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

From Brainwaves to Actions: Evaluating Uncertainty in CNNs and Riemannian Geometry Models for BCI

Bachelor's Project Thesis

Joris Suurmeijer, s4334124, j.m.suurmeijer.1@student.rug.nl,
Supervisor: Ivo de Jong

Abstract: Brain-computer interfaces (BCIs) capture brain signals and transform them into functionally useful output. In the field of Motor Imagery, a Machine Learning model can learn from the captured data to predict the imagined movement of a user. This study aims to guide researchers in their choice for such a Machine Learning model, by comparing the capabilities of Deep Learning and non-Deep Learning models in terms of their classification performance and Uncertainty Quantification (UQ). Convolutional Neural Networks (CNNs) are used as the Deep Learning model and a Minimum Distance to Riemannian Mean (MDRM) model is used as the non-Deep Learning model. The UQ methods used are Deep Ensembles and DUQ for CNNs, and two distance-based uncertainty methods for MDRM. Results show that Deep Ensembles have better accuracy than the other methods, whereas MDRM models offer better UQ despite lower classification performance. DUQ performs worse than Deep Ensembles and MDRM in both areas.

1 Introduction

The field of Brain-computer Interfaces (BCIs) has recently gained substantial attention because of the development of implants like N1 from Neuralink, which is an array of electrodes inside a person's brain to read their brain signals. Brain-computer interfaces are systems that measure brain activity and convert it (nearly) real-time into functionally useful output (BCI-Society, 2024).

A BCI is, for example, used in the field of Motor Imagery, where participants imagine a movement and a BCI captures their brain activity. BCIs are important in this field because they can provide a way for people with heavy motor disabilities to control a certain device using their brainwaves (Barachant et al., 2010). How this works is that, based on the captured brain signals, a prediction can be made about which movement the person imagined and this prediction can be given as a command to, for example, a robotic device. To make such a prediction, a threshold must be determined to distinguish a movement of the right hand from a movement of the left hand, for example. This threshold can be learned by a Machine Learning (ML) model.

In the field of Brain-computer Interfaces, two different types of ML models are used: Deep learning models and Non-Deep learning models. Both types of models, however, lack research in the domain of Uncertainty Quantification (UQ) for tasks related to BCIs (de Jong et al., 2023).

The field of UQ focuses on the confidence of ML models when making predictions. Confidence here is the predicted probability by the model of how likely it estimates its prediction to be correct. The goal of using UQ is to assess whether the confidence of a ML model aligns with the actual accuracy of the model. For example, if a ML model predicts a right-hand movement with a 60% confidence, this means that the model thinks that the chance that this prediction is right is approximately 60%. If the model, on average, has a confidence of 70% in its predictions, then it should be right in 70% of the predictions. To assess if this is the case, the level of calibration can be measured, which compares the confidence of the ML model to its actual accuracy. This comparison to the accuracy is important be-

cause it is desirable for a ML model to be confident in its predictions when its classification performance is strong, and to exhibit low confidence when it is likely to make incorrect predictions. Without this, it is hard to incorporate additional observations, such as human verification, to check the predictions when the ML model lacks confidence.

An example of how UQ can make an important difference is to make a ML model that is able to withhold from making a prediction when it is too uncertain about the input it got (for example for out-of-distribution data). It is, for example, undesirable for a ML model to predict that a patient has a disease if it is not certain that a patient has this disease, as this causes unnecessary harm to the patient. In the context of BCI for Motor Imagery, UQ is important because this can help prevent a device from making a movement when the model used is not certain about the prediction. It is necessary to prevent this, because otherwise a person using a BCI could think of something unrelated and the device will make an unwanted movement, although it was not sure if the person even was thinking about a movement. UQ can help prevent these unwanted predictions when we set a boundary to not perform a movement, unless the model is, for example, 90 percent sure that it is correct. But for this to work the model's confidence should follow its actual accuracy, otherwise it will not perform properly. This alignment of confidence and accuracy is what we will measure.

In this study, a Convolutional Neural Network (CNN; LeCun et al., 1989) will be used as the Deep learning model. Specifically for the CNN, two uncertainty methods will be used: Deterministic Uncertainty Quantification (DUQ; Van Amersfoort et al., 2020) and Deep Ensembles (R. T. Schirrmeister et al., 2017). For the Non-Deep learning model, a Riemannian geometry model will be used, more specifically, a Minimum Distance to Riemannian Mean (MDRM; Barachant et al., 2011) model.

DUQ is a distance-based uncertainty technique, which projects samples to a high-dimensional space. In this high-dimensional space, a distance function can be used to get the distance from a sample to a learned class centroid. Based on this distance the confidence can be calculated for this prediction. This will be explained further in Section 2.5.

Deep Ensembles is an uncertainty technique in which multiple models are trained. The variance of the models tells us something about the uncertainty. The confidence value of a prediction is taken by calculating the mean of the confidence value of all the individual models. This will be explained further in Section 2.5.

For the MDRM model, two different methods to estimate the uncertainty will be used. Both of the uncertainty methods work similarly to DUQ, as the distance to learned class centroids in a high dimensional space will be used. The difference between the two uncertainty calculation methods for MDRM will be explained in Section 3.2.2.

The reason DUQ is chosen to be included in this study is because DUQ works similarly to MDRM with respect to uncertainty. Therefore, comparing both methods is interesting as this will say something about the difference between the models themselves, and thus about Deep Learning and non-Deep Learning models, instead of solely saying something about the way uncertainty is calculated. Deep Ensembles is chosen because it is an approximation of Bayesian Neural networks and it's been shown to get better predictions compared to alternative methods (Manivannan et al., 2024), like MC-Dropout (Gal & Ghahramani, 2016), MC-DropConnect (Mobiny et al., 2021), and Flipout (Wen et al., 2018). Including Deep Ensembles is good for the comparison between Deep Learning and non-Deep Learning models, because this way it can be concluded if a difference between the type of models is the result of the Riemannian geometry or Deep Learning aspect, or because of the UQ strategy that is used.

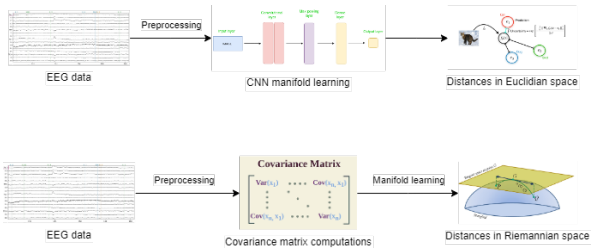


Figure 1.1: Overview of the difference between CNNs and Riemannian geometry models for Uncertainty Quantification

In Figure 1.1 an overview is shown of how CNNs and MDRM work with respect to uncertainty quantification. In Section 2 this will be further explained

and compared to each other.

This paper compares the different types of ML models with their uncertainty techniques to guide researchers in selecting the appropriate model for a BCI task. For this comparison, two categories will be used: First, the classification performance on a BCI task, and second, the UQ performance on the same task. To make this comparison on the two categories, both the classification performance and the UQ performance of the models should be measured. As explained, for uncertainty quantification the level of calibration will be used. To calculate the calibration of the model, the confidence of the model is compared to the observed accuracy, using multiple metrics. More information on this can be found in Section 4.1.

The hypothesis regarding the comparison is that MDRM and DUQ will perform similarly with regard to uncertainty quantification, while Deep Ensembles will perform differently. This is expected because the uncertainty in MDRM and DUQ is calculated similarly, while Deep Ensembles has a very different method to estimate uncertainty.

2 Background

2.1 Brain-computer interfaces

There are three types of BCIs: invasive, non-invasive, and partially invasive. Invasive BCIs are placed inside the skull, directly on the brain, which requires surgery. Non-invasive BCIs are placed on the scalp and do not require surgery to be placed. Partial invasive BCIs have the device implanted inside the skull but outside the brain, which also requires surgery (Alharbi et al., 2023).

In this study, EEG data from non-invasive BCIs is used. This means that the data that is used in this study is recorded from the scalp. The spatial resolution for EEGs is worse than other neuroimaging techniques (Alharbi et al., 2023), due to poor signal-to-noise ratio (Liu et al., 2020). This means that it is hard to know whether a signal came from an area deep in the brain or near the surface. However, the benefits of not requiring surgery outweigh the worse spatial resolution of the data, since there is a lower risk for participants. This results in a lower barrier of entry for participating in an experiment and for using a BCI.

2.2 Motor Imagery

BCIs have shown useful in studies about the human brain (Abdulkader et al., 2015). One example of a sub-field where they are used is Motor Imagery. In Motor Imagery tasks, participants imagine a movement without performing it. The BCI captures the brain signals, which can be used to predict the imagined movement using for example a ML model.

Barachant et al. (2010) described that a BCI might become helpful in the field of Motor Imagery by providing a new way of non-muscular communication for people with heavy motor disabilities, like in Spinal Cord Injury or Locked-In Syndrome. A ML model can translate brain signals into a command, which can be given to an external device (Barachant et al., 2010).

EEG-based BCI systems can have a lot of uncertainty because the EEG signals can have poor signal-to-noise ratio, small amplitudes, and high variability within and between subjects and sessions (Milanés-Hermosilla et al., 2021). This makes it important for a model to emphasize its uncertainty in these cases, as this can make it possible to withhold from performing an undesired movement (for example, moving your hand when you do not want this). In this study, all datasets that are used belong to the field of Motor Imagery.

2.3 Convolutional neural networks

CNNs are a type of feedforward neural network that learns features from the data by using filters, which are also called kernels (LeCun et al., 1989). Kernels are small matrices that can be used to look at specific features in a small piece of the input data. Each kernel can detect specific features, such as edges, textures, or patterns. The kernels are used by 'sliding' over the data piece by piece (e.g. in an image it slides over the pixels) to learn translation-invariant features. By extracting these features, the model can learn information that is relevant for predictions.

CNNs consist of multiple layers which all have a specific function. The convolution layers are the most important, as they apply the filters to the input sequence to extract meaningful features. How this works is that the kernels are moved over the input and the weight for each kernel (meaning

how much influence this kernel has in the process) is learned when training the model. Because of the feature extraction mechanism, CNNs perform particularly well in tasks where pattern detection is needed, such as image processing. For example, in the popular MNIST dataset (Deng, 2012), a CNN can extract features like the brightness of pixels or edges of the figure. These features can help with predicting which digit the picture represents.

In the context of a BCI task, a CNN mostly has two types of convolution layers: one convolution layer that uses kernels to slide over the temporal axis and one convolution layer that does the same for the spatial dimension of the input data. The temporal filter is therefore able to learn patterns anywhere in the sequence, while the spatial filter performs a convolution across channels to learn meaningful combinations of channels. The spatial filter however does not really 'slide' over the data as explained, because it has the same length as the number of channels because the order of the channels is arbitrary.

Multiple Convolutional Neural Network structures are used in the field of BCIs. The most used ones include Shallow Convnet (R. Schirrmeister et al., 2017), Deep ConvNet (R. Schirrmeister et al., 2017), and multiple models from the EEG net family (Lawhern et al., 2018). The best-performing CNN model with EEG data, between the just mentioned models, is a Shallow ConvNet (Köllőd et al., 2023). The evidence the paper provides was that Deep ConvNet and Shallow ConvNet showed the most improvement in accuracy relative to chance level. Both Shallow ConvNet and Deep ConvNet outperformed the other model on one of the tasks, but the paper states that on half of the datasets, there was an insignificant difference. Therefore, in this study, it was chosen to use the Shallow ConvNet model, as it is a less complex model.

In Figure 2.1 the architecture of the Shallow ConvNet is depicted. Here you can see the temporal and spatial convolution layers as discussed, and a Mean Pooling layer, which downsamples the data, without losing important information (Ali et al., 2020).

Next in the structure follows a Dense layer and Softmax layer, which ensure that a prediction is done based on a probability distribution over the classes, made by the Softmax layer.

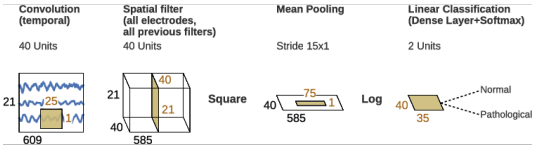


Figure 2.1: Image from R. T. Schirrmeister et al. (2017), showing the architecture of a Shallow ConvNet

2.4 Riemannian geometry models

In Riemannian geometry, covariance matrices are used. In the context of BCI, each timestamp in the EEG data represents a sample, and the channels of the EEG data correspond to the dimensions of the covariance matrix. The covariance matrix shows the variance and correlations between the different channels. The set of all covariance matrices forms the Riemannian manifold. We can 'walk' over this manifold to get the distances in this high dimensional space, with the distances being invariant so it should be more generalized. For a Riemannian geometry model with Motor Imagery, a mean covariance matrix is computed for every class (for example, a two-class task with right-hand vs left-hand has a covariance matrix for both). The mean of all the covariance matrices which belonged to a certain class in training, form this learned class mean or centroid. This centroid represents the class on the Riemannian manifold. An example of the computed covariance matrices can be seen in Figure 2.2.

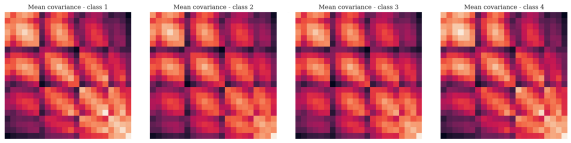


Figure 2.2: Example of the covariance matrices for each class, computed by the MDRM model

A Minimum Distance to Riemannian Mean classifier is used as the Riemannian Geometry model. This model is chosen because it is a simple model that shows good generalization compared to other algorithms (for example CSP spatial filtering algorithm with an LDA classifier; Barachant et al. 2011). The training of this model consists of estimating the mean covariance matrix for each class.

The model makes a prediction by computing a covariance matrix for an input and comparing this covariance matrix to each learned mean covariance matrix for a class. This comparison is done based on the distance between those covariance matrices, which is calculated using the following formula:

$$\delta_g(C_1, C_2) = ||log(C_1^{-1/2}, C_2, C_3^{-1/2})\|_f = \sqrt{\sum_{n=1}^{N}(\log^2 \lambda_n)}, \tag{2.1}$$

where, as explained by Congedo et al. (2017), λ_n are the N eigenvalues of matrix $C_1^{-1/2}, C_2, C_3^{-1/2}$ or, equivalently, matrix C_1^{-1}, C_2 and where in both expressions the indices 1 and 2 can be permuted, as the distance is symmetric (Congedo et al., 2017).

Riemannian models are often used in BCI tasks because they are relatively simple to implement and perform well (Congedo et al., 2017). Recently, Riemannian geometry models have claimed first prices in BCI classification competitions, thus showing competitive classification performance (Yger et al., 2016).

2.5 Uncertainty quantification

Uncertainty quantification is the study of the uncertainty of computational systems. Neural network models often make predictions without indicating if they are confident about their decision, even for data that is vastly different from the training data. Especially in data that differs from most training data, this is a problem because it causes generalization errors. This can be tackled by having a ML model show its uncertainty.

The field of uncertainty quantification aims to create more interpretable models, by providing a measure of confidence in the predictions of the models (Abdar et al., 2021). This helps people understand when not to trust a model blindly, but could also be used to stop a model from predicting if it is uncertain.

Several methods help us add uncertainty quantification to a ML model. As mentioned in Section 1, the UQ methods chosen for the Deep Learning models are Deep Ensembles (Lakshminarayanan et al., 2017) and Deterministic Uncertainty Quantification (DUQ) (Van Amersfoort et al., 2020).

For the MDRM model, the distances are used as an uncertainty measure. There has been little research on Riemannian models with uncertainty. However, this approach is in line with the probabilistic approach by Barthélemy et al. (2023).

2.5.1 Deep Ensembles

Deep Ensembles work by having multiple models make a prediction on the same sample and look at the differences (variance) between the predictions of the models. Then the confidence of the model can be calculated to know if it is sure in its predictions most of the time.

The confidence of a prediction is obtained by taking the maximum value of a softmax function for every model and then the mean of all these values is calculated. The overall confidence of the model can be calculated by taking the mean of all the prediction confidences, which is calculated like this:

$$p_e(y \mid x) = M^{-1} \sum_{i=0}^{M} p_i(y \mid x), \tag{2.2}$$

where $p_i(y \mid x)$ is the probabilistic predictive distribution over the labels, where y are the features output and x the input features, and M is the number of models that will be trained.

2.5.2 DUQ

DUQ works by learning a centroid for each different class, which represents that class as a point in a high-dimensional space. What distinguishes the DUQ model from the normal Shallow ConvNet architecture is the use of a Radial Basis Function (RBF) Layer. This RBF layer is responsible for projecting the input to a high-dimensional space. In this high-dimensional space, the distances from one point to another can be calculated. The output of the RBF layer consists of the distances from this input point to all class centroids in the high dimensional space.

These outputs are calculated using equation:

$$K_c\left(f_\theta(x), e_c\right) = \exp\left(-\frac{n^{-1}\|W_c f_\theta(x) - e_c\|_2^2}{2\sigma^2}\right), \tag{2.3}$$

where, as described by Van Amersfoort et al. (2020), f_θ is the model which has input dimension m and output dimension $d : \mathbb{R}^m \to \mathbb{R}^d$, and parameters θ. e_c is the centroid for class c, which is a vector of length n, W_c is a weight matrix of size n

(centroid size) by d (feature extractor output size), and σ is the length scale. This function is also referred to as an RBF kernel (Van Amersfoort et al., 2020).

In the original paper, the uncertainty of the model is calculated using the distance to the closest centroid (Van Amersfoort et al., 2020). However, in this study, the distances are normalized and then transformed into a probability-like distribution over the classes. This transformation to a probability is done because it is more interpretable than a distance, as it sums to a value of one. More information on our implementation will be given in Section 3.2.1.

2.5.3 UQ for MDRM

The uncertainty in an MDRM classifier works similarly to that in DUQ, since an MDRM also makes use of distances to learned class means (in this case the distance to each mean covariance matrix, as depicted in Equation 2.1 from Section 2.4). Since the model outputs the distances to each class mean, the distances are first normalized and then converted into a probability-like distribution over the classes, following the same approach as DUQ. For the calculation of the probabilities, there are two distinct methods used. These methods are further explained in Section 3.2.2

3 Methods

3.1 Datasets

For this study, data in the field of Motor Imagery was used. The datasets that were used were available through Mother Of All BCI Benchmarks (MOABB; Aristimunha et al. 2023). Multiple datasets were used to add validity to this study since the use of multiple datasets makes it less likely that the results are the consequence of a specific characteristic of a dataset used.

In this study the following datasets are used: a Motor Imagery dataset from Steyrl et al. (2016), a Motor Imagery dataset from Zhou et al. (2016), Dataset IIB from BCI Competition 4 (Leeb et al., 2007), and Dataset IIa from BCI Competition 4 (Tangermann et al., 2012). All ML models in this study use 70% of the data as training data, 10% as validation data, and 20% as test data.

In Table 3.1 an overview of the different datasets and their specifications can be seen. None of the participants were paralyzed in the datasets that were used. More information on the datasets can be found in the original papers, or the dataset documentation on MOABB.*

All data is preprocessed by using a single non-causal (one pass forward, one pass backward) IIR Band-pass filter, which uses a high pass filter with a cutoff frequency of 7.5 Hz and a low pass filter with a cutoff frequency of 30 Hz. The filtering is done because for Motor Imagery only the alpha (8 - 12 Hz), beta (12.5 - 30 Hz) and mu (7.5 - 12.5 Hz) bands were relevant (Scherer & Vidaurre, 2018), all other bands could be filtered out. The mu frequency is not always included in Motor Imagery tasks. However, in this study, it was included. We decided to include the mu frequency because it is strongly related to the motor cortex in the brain (Niedermeyer & da Silva, 2005).

3.2 Machine Learning Models

The ML models used in this study are trained in a BCI Motor Imagery task, where participants took one or more trials in a Motor Imagery task, as is discussed in Section 3.1. All code from this study is publicly available to use. †

3.2.1 Shallow ConvNet

As mentioned, two versions of the Shallow ConvNet model were used in this study (DUQ and Deep Ensembles), both based on the structure proposed by R. T. Schirrmeister et al. (2017).

Both models follow the structure from the original paper, which means that the following settings are used:

Both Shallow ConvNet models make use of a dropout layer, with a dropout rate of 0.5 as this follows the original paper by R. T. Schirrmeister et al. (2017). For both of the models, the labels of the data are transformed into categorical labels using a Label Encoder.

In addition, both ML models use early stopping, which prevents the model from overfitting

*https://moabb.neurotechx.com/docs/dataset_summary.html

†All code for this project is available at: https://github.com/Jorissuurmeijer/UQ-motor-imagery

Table 3.1: Summary of Datasets

Dataset	Subjects	Sessions	Runs	Classes	Additional Info	Channels	Number of trials
Motor Imagery dataset from Steyrl 2016	14	1	8	Right hand, Feet	The participants were aged between 20 and 30 years, 8 naive to the task, and had no known medical or neurological diseases.	15	17920
Motor Imagery dataset from Zhou 2016	4	3	2	Left hand, Right hand, Both feet	The intervals between two sessions varied from several days to several months	14	11496
Dataset IIB from BCI Competition 4	9	5	1	Left hand, Right hand	Right-handed subjects with normal or corrected-to-normal vision. Only channels C3, Cz and C4 were used	3	32400
Dataset IIa from BCI Competition 4	9	2	6	Left hand, Right hand, Feet, Tongue	Recorded on two different days	22	62208

	Original paper
pool_size	1, 75
strides	1, 15
conv filters	1, 25

Table 3.2: The Shallow ConvNet structure parameter values used in this study, from the paper by R. T. Schirrmeister et al. (2017)

(Makarova et al., 2021). The patience for early stopping is set to 20. This value was chosen because, when training the model, we noticed that a too-low value caused worse results (the training stopped before the ML model was fully converged). The early stopping mechanism that is used, ensures that the weights from the best run are restored, based on the validation loss.

Shallow ConvNet: Deep Ensembles

For the model that uses the Deep Ensembles method, the structure is the same as in the original paper (R. T. Schirrmeister et al., 2017). The output layer is a softmax layer that provides a probability-like distribution over all classes. The Deep Ensembles method uses an ensemble size $M = 10$.

The optimizer used by the model is an Adam optimizer with a learning rate of 0.001. This value was selected to use because it performed well on the classification performance of the used datasets. The loss we used is the categorical cross-entropy loss. This loss was chosen because it works well on categorical tasks (Brigato & Iocchi, 2020). Each model was trained for a maximum of 100 epochs if it was not stopped early by the Early stopping mechanism.

Shallow ConvNet: DUQ

For the DUQ model, the difference in structure is that the model does not have a softmax layer as the output layer. Instead, the model uses an RBF layer that predicts the distances to class centroids. The RBF layer as used in this study is an implementation of the Keras-Uncertainty library (Valdenegro, 2023). The RBF layer uses a length scale of 0.2, as this is the value in the range suggested by Van Amersfoort et al.(2020) which gave good classification performance on the used datasets.

The optimizer used by the model is an Adam optimizer with a learning rate of 0.01. The learning rate was changed since DUQ took a long time to converge and the value of 0.01 made the convergence easier. Because of the convergence problem, we also increased the number of epochs to 200 epochs (instead of the 100 used for Deep Ensembles), resulting in a possible longer training time (as long as the validation loss was decreasing, otherwise the Early Stopping mechanism intervened). The loss we chose is the binary cross-entropy loss because it follows the original paper by Van Amersfoort et al.(2020).

As mentioned in Section 2.5, the output layer gives the distances to the different class centroids. In this study, we then normalize the distances and then transform them into a probability-like distribution. The transformation to a probability-like distribution over the classes is done using a softmax function with temperature. This can be seen in the following equation:

$$\text{softmax}(x_i) = \left[\frac{\exp(x_i/T)}{\sum_j \exp(x_j/T)} \right]_i \quad \forall i \in [0, C-1], \tag{3.1}$$

where the logits xi are divided by a temperature factor T, and T is a vector of length C that is optimized to improve the calibration of the model. This has the effect of softening the output distribution for increasing T and it can improve calibration (de Jong et al., 2023). The value of T is determined dynamically in our implementation. The value is determined by doing a search over the values from 0.1 to 2 (with steps of 0.1) and choosing the temperature value that gets the best ECE value.

The use of a softmax function results in the following: If a sample is further away from a centroid or is in between two centroids, the uncertainty increases. The confidence of a prediction can be calculated by taking the value of the class with the highest probability, using the max operator:

$$\max_c K_c(f_\theta(x), e_c), \tag{3.2}$$

where c corresponds to the class with the maximum correlation (minimum distance) between data point x and class centroids E = {e1, . . . , e_C } (Van Amersfoort et al., 2020). The overall confidence of the model is calculated by taking the mean of all these prediction confidences.

3.2.2 MDRM

For the MDRM model, the structure follows that of the PyRiemann library (Barachant et al., 2023). Balanced sample weights were calculated for the MDRM model, which were used to ensure balanced results for the predictions. For the calculation of the covariance matrices, the Ledoit-Wolf estimator (Ledoit & Wolf, 2004) is used as the covariance estimator.

As mentioned in Section 2.4, the MDRM model outputs the distance to each class centroid for an input sequence, where the prediction is the class that is closest to the input. These outputted distances to all class means are normalized to calculate probabilities from them for each class. For this probability calculation, two different methods were used: a softmax function that uses the negative squared distances and a softmax function that uses temperature scaling (Guo et al., 2017).

The softmax of negative squared distances is chosen because this follows the implementation as made by the PyRiemann library (Barachant et al., 2023). The confidence of this method is calculated using Equation 3.3, where dist(x) are the distances to all centroids.

$$\text{softmax}\left(-\left(dist(x)\right)^2\right) \tag{3.3}$$

The softmax with temperature scaling, as depicted in Equation 3.1, is used as this more closely follows the way DUQ calculates the probability distribution over the classes for a prediction. The temperature value is again determined by doing a search over the values from 0.1 to 2.

With these probability-like distributions obtained from the softmax function, the confidence of a prediction is taken by using the max operator,

as earlier described in Equation 3.2. The overall confidence of the model is calculated by taking the mean of all the prediction confidences.

As far as we found in this study, the use of a softmax with temperature scaling is a novel approach for calculating the uncertainty of an MDRM model. Because this approach follows DUQ even more, it is interesting to see what consequences this has on the performance of the MDRM model; the results will be shown in Section 4.

4 Results

To evaluate the Shallow Convnet and MDRM models, several metrics were used to measure the classification performance and the UQ performance of the models. All metrics used will be discussed in Section 4.1.

The results for all models will be discussed in Section 4.2. The calibration plots alongside the performances on all used metrics will be depicted there.

4.1 Evaluation methods

For assessing the classification performance of the models on the BCI task, the F1 score and accuracy are used.

For the evaluation of UQ performance, metrics are needed that indicate how well-calibrated the ML models are. As described in Section 1, calibration is how well aligned a model's confidence is with the probability of the model being correct. In this study, the Expected Calibration Error (ECE; Naeini et al., 2015), Net Calibration Error (NCE; Groot & Valdenegro-Toro, 2024), and the Brier score (Graf et al., 1999) are used. The three metrics together provide a comprehensive analysis of the model's uncertainty quantification.

ECE measures the weighted average of the absolute difference between the accuracy and the confidence of a prediction. It indicates how much the confidence of the ML model corresponds to its performance. Lower ECE values indicate a better calibration, with a perfect score of 0 and a worst score of 1. The calculation is done for each bin, where a bin is a group of samples that fall into a range, measured over the predicted confidence. For this study, a bin size of 10% was used, which means that there is a bin with a 0-10% confidence level, a 10-20% confidence level, and so on. We chose this value for the bin size because it ensured enough data points in most of the bins, without losing important differences between bins by having too few. ECE is defined as:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\text{acc}(B_i) - \text{conf}(B_i)\right|, \quad (4.1)$$

where M is the number of bins used to group the samples, $|Bm|$ is the number of samples that fall in bin M, and N is the total number of samples. $Acc(B_i)$ and $conf(B_i)$ are respectively the accuracy and mean confidence of the predictions in a bin.

ECE, while insightful for the miscalibration, does not tell us whether the miscalibration is because of overconfidence or underconfidence (or a mixture of both). To know the direction of miscalibration another metric is needed: NCE.

NCE works similarly to ECE: it takes the difference between the accuracy and confidence of a prediction. However, instead of the absolute difference, the straightforward difference between the two is taken. This is done because the value now represents the direction of miscalibration of the model. This method gives us insight into whether the model is overconfident or underconfident, based on the direction. A negative NCE means that the model is overconfident, while a positive NCE means an underconfident ML model (Groot & Valdenegro-Toro, 2024). NCE is defined as:

$$NCE = \sum_{m=1}^{M} \frac{|B_m|}{N} \left(\text{acc}(B_m) - \text{conf}(B_m)\right). \quad (4.2)$$

NCE alone, however, is not sufficient for representing the calibration performance of a ML model. NCE is not sufficient because a value of 0 can occur even when the ML model is poorly calibrated, it just is equally over- and underconfident. Therefore ECE and NCE together can provide a better understanding of the calibration of a ML model.

Another metric that is used for uncertainty evaluation in this study is the Brier score. The Brier score measures how well the model's predicted confidence matches the real probabilities. A score of 0 is perfect, while a score of 1 indicates the worst calibration of the model's confidence. The formula

for the Brier score works similarly to the mean squared error, but now on the model's outputted confidences and real probabilities. The Brier score is defined as:

$$\text{Brier}(y, \hat{y} = N^{-1} \sum (y_i - \hat{y}_i)^2, \qquad (4.3)$$

where y_i is the true probability and $\hat{y}_i$ is the predicted probability.

If the ML model had a confidence value of 0.8, and it was correct, the Brier score would be $(0.8 - 1)^2 = 0.04$. However, if it had a confidence value of 0.8 and it was incorrect, the Brier score would be $(0.8 - 0)^2 = 0.64$. The Brier score helps to add extra understanding to the uncertainty of the model, because, contrary to ECE, also the accuracy of the model is incorporated. A perfect ECE value can happen because a model always predicts a probability of 0.5 for both classes and the model is correct 50% of the time. However, a perfect Brier score also needs perfect predictions of the model.

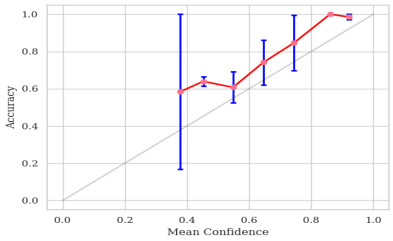
4.2 Model performances

Table 4.1 provides a comparison of all metrics, where the results are averaged over all datasets. It is important to note that there is a high fluctuation between participants per model. This variance between participants can be seen by the big error bars in Figure 4.1. The results for every individual dataset are given in Appendix A.

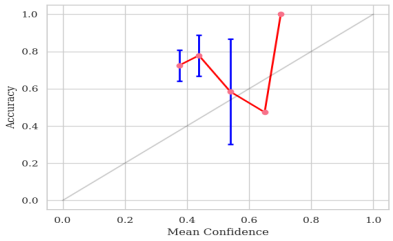
Additionally, for all ML models, a calibration plot is depicted, in which the confidence is plotted against the accuracy. This can be seen in Figure 4.1. In this figure, dataset 2 (the dataset from Zhou et al., 2016) is chosen because that dataset best demonstrates the differences between the models. The calibration plots on all the datasets can be found in Appendix A.

The diagonal line in the plots represents perfect calibration of the confidence. If the model's calibration is overall higher than the diagonal line, the model shows underconfident behavior. If the model's calibration is overall under the line for perfect calibration, the model shows overconfidence. In Figure 4.1, it can be seen that the MDRM models are underconfident, while the Shallow ConvNet models are overconfident.

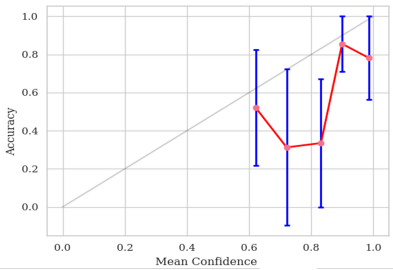
For the calibration plots, the number of bins used is 10. This means that the answers were grouped in



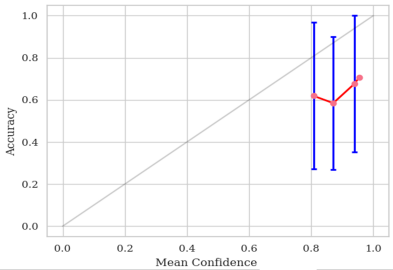
(a) Calibration plot for the MDRM model without temperature



(b) Calibration plot for the MDRM model with temperature



(c) Calibration plot for the DUQ model



(d) Calibration plot for the Deep Ensembles

Figure 4.1: Calibration plots for the different models, on dataset 2. The diagonal line represents perfect calibration, and the red line the model's calibration.

intervals of 10%. This value was chosen as it has a balance between enough data points in most of the bins without losing too much information on the difference between bins.

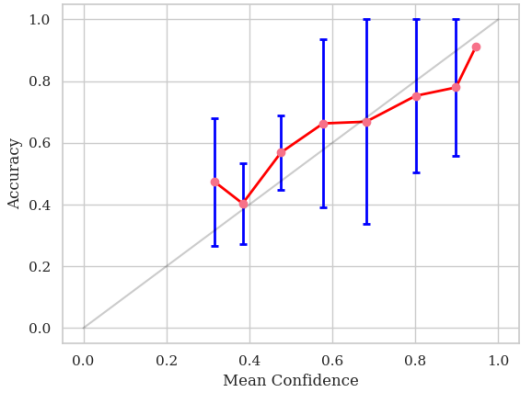
The error bars are calculated by taking the variance between subjects in the dataset. This approach is chosen because there is high variance between subjects, and thus the results may not be consistent for each subject.

4.2.1 MDRM

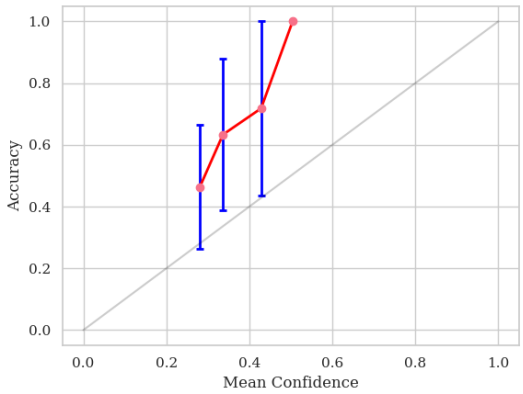
The accuracy and F1 score for both MDRM models are the same because the MDRM models only differ in the way they calculate their confidence. It can also be seen that the models perform slightly better than DUQ, but perform eminently worse than Deep Ensembles on classification performance.

Looking at UQ performance metrics in Table 4.1, it can be seen that the MDRM model that does not use the temperature function, performs better on all metrics than the model with temperature softmax. To investigate the difference between the two different methods of the MDRM model, the calibration plots for these models are depicted next to each other in Figure 4.2. In this figure, a comparison between the two models on their calibration can be seen on a different dataset than the one depicted in Figure 4.1. Another dataset was chosen here because this dataset better shows the difference between the two MDRM methods. The figure shows that the MDRM model without temperature more closely follows the line of perfect calibration and also has more different bins filled with data in the plot, indicating that the model has better aligned confidence with its accuracy. The model that uses temperature scaling is more underconfident in its decisions and does not show predictions that have a high confidence value, although the model does have predictions with high accuracy (as can be seen by the line hitting the maximum accuracy value). This indicates that the MDRM model that does not use temperature scaling is better at predicting its confidence. However, there is high variance between subjects in both models, as can be seen by the big error bars in both models, so the results may vary from what is depicted in the graph.

In both Figures 4.1, 4.2, and Table 4.1 we can observe that the MDRM models are overall underconfident in their decisions, with an NCE score of



(a) Calibration plot for the MDRM model without temperature



(b) Calibration plot for the MDRM model with temperature

Figure 4.2: Comparison between MDRM with and without temperature softmax, on dataset 4. The diagonal line represents perfect calibration, while the red line represents the model's calibration.

Metric	MDRM	MDRM temperature	DUQ	Deep Ensembles
Average accuracy	0.6806	0.6806	0.6556	0.7621
Average F1 score	0.6799	0.6799	0.6553	0.7618
Average Confidence	0.5917	0.5303	0.9003	0.9097
Average Brier score	0.1695	0.3332	0.2237	0.1469
Average ECE	0.1080	0.1756	0.2502	0.2199
Average NCE	0.0935	0.1505	-0.2252	-0.2122

Table 4.1: Comparison of the models over the different metrics

approximately 0.09 and 0.15 and a calibration line mostly above the line of perfect calibration. The models have an ECE of 0.11 and 0.18, which are prominently better compared to Deep Ensembles and DUQ. This means that the models have less miscalibration between their confidence and accuracy.

The Brier score of the MDRM without temperature is slightly worse than that of Deep Ensembles, but better than that of DUQ. MDRM using temperature scaling, however, has a much higher Brier score than that of DUQ and Deep Ensembles. This indicates worse performance.

Figure 4.1 shows that the MDRM methods differ a lot in calibration from DUQ, while they work similarly. This is an interesting observation because it means that the under/overconfidence is not the direct result of the uncertainty techniques used in this study. This observation will be discussed further in Section 5.

4.2.2 DUQ

For DUQ, Table 4.1 shows that the model performs worse than both of the MDRM models as well as Deep Ensembles on accuracy and F1 Score. Also on the UQ performance metrics the model did not perform well; with an NCE of approximately -0.23, the model shows overconfident behavior. This overconfident behavior can also be seen when looking at Figure 4.1, where the calibration line is lower than the diagonal perfect calibration line.

The model has an ECE of approximately 0.3, which is substantially higher than the other models. Therefore the model shows a bigger difference between the accuracy and confidence and thus has a bigger miscalibration. The Brier score of the model is lower than that of the MDRM model with temperature, but remarkably higher than that of Deep Ensembles and the MDRM model without temperature. From these performances, it can be concluded that DUQ is not the best model choice for a BCI task, compared to MDRM and Deep Ensembles.

4.2.3 Deep Ensembles

Table 4.1 shows that Deep Ensembles outperforms both DUQ and MDRM on accuracy and F1 score, with a remarkably higher performance than both.

On UQ performance, the Deep Ensembles method performs better than DUQ but worse than the MDRM models. Although having a better ECE than DUQ, the NCE is slightly worse than DUQ. This means that the model is more overconfident than DUQ in its decisions. The Brier score of Deep Ensembles is the best among all the ML models. It outperforms MDRM without temperature only slightly, as the Brier score is a bit lower. However, Deep Ensembles outperforms DUQ and MDRM with temperature substantially when looking at the Brier score.

5 Discussion

This study aimed to guide researchers in the choice of a model for a BCI task. The implications of the results, the limitations of the study, and the suggestions for future research will be discussed below.

5.1 Implications

A primary observation from the average results over all datasets in Section 4 is that, in terms of accuracy and F1 score, the Deep Ensembles method performs the best with scores around 0.76, meaning that the

model correctly predicts a sample around 76% of the time. The MDRM model (with and without temperature) and DUQ are close to each other in performance; the MDRM models slightly outperform DUQ with a difference of around 2% accuracy.

From this, it can be concluded that classification performance-wise, Deep Ensembles is the best choice of the models used in this study. It was expected that Deep Ensembles has better classification performance compared to DUQ, as this follows findings in literature (de Jong et al., 2023). However, Riemannian geometry models for BCI tasks are often well-performing, as they have claimed the first prize in five international predictive modeling competitions (Congedo et al., 2017), so their substantially worse performance compared to Deep Ensembles is slightly unexpected.

From Table 4.1 it is also noticeable that both DUQ and Deep Ensembles are overconfident, as they have a highly negative NCE value. This is in line with other literature, as Deep Neural Networks often experience overconfidence (Hwang et al., 2023; Guo et al., 2017).

Unlike DUQ and Deep Ensembles, the MDRM models are underconfident in their decisions, which is the opposite issue as the Shallow ConvNet models showed (which were overconfident). As far as we are aware in this study, no research has been done on the evaluation of uncertainty in MDRM models. There has been research on Riemannian models using uncertainty (Barthélemy et al., 2023), however, they did not evaluate the uncertainty performance. Therefore this study adds to the unstudied field of Uncertainty Quantification for Riemannian Geometry models, but the findings cannot be compared to the results in literature.

Seeing that DUQ and Deep Ensembles are both overconfident, while MDRM is underconfident, is interesting. This is interesting because both DUQ and Deep Ensembles have somewhat comparable UQ performance (they are both overconfident) although they work very differently, while the UQ performance of MDRM is very different although it works similarly to DUQ. This means that the reason MDRM is better at UQ performance is not because of the density-based method; otherwise, DUQ would likely also have good UQ performance (which is not the case). The difference between the UQ performance of DUQ and MDRM is therefore likely because of the nature of both models (Deep Learning vs non-Deep Learning), instead of their Uncertainty Quantification techniques. Since there is little to no research done on UQ in Riemannian geometry models, the findings cannot be placed in the context of other literature.

On the other hand, although DUQ and Deep Ensembles work very differently they are both very overconfident. This might therefore be the result of the working of CNNs rather than the specific UQ method used.

For the MDRM model with temperature scaling, the Brier score is worse by quite some distance compared to the other methods, even when compared to the MDRM model without temperature scaling. The high Brier score for this MDRM model indicates that the model is confident in wrong predictions, and/or not confident with correct predictions. This must be the case since there is no difference in classification performance compared to the MDRM model without temperature scaling (so this does not influence the Brier score).

Comparing all UQ results, it can be concluded that the MDRM model without temperature most accurately estimates its uncertainty. Therefore, this MDRM model is the best choice of the models used in this study regarding UQ performance.

Therefore the choice of ML model for a BCI task is dependent on what the main focus is. If the main goal is to get good performance then a Deep Ensembles method is best suited. However, if the focus is to accurately predict the uncertainty of the ML model, then an MDRM model that does not use the temperature softmax function is best suited. This focus on UQ can be important for a BCI task because, to get a model that will not make a prediction when it is uncertain, the uncertainty should be well calibrated. Without proper calibration of a model, the confidence does not say enough about the model's performance and thus in real-life scenarios it cannot be trusted to reject samples that are different from the data it was trained on. This is the case because in real-life scenarios the data can have a lot of variance, in the form of artifacts or mind-wandering for example. If the model is not good at estimating its uncertainty, then setting a boundary for the model not to predict if it is less confident than 90 percent, for example, would not work properly.

5.2 Limitations

While this study provides an analysis of the performances of four different UQ methods, several limitations should be considered when adapting this study's findings.

First of all, the study does not focus on data that is out of distribution. This means that the ML models tested in this study are not scored on how well they perform on UQ performance when data is suddenly very different from what it was trained on. In real-life scenarios, however, this can happen when a BCI receives artifacts or off-task EEG data. It is uncertain that the discovered results hold in that scenario.

Secondly, the calibration plots were created by grouping data into bins which represent 10% of the data. Given that both used Deep Learning models frequently show high confidence, some bins contain very little data while others contain almost all, leading to potential variability and unbalanced representations in the calibration plots. The same holds for the underconfident behavior in the MDRM models.

Thirdly, two of the chosen datasets only had two possible classes, which simplifies the classification task compared to the categorical nature of the other two datasets. This binary classification may not fully represent the models' capabilities in more complex, multi-class scenarios (although the performances weren't very different from the other datasets, as can be seen in Appendix A).

Lastly, in this study, there was no hyperparameter tuning performed for any of the models, which could have led to suboptimal performance. Hyperparameter choices were based on literature (as described in 3) instead of optimizing them on the validation data. Since this was done for all models, the comparison is still fair, but the true performances of each model can be potentially underestimated.

5.3 Future research

An interesting addition to this study would be to include out-of-distribution data in the study and see what happens with the UQ performance when the ML models get this data. It would be interesting to see the performances of the models in detecting this and see if the Riemannian geometry model still performs best.

An interesting thing would be to include a binary classification task for the ML models to predict when a sample is out of distribution and compare the performances of the ML models on that task. For example, a model can be trained on certain participants and then tested on other participants. This causes a shift in data which is out of distribution for the model. Alternatively, a model can be trained on data with two possible classes and we can test if the model can detect the third class based on its high uncertainty.

Additionally, an interesting addition for future research can be to include more model structures, such as a Tangent space model for the Riemannian geometry models. This can lead to more interesting results that can not be attributed to only the choice of a specific structure for the ML models.

Lastly, there has been little research on the combination between Riemannian Geometry and Neural Networks. A recent study by Gao et al. (2022) has shown promising results with regard to classification performance. Interesting for future research would be to see how well this combination could perform UQ performance-wise.

6 Conclusion

In this study it was tried to compare the classification performance and UQ performance of Deep Learning and non-Deep Learning models, to guide researchers in their choice of a model. The comparison was done between an MDRM model with distance-based uncertainty, and a CNN with either Deep Ensembles or DUQ as the UQ strategy.

Although DUQ and MDRM work similarly, the results indicate that the MDRM model, without the temperature softmax function, outperforms DUQ in both classification performance and UQ performance. This indicates that a DUQ model is less suited for a BCI task than the MDRM model.

Furthermore, the results show that the MDRM model without temperature has better uncertainty estimation than the Deep Ensembles method. The MDRM model shows better ECE and NCE values, but Deep Ensembles scored better on the Brier score. However, Deep Ensembles outperforms the MDRM models in terms of accuracy and F1 score, demonstrating substantially higher performance. The direction of miscalibration is very different be-

tween both methods. The MDRM models showed underconfident behavior, while the Deep Ensembles and DUQ methods showed overconfidence.

From these findings, it can be concluded that an MDRM model (without temperature) is best suited for a BCI task if the primary concern is the UQ performance of the model. However, if the main goal is to maximize the model's classification performance, Deep Ensembles is a better fit.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... others (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, *76*, 243–297.

Abdulkader, S. N., Atia, A., & Mostafa, M.-S. M. (2015). Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, *16*(2), 213–230.

Alharbi, H., et al. (2023). Identifying thematics in a brain-computer interface research. *Computational Intelligence and Neuroscience*, *2023*.

Ali, S., Li, J., Pei, Y., Aslam, M. S., Shaukat, Z., & Azeem, M. (2020). An effective and improved cnn-elm classifier for handwritten digits recognition and classification. *Symmetry*, *12*(10), 1742.

Aristimunha, B., Carrara, I., Guetschel, P., Sedlar, S., Rodrigues, P., Sosulski, J., ... Chevallier, S. (2023). *Mother of all BCI Benchmarks.* Retrieved from `NeuroTechX/moabb` doi: 10.5281/zenodo.10034223

Barachant, A., Barthélemy, Q., King, J.-R., Gramfort, A., Chevallier, S., Rodrigues, P. L. C., ... Corsi, M.-C. (2023). *pyriemann/pyriemann: v0.5.* Zenodo. Retrieved from `https://doi.org/10.5281/zenodo.8059038` doi: 10.5281/zenodo.8059038

Barachant, A., Bonnet, S., Congedo, M., & Jutten, C. (2010). Riemannian geometry applied to bci classification. In *International conference on latent variable analysis and signal separation* (pp. 629–636).

Barachant, A., Bonnet, S., Congedo, M., & Jutten, C. (2011). Multiclass brain–computer interface classification by riemannian geometry. *IEEE Transactions on Biomedical Engineering*, *59*(4), 920–928.

Barthélemy, Q., Chevallier, S., Bertrand-Lalo, R., & Clisson, P. (2023). End-to-end p300 bci using bayesian accumulation of riemannian probabilities. *Brain-Computer Interfaces*, *10*(1), 50–61.

BCI-Society. (2024). *Bci definition.* Retrieved 2024-01, from `https://bcisociety.org/bci-definition/`

Brigato, L., & Iocchi, L. (2020). On the effectiveness of deep ensembles for small data tasks. *arXiv preprint arXiv:2111.14493*.

Congedo, M., Barachant, A., & Bhatia, R. (2017). Riemannian geometry for eeg-based brain-computer interfaces; a primer and a review. *Brain-Computer Interfaces*, *4*(3), 155–174.

de Jong, I. P., Sburlea, A. I., & Valdenegro-Toro, M. (2023). Uncertainty quantification in machine learning for biosignal applications–a review. *arXiv preprint arXiv:2312.09454*.

Deng, L. (2012). The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, *29*(6), 141–142.

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).

Gao, C., Liu, W., & Yang, X. (2022). Convolutional neural network and riemannian geometry hybrid approach for motor imagery classification. *Neurocomputing*, *507*, 180–190.

Graf, E., Schmoor, C., Sauerbrei, W., & Schumacher, M. (1999). Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, *18*(17-18), 2529–2545.

Groot, T., & Valdenegro-Toro, M. (2024). Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. *arXiv preprint arXiv:2405.02917*.

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321–1330).

Hwang, Y., Jo, W., Hong, J., & Choi, Y. (2023). Overcoming overconfidence for active learning. *arXiv preprint arXiv:2308.10571*.

Köllőd, C. M., Adolf, A., Iván, K., Márton, G., & Ulbert, I. (2023). Deep comparisons of neural networks from the eegnet family. *Electronics*, *12*(12), 2743.

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.

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., & Jackel, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of multivariate analysis*, *88*(2), 365–411.

Leeb, R., Lee, F., Keinrath, C., Scherer, R., Bischof, H., & Pfurtscheller, G. (2007). Brain–computer communication: motivation, aim, and impact of exploring a virtual apartment. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *15*(4), 473–482.

Liu, X., Makeyev, O., & Besio, W. (2020). Improved spatial resolution of electroencephalogram using tripolar concentric ring electrode sensors. *Journal of Sensors*, *2020*(1), 6269394.

Makarova, A., Shen, H., Perrone, V., Klein, A., Faddoul, J. B., Krause, A., ... Archambeau, C. (2021). Overfitting in bayesian optimization: an empirical study and early-stopping solution. In *2nd workshop on neural architecture search (nas 2021 collocated with the 9th iclr 2021)*.

Manivannan, P., de Jong, I. P., Valdenegro-Toro, M., & Sburlea, A. I. (2024). Uncertainty quantification for cross-subject motor imagery classification. *arXiv preprint arXiv:2403.09228*.

Milanés-Hermosilla, D., Trujillo Codorniú, R., López-Baracaldo, R., Sagaró-Zamora, R., Delisle-Rodriguez, D., Villarejo-Mayor, J. J., & Núñez-Álvarez, J. R. (2021). Monte carlo dropout for uncertainty estimation and motor imagery classification. *Sensors*, *21*(21), 7241.

Mobiny, A., Yuan, P., Moulik, S. K., Garg, N., Wu, C. C., & Van Nguyen, H. (2021). Dropconnect is effective in modeling uncertainty of bayesian deep networks. *Scientific reports*, *11*(1), 5458.

Naeini, M. P., Cooper, G., & Hauskrecht, M. (2015). Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 29).

Niedermeyer, E., & da Silva, F. L. (2005). *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins.

Scherer, R., & Vidaurre, C. (2018). Motor imagery based brain–computer interfaces. In *Smart wheelchairs and brain-computer interfaces* (pp. 171–195). Elsevier.

Schirrmeister, R., Springenberg, J., Fiederer, L., Glasstetter, M., Eggensperger, K., Tangermann, M., ... Ball, T. (2017, 08). Deep learning with convolutional neural networks for eeg decoding and visualization: Convolutional neural networks in eeg analysis. *Human Brain Mapping*, *38*. doi: 10.1002/hbm.23730

Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., ... Ball, T. (2017). Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, *38*(11), 5391–5420.

Steyrl, D., Scherer, R., Faller, J., & Müller-Putz, G. R. (2016). Random forests in non-invasive sensorimotor rhythm brain-computer interfaces:

a practical and convenient non-linear classifier. *Biomedical Engineering/Biomedizinische Technik*, *61*(1), 77–86.

Tangermann, M., Müller, K.-R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., ... others (2012). Review of the bci competition iv. *Frontiers in neuroscience*, *6*, 55.

Valdenegro, M. (2023). *Keras-uncertainty.* `https://github.com/mvaldenegro/keras-uncertainty`.

Van Amersfoort, J., Smith, L., Teh, Y. W., & Gal, Y. (2020). Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning* (pp. 9690–9700).

Wen, Y., Vicol, P., Ba, J., Tran, D., & Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*.

Yger, F., Berar, M., & Lotte, F. (2016). Riemannian approaches in brain-computer interfaces: a review. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *25*(10), 1753–1762.

Zhou, B., Wu, X., Lv, Z., Zhang, L., & Guo, X. (2016). A fully automated trial selection method for optimization of motor imagery based brain-computer interface. *PloS one*, *11*(9), e0162657.

A Appendix

Dataset	Validation Accuracy	F1 Score	Overall Confidence	Brier Score	ECE	NCE
Dataset 1	0.7143	0.7137	0.9291	0.2329	0.2056	-0.2056
Dataset 2	0.8227	0.8236	0.9473	0.0892	0.3333	-0.3333
Dataset 3	0.7891	0.7890	0.9144	0.1537	0.1698	-0.1698
Dataset 4	0.7222	0.7207	0.8481	0.1117	0.1712	-0.1401

Table A.1: Deep Ensemble performance per dataset

Dataset	Validation Accuracy	F1 Score	Overall Confidence	Brier Score	ECE	NCE
Dataset 1	0.5134	0.5132	0.8582	0.4043	0.2440	-0.2162
Dataset 2	0.7507	0.7517	0.9564	0.1436	0.3270	-0.2548
Dataset 3	0.7730	0.7729	0.9327	0.1818	0.1894	-0.1894
Dataset 4	0.5852	0.5833	0.8542	0.1652	0.2402	-0.2402

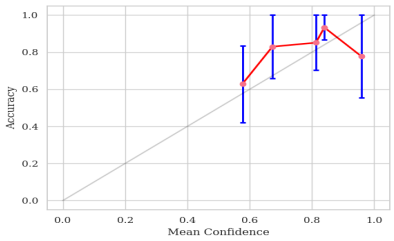
Table A.2: DUQ performance per dataset

Dataset	Validation Accuracy	F1 Score	Overall Confidence	Brier Score	ECE	NCE
Dataset 1	0.7031	0.7027	0.6390	0.3823	0.0984	0.0929
Dataset 2	0.7230	0.7228	0.4402	0.2824	0.2520	0.1814
Dataset 3	0.7140	0.7138	0.7120	0.4526	0.0294	0.0053
Dataset 4	0.5824	0.5803	0.3299	0.2159	0.3226	0.3226

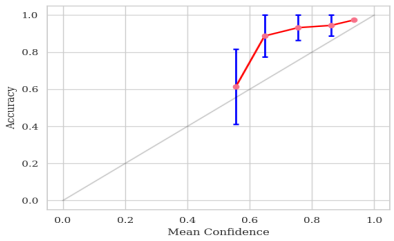
Table A.3: MDRM with temperature scaling performance per dataset

Dataset	Validation Accuracy	F1 Score	Overall Confidence	Brier Score	ECE	NCE
Dataset 1	0.7031	0.7027	0.6647	0.1833	0.0772	0.0615
Dataset 2	0.7230	0.7228	0.6104	0.1423	0.1232	0.1232
Dataset 3	0.7140	0.7138	0.5684	0.2115	0.1535	0.1535
Dataset 4	0.5824	0.5803	0.5234	0.1410	0.0783	0.0359

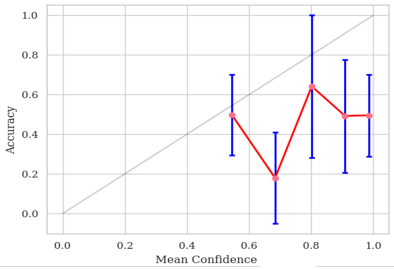
Table A.4: MDRM without temperature scaling performance per dataset



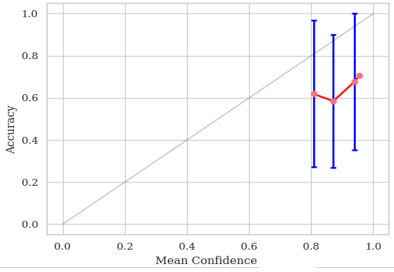
(a) Calibration plot for MDRM without temperature



(b) Calibration plot for MDRM with temperature

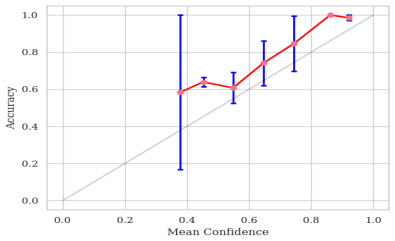


(c) Calibration plot for DUQ

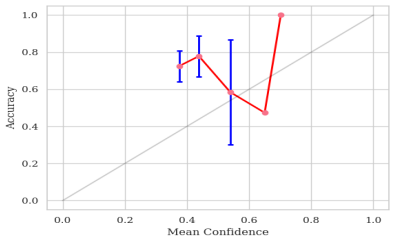


(d) Calibration plot for Deep Ensembles

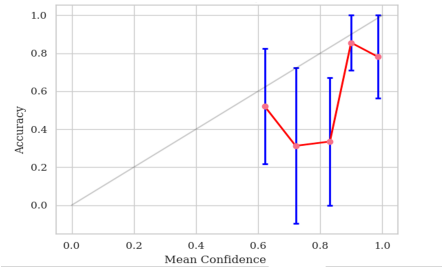
Figure A.1: Calibration plots for the different models on dataset 1



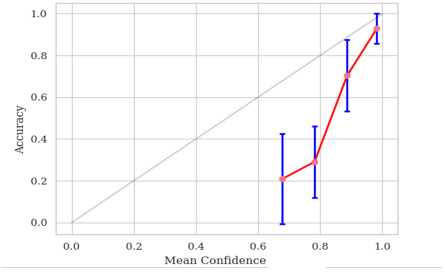
(a) Calibration plot for MDRM without temperature



(b) Calibration plot for MDRM with temperature

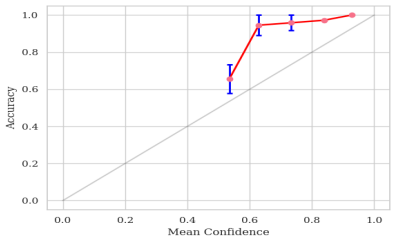


(c) Calibration plot for DUQ

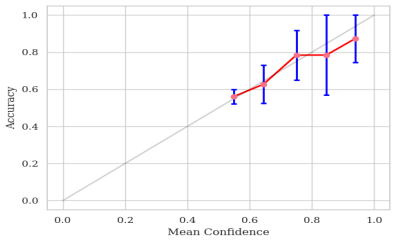


(d) Calibration plot for Deep Ensembles

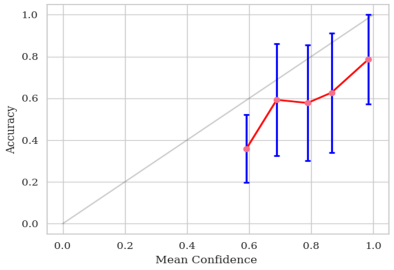
Figure A.2: Calibration plots for the different models, on dataset 2



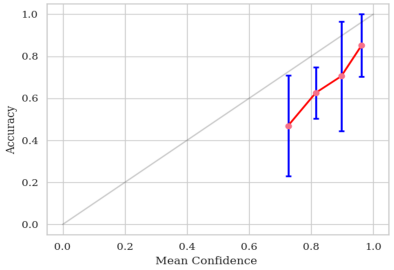
(a) Calibration plot for MDRM without temperature



(b) Calibration plot for MDRM with temperature

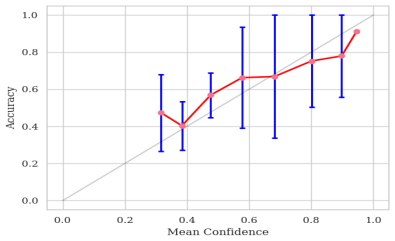


(c) Calibration plot for DUQ

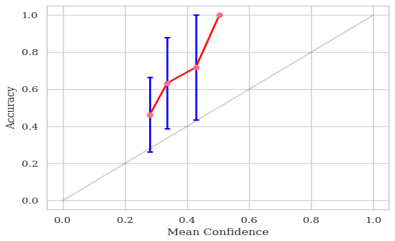


(d) Calibration plot for Deep Ensembles

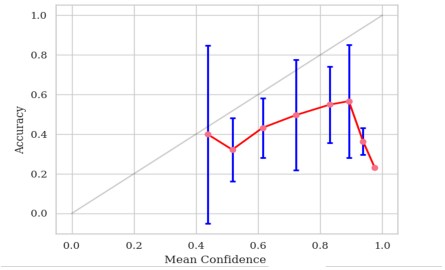
Figure A.3: Calibration plots for the different models on dataset 3



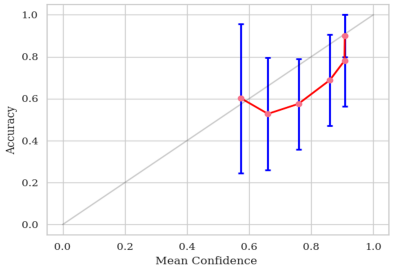
(a) Calibration plot for MDRM without temperature



(b) Calibration plot for MDRM with temperature



(c) Calibration plot for DUQ



(d) Calibration plot for Deep Ensembles

Figure A.4: Calibration plots for the different models on dataset 4