediction. The higher the model's predicted variance becomes, the more often the classification is incorrect. Although the two histograms overlap, there is a clear skew towards higher variances for the incorrect predictions. On the highest variance predictions ranging from 0.24 to 0.25, the distribution between correct and incorrect classifications is nearly equal, indicating
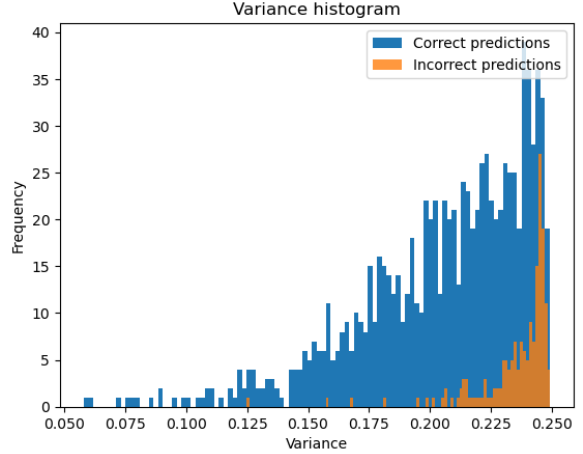

Figure 3.1: Histogram with the frequencies of variance predictions for correct and incorrect predictions.

that the model is randomly guessing the classification.

3.2 General noise

Figure 3.2 shows the effect of shuffling the labels of training and testing data on the model's predicted variance. Here, it is clear that there is an increase in variance as more labels in the training and testing set are shuffled. However, this difference is not very large, as there is only an increase of approximately 0.027, with relatively high variance between runs. However, the overall increasing behavior of the variance's head output is as expected. It indicates that the model is working as intended and is predicting the variance based on the quality of the input.

Figure 3.3 shows two measurements of the model. The two leftmost plots show the predicted variance of the model, whereas the two rightmost plots show the model's accuracy. The top plots show the behavior when Gaussian noise is added to the entire length of the signal, whereas the bottom plots show the behavior when channels are entirely zeroed. The x-axis here represents the `severity`, which indicates the strength of the noise. Higher values indicate that the added noise is of higher magnitude (in the case of Gaussian noise), occurring in more channels per episode and affecting more episodes in general.

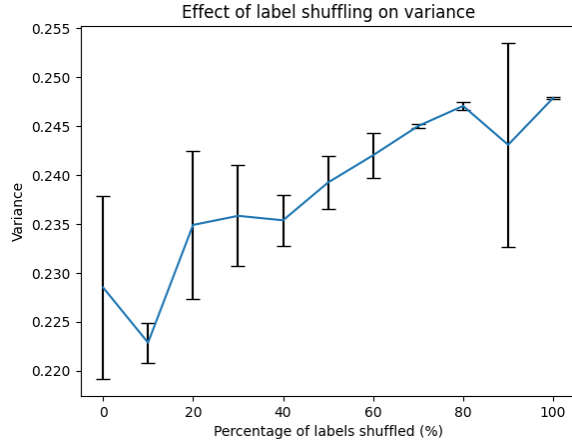As can be seen from these plots, there is a clear

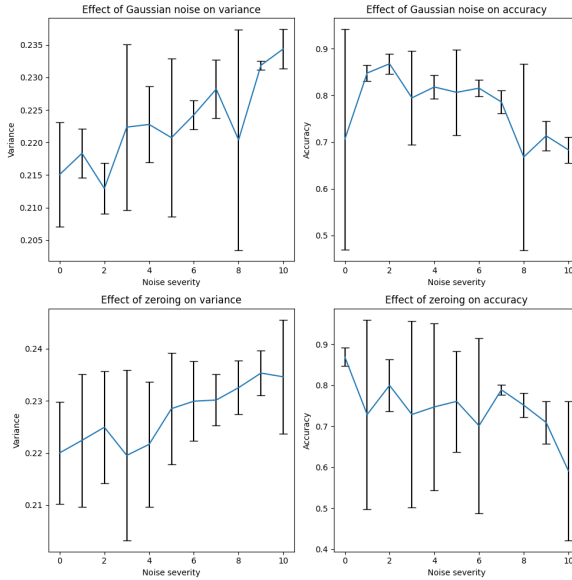**Figure 3.2: The effect of label shuffling on the output of the variance head.**

Figure 3.3: The effect of different intensities of artificial noise added to the entire signal length on the model's accuracy and variance.

increase in the predicted variance as the severity of the noise increases. This increase is present for the Gaussian noise and zeroing methods. Secondly, there is a clear decrease in accuracy as the severity of the noise increases. This behavior is also present for both types of noise. However, like the previous experiment, it is paired with a high variance between runs.

3.3 Localized noise

Figure 3.4 shows the influence of the location to which the noise has been added on both the variance and accuracy of the model. The leftmost plots show the predicted variance of the model, whereas the rightmost plots show the model's accuracy. The top plots show the behavior when Gaussian noise is added to the data, whereas the bottom plots show the behavior when channels are zeroed. The leftmost column shows the effect of the noise being added on the 0-250ms region, whereas the rightmost column shows the effect of the noise being added on the 250-500ms region.

As can be observed from these plots, the general trend for both models is that the variance is higher when noise is added to the 250-500 ms region than the 0-250 ms region. The accuracy is slightly lower when the noise is added to the 250-500 ms region than the 0-250 ms region. It also observed that zeroing has a stronger effect on the variance and accuracy than Gaussian noise. This behavior is according to expectation since a Gaussian noise corrupted signal still resembles the original signal to some extent, regardless of the degree of Gaussian noise. Moreover, when averaging many signals corrupted with Gaussian noise, the noise will balance each other out. In contrast, the zeroing of signals does not resemble the original signal. The observation that noise on the 250-500 ms region substantially influences the accuracy and variance compared to the 0-250 ms region is as expected, as previous research showed that the most prominent part of the `ErrP` signal occurs within this window.

However, the differences between these two regions in accuracy and variance are minimal and are similar to the previous experiments, paired with high variance between runs.

3.4 Tracable noise

Figure 3.5 shows the single-dimensional heatmaps of the Shapley values of the predictions of the variance head. Here, the Shapley values are depicted as a continuous color palate referring to the contribution of the data points. Lighter colors (yellow) indicate a strong positive contribution to the prediction, and darker colors (blue) indicate a strong negative contribution to the variance prediction. Here, the Shapley values are averaged over all channels.
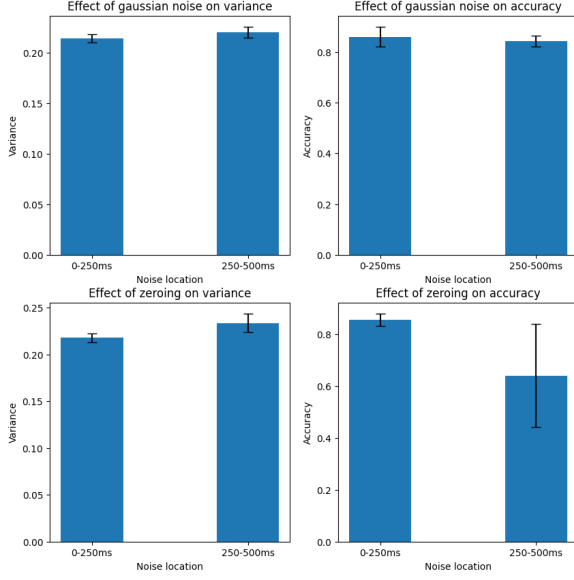
Figure 3.4: **The effect of strong noise added to two different regions of the signal on the accuracy and variance of the model. The two regions tested are the ranges of 0ms to 250ms and 250ms to 500ms.**

Figure 3.5: **A one-dimensional heatmap of the SHAP values of the variance had averaged over all channels on the original signal, a signal with noise in the 0-250ms region, and a signal with noise in the 250-500ms region. Lighter colors indicate a strong positive contribution, whereas darker colors indicate a strong negative contribution.**

Comparing these single-dimensional heatmaps yields various observations. Firstly, there is a noticeable difference between the Shapley values of the original signals and the signals with artificial noise added to them. In the case of the Gaussian Noise variant, it is clear to which area the noise has been added due to the oscillating behavior of the Shapley values. This oscillating behavior originates from the Gaussian noise adding large differences between adjacent data points. This oscillating behavior will most likely disappear when computing and averaging the Shapley values over more episodes since the noisy signal will average to the original signal. However, the computational cost of approximating these Shapley values is very high. Thus this plot only shows an average of 10 test episodes. When ignoring this oscillating behavior and judging only from the contribution distribution, one cannot find the location of the noise present in the signal. The contribution distribution between the data with noise on the 0-250ms and 250-500ms regions is nearly indistinguishable.

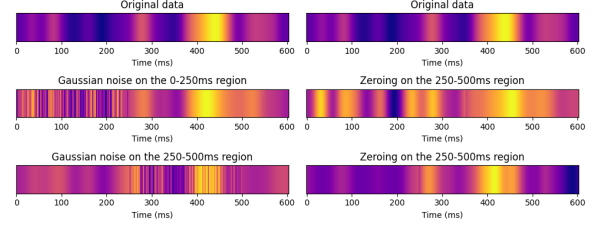As for the signal with the artificially zeroed channels, much of the same behavior is present. There is a clear distinction between the Shapley values of the original and modified signals. In this case, there is a noticeable difference between the two noisy signals, and in the case of the zeroing being added to the 0-250ms region, a higher contribution to the variance is present in this region. However, this behavior is unclear on the signal zeroed on the 250-500ms region.

4 Discussion

In this section, the results of the aforementioned experiments will be discussed, and problems with the current experimental setup and points for improvement, aswell as further research will be adressed

4.1 Uncertainty quantification

The first experiment showed that the model predicts high variance on incorrect classifications. With the extremely high variances pointing towards random guession of the model. This observation allows for an interesting implementation of a sanity check for the model. This does not only apply for these specific EEG and BCI classification models, but to classification models in general. One could put a threshold on the maximum variance which is allowed in a prediction. All classifications with a predicted variance below this threshold are accepted, whereas all classifications with variance exceeding this threshold are rejected. This can drastically improve the accuracy of the model. However,

9

this does come at the cost of rejecting some classifications, which would need manual classification for example. This can be a worthwhile trade-of for classification tasks in which false positives and false negatives have to be avoided at all costs, such as medical classifications.

The experiment in which the labels were shuffled showed an increase in the predicted variance as the percentage of labels shuffled increases. The second experiment, which shows that the addition of noise, in the form of Gaussian noise and the zoering of channels, on the entire length of the signal results in an increase in variance. These two experiments give a good indication that the model is behaving as expected, and is actively basing its variance prediction on the quality of the input data, and is capturing the artifically added heteroscedastic aleatoric uncertainty.

The third experiment shows that the noise location affects the predicted variance, with noise on the 250-500ms region resulting in higher variance and lower accuracy than noise on the 0-250ms region. However, this difference is small. Previous research shows that this was to be expected. Experiments with varying lengths of the input windows showed that there was only a 6% increase when increasing the window from 300ms to 600ms after the event's presentation (Correia et al., 2021). Thus, the model was already capable of predicting the label accurately. However, the difference in accuracy and variance between noise in the two regions confirms that the model shifts its attention to the $250 - 500$ ms region when it has access to it, since noise in this range has a stronger effect. This aligns with the prominent features of the Error-related potential.

From the the localized and global noise experiment, one can observe that the type of noise added to the signal matters in the variance and accuracy of the model. This is to be expected. Zeroing has a larger effect, since the zeroed signal has no resemblance to the original signal, and thus the original signal cannot still effect the model. Whereas in Gaussian noise, the signal still has some resemblance to the original signal, regardless of the intensity of the Gaussian noise. The original signal will still effect the model.

4.2 Uncertainty traceability

The fourth and final experiment shows whether the origin of noise can be traced down by using explainable AI, in the form of Shapley values. From this, we can observe that there is a clear difference between the oriinal signals, and the signals with artificially added noise. There is also a clear indication that the type of noise has an effect on the resulting Shapley values, with zeroing having an overal smoother uncertainty distribution, whereas the Gaussian has a rougher uncertainty distribution. Although there is a clear difference between all these signals, it is nearly impossible to accurately distinguish the location of the noise, purely based from the Shapley values. The difference between the original and the two noisy signals is easy to spot for the Gaussian noise, butt he difference between the two noisy regions is almost impossible to distinguish.

4.3 Problems with experimental setup

As one may have noticed in this paper, the dataset consists of data from six different participants. One may have also noticed, that thus far we have been training on the same participants (2-6), and test on the same participant (1). This was done deliberately. EEG signals are prone to a very high inter-participant variability. Due to this high variability, and only having six participants, testing on all participants was not a viable choice due to the low consistency in results between participants. Due to this high variance it was impossible to gather concise results, and the difference between training sessions are much larger than the already quite large difference that is experienced with only one test participant.

4.4 Noise

The two methods of generating artificial noise, Gaussian noise and the zeroing of channels, used for the experiments are only rough simulations of noise and could be more realistic. The noise used in these experiments, which showed results are magnitudes higher than what is encountered in real-life applications. A more realistic approach would be adding noise variants that are more common in EEG sig-

nals. One such approach would be the recreation of physiological artifacts, such as the signals originating from the motor complex for actions like blinking. This method drastically increases the experiment's complexity and is way beyond the scope of this research paper, but it can potentially be interesting to research in the future.

4.5 Model

For this experiment, I put the focus on a single model: EEGNet, developed by Lawhern et al. (2018). This model is a compact Convolutional Neural Network (CNN). However, this is not the only model architecture which is commonly used for EEG classification. One of these architectures which is commonly used is the Recurrent Neural Network (RNN). It is possible that the localization of the noise can be achieved when using the other architectures, since they observe and train from the data in different ways, and thus can be an interesting research for the future.

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).

Seitzer, M., Tavakoli, A., Antic, D., & Martius, G. (2022). On the pitfalls of heteroscedastic uncertainty estimation with probabilistic neural networks. *arXiv preprint arXiv:2203.09168*.