UTRECHT UNIVERSITY
Faculty of Science
Artificial Intelligence
Master's thesis

# Predicting developmental age in young children by applying deep learning approaches to EEG data

B.M.A. Bruns
6980341
August 5, 2021


*Daily supervisor*
Dr. F. Huber
*Supervisor and first examiner*
Dr. H.G. Schnack
*Second examiner*
Prof. Dr. F.N.K. Wijnen

**Acknowledgements**

I would like to express my gratitude to my daily supervisor Dr. Florian Huber for his excellent guidance over the past nine months. Even though we have not been able to see each other often due to the coronavirus pandemic, it felt like we were in close contact. He always took the time to answer my questions, brainstorm on a topic or help me in any other way and I appreciate that a lot. I also want to thank the Netherlands eScience Center for allowing me to be a research intern at the company and do my research there.

I'm also grateful for the guidance of Dr. Hugo Schnack, my university supervisor, and the great opportunity he offered me to work on the ePodium project and suggesting this research topic. His constructive feedback definitely increased the quality of this thesis. I would also like to thank Prof. Dr. Frank Wijnen for acting as an examiner for this thesis and Dr. Karin Wanrooij for her help, comments, and showing interest in my thesis.

Many thanks as well to my study friends, Nikos and Giannos, for making the past two years more enjoyable. Thanks to my friends and family as well for their love and supporting me in moving to Amsterdam years ago for my studies. Lastly, I would like to thank my girlfriend Alet for her unconditional support, love, and encouragement. Without her, I might never have started studying for this master's degree, which I have enjoyed tremendously.

**Abstract**

EEG-based age prediction models could be suitable indicators of brain maturational levels in children (from infancy till adolescence). In the past, traditional machine learning (ML) methods have been applied for predicting the age of young children using EEG data demonstrating the general feasibility of such an approach. Using an EEG data set (N=1,368,001 EEG epochs) from an extensive longitudinal study (children aged 11 to 47 months, N=304) we adapted state-of-the-art deep learning (DL) techniques to obtain improved age predictions for young children. This included implementing suitable tools to assess the uncertainty of the DL model, as well as tools to gain further insight into the models' decision-making process ("explainable AI"). First, using feature extraction techniques and traditional ML tools, we created a reference age prediction model. We then implemented and compared seven DL regression architectures, trained on raw EEG data. To test the hypothesis that EEG-based age estimates reflect brain maturational level, we investigated if advanced/delayed EEG-based ages correlate with expressive and receptive vocabulary size. Finally, we investigated if the models' age estimates are predictive of the presence or absence of recorded dyslexia predisposition. The best model, the cross-validated Encoder DL model, produced a mean absolute error of 4.82 months, a root mean squared error of 5.90 months, and an R-squared of 0.674. The brain age gaps (age estimate minus chronological age) were moderately to highly stable over time ($0.534 \leq r \leq 0.835$). Depending on the model, vocabulary sizes and brain age gaps have a weak to strong positive correlation at the age of 17, 23 (only expressive), and 35 months. A significant relationship between the brain age gap estimate and dyslexia predisposition was not found. We showed that DL models can outperform traditional ML models on infant age prediction using EEG data. DL models can extract features from raw EEG data, indicating the feature extraction and selection process can be avoided. For some DL models, the brain age gaps contain information about the subject's development (vocabulary), depending on the model and age group. We presented an outlook on DL model explainability possibilities. Based on our findings, we expect that it is possible to use DL models to find other biomarkers in EEG data as well.

# Contents

# 1 Introduction

One out of every six children between the age of three and seventeen years has some kind of developmental disability, as reported by their parents [1]. A non-optimal early development in a child can have negative consequences for its further life, for example on educational performances, productivity, and health [2][3]. Being able to identify this in infants is essential, as early intervention could help to optimize the developmental process and help a child to reach its full potential. For adults, it is possible to assess someone's mental development or cognitive capacity utilizing standardized tests like IQ tests, education level, or other proxies. For infants, however, it is harder to find a proxy for development. One possible measure for this is the developmental age prediction.

Predicting age with neuroimaging techniques is well established, and many studies have done this using MRI data [4]. Age prediction with EEG data and machine learning is relatively new, and therefore the number of studies is not very large. In the past years, it has been shown that traditional machine learning techniques can be used to predict (developmental) age in humans by the use of EEG data recordings [5][6]. Studies like these, both using MRI and EEG data, often display a considerable error between the chronological age of the subject and the predicted age. This brain age gap estimate is hypothesized to be a biomarker of brain maturation [7].

While multiple studies use traditional machine learning techniques such as support vector machines and random forests for age prediction with EEG data, there is little literature on the use of deep learning techniques on these kinds of predictions. Deep learning has shown impressive results in many fields in recent years, like image classification [8] and natural language processing [9]. One of the aims of this study is, therefore, to determine whether it is possible to improve the developmental age prediction performance on an EEG data set by using deep learning techniques. The data set we use comes from the Dutch Dyslexia Programme (DDP) [10], an extensive longitudinal study containing EEG data and other (developmental) information of infants and children in an age range of 11 to 47 months. If we can improve the performance by using deep learning techniques and predict development age accurately, this shows that these techniques can be used to determine properties of a subject. This might be an indication that EEGs can be used for finding other properties or biomarkers.

Another departure from former (deep learning) studies is that we use regression rather than classification for predicting age. Kaushik and colleagues make use of deep learning [11], but divide the subjects into different age groups, use these age groups as class labels, and train a model based on these labels. The model is then validated by classifying unseen data into one of these groups. However, the choice of the age group range has a large influence on the performance of these models. Also, when the groups have large ranges, it is relatively easy to achieve high

accuracy. When using regression, this opens up two possibilities: (1) analyzing the potential brain age gap, and (2) interpret the model better by being able to see how close the prediction is compared to the chronological age. Therefore, we aim to use regression to predict an exact age (in months) rather than an age group.

Deep learning models are often described as "black boxes" that are hard to interpret because of many parameters and many layers. There is a trade-off between the explainability of a model and the predictive performance. Nowadays, the field of explainable AI attempts to open these black boxes. If deep learning models were to be used for medical diagnosis, tools to assess the uncertainty of the model and the inner workings of the decision-making process are necessary. Therefore, the last aim of our study is to provide insight into the decision-making process of the models we propose.

An important part of brain age modeling and machine learning, in general, is validation. The data set that we use for this study comes with some additional features that we can use for the validation of the model. We use the vocabulary size of the subjects and see whether these correlate with the predicted developmental age. If so, this supports the hypothesis that EEG-based age prediction reflects the brain's maturational level. Furthermore, the models' age prediction will be analyzed in the context of the presence or absence of recorded dyslexia predisposition.

The rest of this thesis is structured as follows, in the next section we will provide background information on electroencephalography (EEG), machine learning, deep learning, and developmental age. We will also provide an overview of related literature on the combination of machine learning/deep learning and EEG data. We then explain the methods we have used for our experiments. Finally, we will report the results of our experiments, discuss them and draw conclusions from these results.

# 2 Background

In this section, we will provide background information on the topics EEG, machine learning, deep learning, and developmental age, which can be used for readers as an introduction on these topics to fully understand the contents of this thesis.

## 2.1 Electroencephalography

In this subsection, we will introduce electroencephalography (EEG), its (dis)advantages, and we will briefly compare it with other techniques.

EEG is a non-invasive technique that measures the electrical activity of the brain. These measurements are being used for brain mapping and neuroimaging, both in clinical and non-clinical applications. The measurement of this electrical activity is done by placing electrodes on the scalp of the subject. These electrodes can record voltage potentials on the skin of the subject that come from electrical flow in and around neurons of the brain [12].

More specifically, this electrical activity is generated by groups of neurons with similar orientations near the scalp. Every electrode on the scalp can record, minimally, an estimated 6 $cm^2$ synchronous cortical activity. The majority of the electrical activity measured is generated by groups of so-called pyramidal neurons. The electrical activity measured represents the sum of the inhibitory or excitatory postsynaptic potentials from thousands of pyramidal cells near each recording electrode. It should be noted that the potentials recorded are postsynaptic potentials and not action potentials, as those are too short (in time) to be recorded via EEG [13].

There are many different EEG devices and setups on the market, with a varying number of electrodes. The standard setup uses 22 electrodes of which one is a ground electrode [13]. Even though 22 electrodes is the standard, there are many setups with a much higher number of electrodes like 64, 128, or even 256 electrodes [12][14]. All these electrodes record a separate electrical signal over time, a so-called channel. The frequency of these recordings is dependent on the sampling rate of the device used. Sampling rates (in Hertz) could go as high as 10 kHz for standard commercial EEG devices [12], but for the majority of the research related to this one, a sampling rate of 250 Hz or less was used [14]. The electrodes are often placed on the scalp using the standard international 10-20 electrode site placement strategy, where 10-20 means that the electrodes are placed either 10 or 20 percent apart from each other starting from the bridge of the nose (nasion) to a prominent bump on the back of the head (inion) [13]. Using proportional placement systems like this one and others, have the main advantage of using the same relative placement on the scalp. This enables fair comparison of measurements between different subjects with varying head sizes, but also when comparing a subject to itself, for example when an infant

or child is still growing and EEGs are recorded at several moments in its development.

Since the invention of the EEG for humans by Hans Berger in 1924, several other non-invasive brain mapping and neuroimaging techniques have been developed like MRI and MEG. There are some advantages of EEG compared to those other techniques. The first advantage of EEG is that it has a relatively low cost. A standard commercial EEG device would cost around $60,000, whereas a 3 tesla MRI or a similar MEG scanner would cost around $2-3 million. Another advantage is the excellent temporal resolution of EEGs compared to other techniques [12]. This means that when a subject has been presented with a stimulus like a sound being played, or an image is shown, the EEG can record the immediate response of the subject to this stimulus. Also, due to this excellent temporal resolution, the brain's state can be monitored in real-time.

When comparing EEG to other techniques, there are also important disadvantages. The first one is that EEG has a poor spatial resolution. As mentioned before, the electrical activity measured at each of the electrodes is the sum of the inhibitory or excitatory postsynaptic potentials from thousands of pyramidal cells. Strong electrical activity can be picked up by multiple electrodes at once. This makes it hard to determine exactly where the activity originates from. Another disadvantage of EEG is that it has a low signal-to-noise ratio, caused by different "artifacts", the electrical activity measured is affected by several sources of environmental (caused by other electrical devices present in the room), physiological (eye blinking, heart beating) or activity-specific noise of varying amplitude [14][15].

In most EEG research, there is the convention that the EEG signal is split into different waveform bandwidths (bands). Splitting the EEG data into these bands enables researchers to investigate several states a subject might be in, which are predominantly visible within specific frequencies. The exact ranges of the bands differ slightly between sources [12][16]. The bands are called the delta (∼0.2-3.5 Hz), theta (∼4–7.5 Hz), alpha (∼8–13 Hz), beta (∼14–30 Hz), and gamma (∼30–90 Hz) bands [12]. Frequencies above 90 Hz are very high frequencies and are not a named band. When the EEG signal is split into different bands, it can be explored further, for example by extracting features from it like the spectral power in 1 Hz bins, mean power, relative power, and many more.

Another common technique that is used in EEG research is ERP, the event-related potential. ERPs are EEG measurements during which the subject has been exposed to certain stimuli, for example, sounds or images. The recorded EEGs are then often averaged to a single signal, canceling out noise or signals that are not related to the stimulus.

Currently, EEGs are used in clinical and non-clinical applications. Non-clinical applications can, for example, be found in neuroscience and psychological research. There are also many clinical applications of EEG. Examples are sleep pattern studies or epilepsy diagnosis. Another

application can be found in brain-computer interfaces (BCIs), where EEG signals can be used to directly control devices that impact the subject's environment.

## 2.2 Machine learning

In this subsection, we will introduce the field of machine learning. For convenience, we have split this subsection into three parts: machine learning basic concepts, traditional machine learning, and artificial neural networks (ANN). The term "traditional" does not necessarily refer to a certain method being older or that it is outdated by now, it is only traditional in a sense that it is not an ANN. This split is only made because the next subsection on deep learning will build upon the foundations laid in the ANN part.

### 2.2.1 Machine learning basic concepts

Machine learning is a sub-field of artificial intelligence and refers to a collection of multiple learning algorithms. Machine learning algorithms are algorithms that can learn from data. A brief definition of (machine) learning was given by Tom M. Mitchell in 1997: "A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$" [17].

The task $T$ can be a variety of things, but often refers to how the algorithm handles an input example, which is a collection of multiple features. An example of a feature is the voltage potential measured by a single electrode at a single point in time. The collection of features then consists out of the potentials measured by all electrodes over a period of time (e.g. 1000 milliseconds). Several tasks can then be solved with this input example, like classification (male vs. female), regression (age in years), transcription (sound to text, image to text), machine translation (Dutch to English), structured output (natural language to grammar tree), anomaly detection (missing EEG channel), synthesis and sampling (music generation), imputation of missing values, denoising, or density estimation [18].

The performance measure $P$ describes how well machine learning algorithms perform. In the case of classification, the most straightforward performance measure is accuracy. The accuracy of a model is determined by taking the proportion of examples that are correctly classified. During the learning process (training), the loss (error) of the model is often used, which is calculated via a loss function. In general, the loss captures the proportion of incorrectly classified examples and therefore the objective is to minimize the loss. The minimization of the loss can be done via several techniques like gradient descent, depending on the machine learning approach used. The lower the loss, the higher the model's performance. When the model is finalized, the performance

(accuracy) is then evaluated on a test data set. The test set is a part of the total data set which is excluded from training. The idea behind excluding the test set from training is that the performance on the test set is a proxy of how the trained model would perform on real-world unseen data. Including the test data in the model would create a bias towards this data and it would not be an unseen proxy anymore. For other tasks, like regression, accuracy can not be used. Other performance measures like the mean squared error or mean absolute error are used in this case.

Lastly, the experience $E$ is what the algorithm uses to improve its performance on the task. In most cases, this experience is the data set that is used for training. The data set consists of many input examples, or data points, with often many different features. These examples can then be either labeled or unlabeled, meaning that they have a certain class, target, or value associated with it. An example labeled data set would be EEG data of multiple subjects split into 1-second examples, with a gender label associated with them.

Machine learning algorithms can usually be divided into three categories, supervised, unsupervised, and reinforcement learning. Supervised learning algorithms use labeled data to train, whereas unsupervised learning algorithms use unlabeled data to train. Reinforcement learning is are algorithms that learn based on the principle of maximizing rewards via a score function. In this thesis, we will only focus on supervised learning, as this is the only type of learning we are using. Supervised learning is used for classification (discrete values) and regression (continuous values). Examples of algorithms that can be used for supervised learning are tree-based models (decision tree, random forest), Naive Bayes classifiers, K-nearest neighbors, linear regression, logistic regression, support vector machines, and neural networks [19].

The objective of machine learning algorithms is to create a model that performs well on data that it has not seen before. Performing well on unseen data is called generalization. The algorithm is trying to minimize the loss and by doing so, it is improving its performance. The loss it is minimizing is the training loss, i.e. the error on the training set. The loss on the unseen data is called the test loss or generalization error. The algorithm may adjust the parameters of the model in a way that fits the training data really well and the training loss is low. However, at the same time, it might be the case that the model fits the training data so well that it has adjusted to all the details and peculiarities of the training data, and the performance on the test set is substantially lower. This phenomenon is a challenge in machine learning and is known as overfitting. Another challenge in machine learning is underfitting, which means that the algorithm is not able to decrease the training error to a value that is low enough (for the model to be usable). There are several techniques to deal with underfitting and overfitting, for example by decreasing or increasing the model's complexity by adding parameters to or removing

parameters from the model, respectively. Decreasing the model's complexity is often done via a technique called regularization.

Machine learning algorithms try the determine the optimal parameters of a model to achieve a low error. These parameters are determined in the training process. However, there are also several hyperparameters that can be tuned to improve performance. These hyperparameters are not set by the learning algorithm but have to be determined by the designer of the model. Examples of hyperparameters are for example the number of terms a polynomial has, the maximum depth of a decision tree, the number of trees in a random forest model, or the number of hidden layers in a deep neural network. If the hyperparameters are tuned by looking at the training error, the model will overfit. If the hyperparameters are tuned by looking at the test set, a bias towards the test set is introduced and the final evaluation is not correct anymore. There are several ways to tackle this problem, but the two main solutions are creating a validation set and cross-validation. A validation set is a part of the data set (like the test set) that is excluded from training and is then used to determine the optimal hyperparameters. This is done by making predictions on the validation set with different hyperparameter settings and compare the performances. This way there is no bias introduced towards the test set. Cross-validation is a technique where the data set is split into several parts or folds. It is usually used when the data set is relatively small and splitting the data into multiple sets decreases the number of examples in a way that impacts the learning ability of the algorithm too much. The most common cross-validation technique is called k-fold cross-validation. Every time the algorithm is training, one of the folds is left out as a test or validation set, which can then be used for tuning the hyperparameters. Finally, when the hyperparameters are determined, the whole data set can be used to train the final model.

### 2.2.2 Traditional machine learning methods

#### Support vector machines

A prominent traditional supervised learning technique known to frequently compete well even against (deep) artificial neural networks is the support vector machine [20][21]. Support vector machines (SVMs) are algorithms that make a class decision (positive class versus negative class) instead of providing probabilities as an output.

The principle behind the support vector machine is that it separates the data with a decision boundary, a line or hyperplane in higher dimensions, and doing so whilst maximizing the margin between the separating plane and the nearest data points from both classes. The SVM makes use of a (sparse) non-linear kernel function to transform the data into a different (higher) dimensional space, which makes the data separable. This is also known as the "kernel trick". Multiple other algorithms make use of kernel functions but those often look at all possible combinations of

training points. This quickly becomes computationally infeasible during training when data sets become larger and also leads to excessive computation time when making predictions on new input. Sparse kernel functions only look at a subset of the training data set, tackling these two drawbacks [19]. The hyperplane of the SVM is represented as follows:

$$\mathbf{w}^T \phi(\mathbf{x}) + b = 0 \tag{1}$$

Here, $\mathbf{w}$ represents the weight vector, $\phi(\mathbf{x})$ denotes a fixed feature-space transformation, and $b$ is the bias term. In the case of perfectly separable data, where there is at least a single hyperplane that perfectly separates the data, the algorithm searches for the hyperplane with the maximum margin to the closest data points (maximizing the minimum distance to the data points). When filling in a positive data point in the hyperplane equation, the result will be positive. Filling in a negative one, the result will be negative. If the true label (-1 negative and +1 positive) for example $n$ is represented as $t_n$, the following constraint can be constructed for when the classifier classifies correctly for all $n$:

$$t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1 \tag{2}$$

As said, the margin must be maximized, which can be represented as $\frac{1}{||\mathbf{w}||}$. For mathematical convenience, often the inverse of this margin is minimized and it is multiplied by a factor of 0.5. Thus, for finding the maximal margin separating hyperplane, we must minimize the following, constrained by (2):

$$\frac{1}{2}||\mathbf{w}||^2 \tag{3}$$

To solve this constrained optimization problem (quadratic programming problem), Lagrange multipliers are used. The exact formulation and how to solve this is beyond the scope of this thesis as it is not the main focus area, but a more extensive explanation can be found in Bishop (2006) [19]. New data points can be classified by looking at the sign of $y(\mathbf{x})$ in the final equation:

$$y(x) = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b \tag{4}$$

Here, the sparse kernel function is $k(x, x') = \phi(x) \cdot \phi(x')$. This kernel function performs a fixed feature-space transformation enabling the algorithm to learn non-linear relationships using convex optimization techniques that converge efficiently [18]. The parameters $a_n$ are the Lagrange multiplier terms, with the condition $a_n \geq 0$. During solving the constrained optimization problem described above, for many (most) of the data points, this multiplier is set to zero, and therefore those data points are ignored whilst making predictions for new data points. The data points

that do not have a multiplier of 0 are called the support vectors and lie on the maximum margin of the hyperplane.

Often, the data is not perfectly separable by the hyperplane. And even when this is possible, it might lead to overfitting because it is susceptible to noise in the data. Therefore, SVMs often make use of a so-called soft margin. A soft margin allows data points to be inside the margin or even to be misclassified. To allow this, slack variables are used. Each data point has a slack variable associated with it, defined as $\xi_n \geq 0$. Data points that are exactly on the margin or the correct side of the margin have $\xi_n = 0$, for all other data points it is defined as $\xi_n = |t_n - y(\mathbf{x}_n)|$, meaning that data points on the decision boundary have a value of 1, data points on the correct side of the boundary but inside the margin have a value of $< 1$, and values that are on the incorrect side of the decision boundary have a value of $> 1$. This last group of data points will be misclassified by the model. The constraint (2) where all data points are being classified correctly will be replaced the following, for all $n$:

$$t_n y(\mathbf{x}_n) \geq 1 - \xi_n \tag{5}$$

Now data points are allowed to be inside the margin or even on the wrong side of the decision boundary, we want to control how much they are allowed to do so. We still want to maximize the margin, whilst penalizing data points that are within the margin or misclassified. Similar to (3) we minimize the following, constrained by (5):

$$C \sum_{n=1}^{N} \xi_n + \frac{1}{2}||\mathbf{w}||^2 \tag{6}$$

In the above equation, the parameter $C > 0$ determines how many data points that are on the incorrect side of the margin are penalized. If $C$ goes towards infinity, the hard margin case will be reached again. The parameter $C$ is a trade-off between maximizing the margin and penalizing errors. Classification of new data points can still be done with the same equation in (4).

**Support vector machine regression**

SVMs are initially designed for classification, but can also be used for regression problems [22]. To do so, the algorithm has to be modified, but the principles remain the same. Support vector machine regression also makes use of support vectors, and therefore a lot of the training data can be ignored. The main difference between classification and regression is that we are dealing with continuous output instead of a class. Therefore, an $\epsilon$-insensitive error function is used. This function returns a error of zero when the (absolute) difference between the target value $t$ and

the prediction $y(\mathbf{x})$ is less then the hyperparameter $\epsilon$. It is up to the designer of the model to define the exact error function, but an easy example is a linear one [19]:

$$E_\epsilon(y(\mathbf{x}) - t) = \begin{cases} 0, & \text{if } |y(\mathbf{x}) - t| < \epsilon \\ |y(\mathbf{x}) - t| - \epsilon, & \text{otherwise} \end{cases} \tag{7}$$

Then, we can minimize the following to get the maximum margin:

$$C \sum_{n=1}^{N} E_\epsilon(y(\mathbf{x}_n) - t_n) + \frac{1}{2}||\mathbf{w}||^2 \tag{8}$$

Similarly to SVM classification, we can introduce slack variables to get a soft margin and allow some error. However, in contrast to the classification case where being on the wrong side of the decision boundary led to misclassification, we now need two slack variables $\xi_n \geq 0$ and $\hat{\xi}_n \geq 0$, because it does not matter whether the point is below or above the curve, the error is similar. Therefore, we find $\xi_n > 0$ for a point where the true value is higher than the predicted value plus the error term $\epsilon$. On the other side of the curve, we find $\hat{\xi}_n > 0$ for a point where the true value is lower than the predicted value minus the error term $\epsilon$. This leads to the following conditions:

$$t_n \leq y(\mathbf{x}_n) + \epsilon + \xi_n \tag{9}$$

$$t_n \geq y(\mathbf{x}_n) - \epsilon - \hat{\xi}_n \tag{10}$$

And the following equation should then by minimized, constrained by $\xi_n \geq 0$, $\hat{\xi}_n \geq 0$, (9), and (10):

$$C \sum_{n=1}^{N} (\xi_n + \hat{\xi}_n) + \frac{1}{2}||\mathbf{w}||^2 \tag{11}$$

The actual minimization is beyond the scope of this thesis, but it again makes use of Lagrangian multipliers, and predictions for new inputs can then be made with:

$$y(\mathbf{x}) = \sum_{n=1}^{N} (a_n - \hat{a}_n) k(\mathbf{x}, \mathbf{x}_n) + b \tag{12}$$

**Relevance vector machine regression**

Support vector machine regression has a few limitations, one of them is that the parameter $C$ has to be found via a cross-validation method. Another downside is that the SVM generates decisions rather than probabilities. The relevance vector machine (RVM) regression method is an

alternative, Bayesian sparse kernel technique that is similar to support vector machine regression that tackles a few of the limitations. The RVM was developed by Tipping [23] in 2001 and has the benefit of often generating sparser models leading to faster predictions on the test set, with little or no reduction in generalization error compared to the SVM.

The exact definition of the relevance vector machine regression method is beyond the scope of this thesis, as it is quite involved and not the main focus of this research. Bishop [19] offers a clear overview and explanation of the RVM method. The general RVM regression prediction takes the form:

$$y(\mathbf{x}) = \sum_{n=1}^{N} w_n k(\mathbf{x}, \mathbf{x}_n) + b \tag{13}$$

This equation is similar to (12), except for the coefficients $a_n$ and $\hat{a}_n$, which are replaced by $w_n$. This weight variable $w_n$ is considered to be zero-mean Gaussian and dependent on two parameters, $\alpha$, and $\beta$, which have to be estimated, often via an iterative EM algorithm.

As a result of the estimation and optimization of the RVM, a significant proportion of these $\alpha$ parameters will become large (to infinity), leading to weights with zero mean and zero variance, which therefore can be removed from the model. The remaining non-zero weights are the relevance vectors, and these are comparable to the support vectors of a regular SVM. The advantage of a relevance vector machine, however, is that there are generally fewer relevance vectors than support vectors, leading to faster predictions.

### Random forest

Totally different machine learning techniques being used very often, are tree-based models and random forests specifically. The random forest algorithm is an extension of the decision tree. The foundations for classification and regression trees were laid by Breiman in 1984 [24]. A decision tree is a simple, interpretable, and explainable model. They have the advantage that they can be displayed graphically, especially when they are small, which makes interpretation possible by people non-expert users. For our purposes, we will only focus on trees for regression. To fully understand the random forest algorithm, we explain how a decision tree works and then extend it to the full random forest algorithm.

A decision tree works via the generation of (binary) splitting rules on the training data set features. Each split on a feature sends a part of the data to the left branch and the other part to the right branch. An example rule could be $age \geq 18$, if the *age* feature of the data point is greater than or equal to 18, it will go to the right branch, otherwise to the left one. Then, these branches can either be split again with another splitting rule (internal node) or it could

end in a so-called terminal node or leaf. In the case of regression, this terminal node would then represent a continuous value. In a decision tree, the most important features are often at the top of the tree.

The training data will be split into separate regions, and the mean target value of the data in this region will be the prediction in the terminal node of this region. The regions are determined via a top-down, greedy approach, meaning that it starts at the top of the tree and chooses the best split at that point inside the tree. The best split is determined by looking at all the features of the training data, it also looks at all the different possible split values for these features. The algorithm then chooses a split on one of the features that achieves the greatest reduction in the residual sum of squares (i.e. the resulting tree with the lowest RSS). Thus, we are trying to minimize the following equation [25]:

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{14}$$

where $j$ and $s$ are the feature to split on and the split value, respectively. $R_1(j,s)$ and $R_2(j,s)$ are the regions which the data is split into, $x_i$ is a training data observation and $y_i$ its corresponding value. Lastly, $\hat{y}_{R_1}$ and $\hat{y}_{R_2}$ are the mean values for all the training observations combined in the regions $R_1(j,s)$ and $R_2(j,s)$, respectively. This process can then be continued for these two regions, splitting both of them again into two new regions. This process can continue until a stopping criterion is reached. Examples of stopping criteria are a minimum of observations per terminal node, a maximum depth of the tree, or a minimum of observations in an internal node to be allowed to split.

When the decision tree is created (grown), the value for a new observation can be predicted by traversing the tree top-down, ending up in one of the terminal nodes. However, this top-down, greedy approach has a downside in that it is likely to overfit and it only considers the best split at that current moment in the tree-generation process. Therefore, there are several techniques, like cost-complexity pruning combined with cross-validation to generate better performing trees. For an exact formulation of this process, we refer to (for example) James and colleagues (2013)[25] and Russell and Norvig (2010)[26].

Simple decision trees tend to have relatively poor performance when compared to other machine learning approaches. However, the ideas of bagging and random forest can improve the performance significantly. Bagging is short for bootstrap aggregating, and it can be used to reduce the problem of high variance that decision trees often have. When bagging, we generate $M$ different training data sets of the same (original) size, by taking random samples with replacement from the original data set and then generate $M$ trees. The final prediction can then be determined by averaging the predictions of these $M$ number of trees.

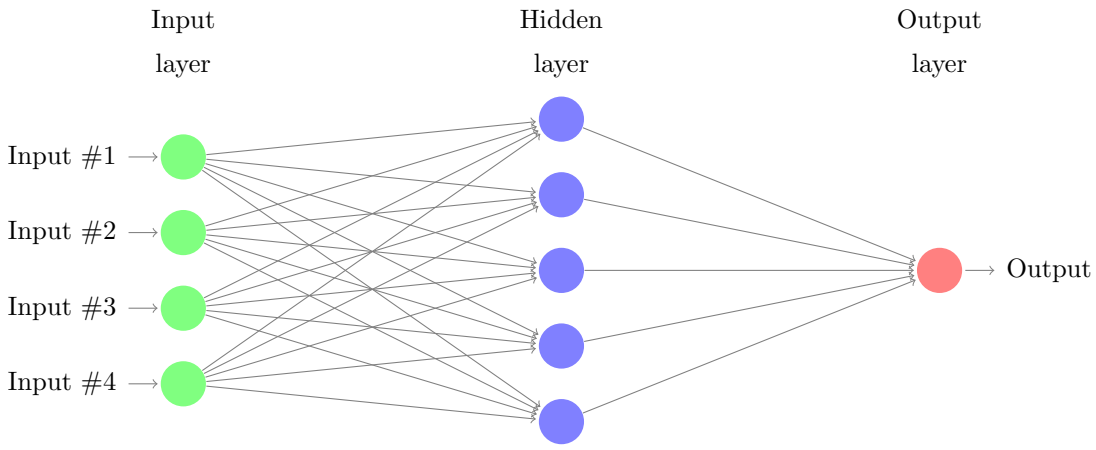



Figure 1: Simple graphical representation of an artificial neural network. Taken from [27].

The last extension to the tree-based methods is random forests. This is similar to bagging, but it also decorrelates the generated trees by only considering a subset of the features for each split. Every time a tree attempts to generate a split in a node, it is only allowed to consider a subset of $n$ features that are randomly chosen from all the features. The value of $n$ is often the square root of the number of features, but can also be determined via a cross-validation approach. When a split is made, the algorithm will choose another random subset of the features that can be considered for the next split. The idea behind this is that when there is a very strong predictor, it will be in the top node of almost all the bagged trees. As a consequence of this, the trees are all highly correlated and look very similar, which often still causes high variance. Random forests are less likely to overfit, and therefore the number of trees $M$ can be increased until the error rate is sufficiently low.

### 2.2.3 Artificial neural networks

In the previous part, we have discussed traditional machine learning methods. In this part, we will discuss the artificial neural network, which is the foundation of deep learning models.

The most simple form of an artificial neural network (ANN) is the feedforward or fully-connected neural network. The feedforward neural network consists out of (multiple) nodes or units in a layer that are connected via links between the layers. Figure 1 shows a graphical representation of a so-called fully-connected feedforward neural network. Fully-connected means that all the nodes in a layer are connected to all the nodes in the next layer. The network consists out of three parts, the input layer, the hidden layer, and the output layer. The connections between a node $i$ in the previous layer and node $j$ in the current layer are propagating an

| Name | Function |
|---|---|
| Sigmoid | $g(x) = \dfrac{1}{1+e^{-x}}$ |
| Tanh | $g(x) = \dfrac{e^{x}-e^{-x}}{e^{z}+e^{-z}}$ |
| ReLU | $g(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{otherwise} \end{cases}$ |
| Leaky ReLU | $g(x) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha \cdot x & \text{otherwise}(\alpha \geq 0) \end{cases}$ |

Table 1: Non-linear activation functions

activation $a_i$ from $i$ to $j$. All the connections have a weight $w_{ij}$ that determines the strength of the connection between the two nodes. Every node in the hidden and output layers first calculate a (linear) weight sum of its inputs:

$$in_j = \sum_{i=0}^{n} w_{ij} a_i + w_{0j} \tag{15}$$

In this equation, $w_{ij}$ are the weights and $w_{0j}$ is called the bias. The bias could also be expressed as $b_j$. Finally, a non-linear activation function $g(x)$ is applied to this weighted sum to calculate the output of the current layer:

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{ij} a_i + b_j\right) \tag{16}$$

There are various commonly used activation functions, which can be found in table 1. They are used to introduce non-linearity in the model, enabling learning more complex relationships. Four activation functions are most commonly used, the sigmoid, tanh, ReLU, LeakyReLU.

The above calculations are only made in the hidden and output layers, not in the input layer. The input feature vector is used as the input "activation" $\alpha_i$ (or $x_i$) for the next layer, the so-called hidden layer. The activation of the hidden layer is then the input for the output layer, which often is a single node for feedforward neural networks. Different tasks come with different activation functions in the output layer. For binary classification, the sigmoid activation function is common, whereas for regression just the identity of the weighted input activation is used.

Making predictions with a feedforward neural network is simple: feed an observation to the

input layer, this is propagated forward to the hidden layer where some mathematical transformations as in (16) are made. Lastly, it is propagated forward again to the output layer where a similar transformation happens, and a prediction is returned. This prediction can be several things, for example, a binary classification (using a sigmoid activation function) or just a number for regression (using the identity of the weighted sum of the hidden layer). The prediction $y_k(\mathbf{w}, \mathbf{x})$ for these three stages combined in a simple network like this one can actually be represented in a relatively compact equation:

$$y_k(\mathbf{w}, \mathbf{x}) = g\left(\sum_{j=1}^{M} w_{jk}\, g\left(\sum_{i=1}^{D} w_{ij}x_i + w_{j0}\right) + w_{k0}\right) \tag{17}$$

where $\mathbf{x}$ is the input vector and $\mathbf{w}$ are the weights and biases in the network.

What remains is how these weights (and biases) in $\mathbf{w}$ are determined. To find these parameters, the network is trained using a training data set, containing labeled data (class label or true value). There are several procedures to set the initial weights, but the easiest method is to set them randomly within a certain range (other methods are not discussed here, see chapter 8.4 of Goodfellow [18] and [28]). Then, training examples are passed through the network and the predicted value is compared to the true value via the use of a loss function $L(\mathbf{w})$ (or error function). The choice of the loss function depends on the task at hand, a few examples of loss functions are given in table 2. The task is then to minimize the chosen loss function (which is a function of $\mathbf{w}$), by finding the correct weights $\mathbf{w}$. When the loss on the training examples is calculated, the weights can be updated via a procedure called gradient descent and backpropagation.

The goal is to find a set of weights $\mathbf{w}$ where the loss $L(\mathbf{w})$ is smallest. Normally, this can be done by taking the gradient of the loss function and set it equal to zero.

However, the loss function has a highly non-linear dependence on the weights with many possible weight configurations, potentially leading to (nearly) vanishing gradients [19]. Those configurations are essentially local minima and it is usually not possible to determine if a global minimum has been reached.

Even though it is hard to know whether the optimal solution (global minimum) has been reached, reaching the optimal solution is not necessary to get a sufficient configuration of weights. To find a solution, an iterative approach is used, called gradient descent.

Gradient descent updates the weights in small steps in the opposite direction of the gradient of the loss function, such that the weights in time $t$ are:

$$\mathbf{w}^t = \mathbf{w}^{t-1} - \eta \nabla L(\mathbf{w}^{t-1}) \tag{18}$$

where the parameter $\eta > 0$ is called the learning rate. The learning rate is a hyperparameter

| Name | Function |
|---|---|
| Mean absolute error (L1) | $MAE = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - \hat{y}_i\rvert$ |
| Mean squared error (L2) | $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$ |
| Root mean squared error | $RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$ |
| Mean squared logarithmic error | $MSLE = \frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2$ |
| Root mean squared logarithmic error | $RMSLE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$ |

Table 2: Loss function examples for regression problems, $y_i$ is the true value, $\hat{y}_i$ the predicted value, and $N$ the total number of training samples

and is set by the designer of the network.

The weights are now updated and the process can start again by propagating the data through the network. A variation on gradient descent that is used in practice is called stochastic gradient descent (SGD). The difference is that the weights are not updated after the whole training data set is passed through the network, but when a single data point or a subset of the data (mini-batches) is passed through the network. When all the training data points have been through the network once, it is called an epoch. Advantages of using SGD are that it can handle redundancy in data more efficiently and it has the possibility to escape local minima [19]. In practice, there are several optimizers implementing variations of gradient descent and other weight-updating algorithms that can be used, like Momentum, Adagrad [29], Adadelta [30], Adam [31], RMSprop [32], and NAdam [33].

Backpropagation is a technique that enables the computationally efficient calculation of the loss function derivative with respect to the weights (and biases) in the different parts of the neural network. The loss derivatives are propagated back through the network to earlier layers. These can then be used for the process of gradient descent (or other processes to update the weights) described above. Backpropagation is mathematically slightly involved, and out of scope for this thesis, but fine explanations can be found in Bishop (2006) or Russell and Norvig (2010) [19][26]. The main idea is that, when calculating the derivatives with respect to the weights from

the output layer of the network towards the input layer, the calculated gradients in a layer can be reused in the calculation of the gradients in the previous layer because most of the calculation consists out of the already calculated part for the next layer.

A feedforward neural network has many properties (hyperparameters) that can be tuned to optimize its performance. The number of units in the hidden layer can be increased or decreased, the learning rate can be changed, different optimizers can be used, the size of the training data batches can be varied, the number of epochs trained can be increased or decreased, and of course a different loss function can be chosen. To avoid overfitting on the training data, cross-validation or a validation set are used to determine the best hyperparameter configuration.

## 2.3 Deep learning

In the previous subsection, we have explained some machine learning basic concepts, a few traditional machine learning techniques that are relevant for our purposes, and the basics of artificial neural networks. The part on the ANN was purposefully separated from the other machine learning methods, as it forms the basis for this subsection on deep learning. In this section, we will explain what deep learning is, and we will discuss two deep learning architectures: the recurrent neural network (RNN) and the convolutional neural network (CNN).

### 2.3.1 Deep learning basic concepts

In the part about artificial neural networks, we have shown that a simple feedforward network consists out of an input, hidden, and output layer. It is possible to add more (hidden) layers to the network to make the network deeper, hence the name deep learning. "Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction"[34]. Even though there are multiple definitions of what can be considered a deep neural network, the most simple definition is that when a neural network has multiple (more than 1) hidden layers, it is a deep neural network. The advantage of adding more layers to the network is that those layers are not directly dependent on the training data anymore. The network can extract more features than it was initially given from the data via the additional relationships within the network. Deep neural networks are trained in a similar way to simple ANNs, based on the concepts of gradient descent and backpropagation. Deep neural networks have shown to be more expressive function approximators [35]. In practice, this leads to deep neural networks being capable of learning more complex relationships in the data (sometimes impossible for humans to see), and therefore they have a better performance on many tasks.

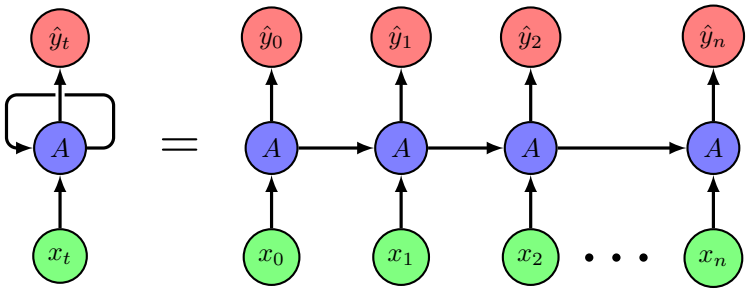



Figure 2: Simple graphical representation of an (unrolled) recurrent neural network

Some issues come along with building a deeper neural network. One of the problems is vanishing (or exploding) gradients when training the network. When the gradients are being propagated back deeper into the network, the gradients tend to get smaller through the layers, causing the network to learn the correct weights in the earlier layers much slower than in the later layers. Luckily, there are multiple solutions to this problem. When using the ReLU activation function, the vanishing gradients problem is less prominent. Another solution is using recurrent units in the network, which can mitigate the problem of multiplying many small values leading to a vanishing gradient. Recurrent units are used in recurrent neural networks, discussed in the next part.

### 2.3.2 Recurrent neural network

Recurrent neural networks (RNNs) are artificial neural networks that are designed to work well with sequential data. The best-known examples of sequential data can be found in natural language processing, for example determining whether a sentence reflects a positive or negative sentiment. When processing the sentence, the words are provided as input one-by-one (sequential). But, the meaning of a word depends a lot on the words before (and after) it. Therefore, an RNN makes use of recurrent units that can capture dependencies between words in a sentence.

As said, the RNN makes use of recurrent units that are not only dependent on the current input but also the output of the previous unit. Recurrent neural networks could have many different architectures, some take a single input and produce many outputs, some take many inputs and produce many outputs, some take many inputs and produce a single output, et cetera. Figure 2 shows that a recurrent unit receives two inputs: the input of the current time step $x_t$ and the hidden state of the previous time step $h_{t-1}$, represented by the arrows. This is a many-to-many architecture, but if only the last output is considered, it is a many-to-one architecture. To determine whether a sentence is positive versus negative or to return a sentiment score, a many-to-one architecture is needed. The equations for a simple recurrent unit are

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \tag{19}$$

$$\hat{y_t} = W_{hy}h_t \tag{20}$$

where $h_t$ and $h_{t-1}$ represent the hidden state in the current time step $t$ and previous time step $t-1$, respectively. $x_t$ is the input of the current time step, $\hat{y_t}$ is the output of the current time step, and $W$ is the set of weights. As with a feedforward neural network, the set of weights is determined via a process called backpropagation (through time).

The issue of the recurrent network presented above is that it is only capable to capture dependencies from prior inputs. To tackle this problem, bidirectional RNNs are created. As the name suggests, this network processes the data in two directions. Bidirectional RNNs are actually two stacked RNNs, one processes the data from the start of the sequence to the end, and the second one processes the data from the end to the start. The output of both RNNs can then be combined by averaging, stacking, or some other transformation.

A problem with simple RNNs is that it is relatively hard to capture longer-term dependencies due to the vanishing gradient problem mentioned in the previous part. Luckily, there are improved recurrent units like the long short-term memory (LSTM) and the gated recurrent unit (GRU) that can tackle this problem effectively. These gated RNNs work based on the idea that some of the information has to be passed through to the next unit and some information can be forgotten, information that is passed through can go through the unit almost fully unmodified. By being able to pass information without modifying it, the gradients can be passed through these units unmodified as well, also removing the vanishing gradient problem.

The LSTM model [36][37] is a unit that receives three inputs at every time step $t$, it receives the current input $x_t$, the hidden state of the previous unit $h_{t-1}$, and it also receives a cell state $c_{t-1}$ of the previous unit. These inputs go through a couple of gates, before leaving the unit again. By going through these gates, the LSTM can forget irrelevant parts of the previous cell's state $h_{t-1}$ based on the current input $x_t$. It can also update the cell state $c_t$ based on the previous cell state $c_{t-1}$, the modified (partly forgotten) previous hidden state $h_{t-1}$ and the current input $x_t$. Then lastly, it can determine its output $h_t$ based on the previous hidden state, the current input, and the current cell state. These three steps are done by the use of a forget, update, and output gate. The gates are basically simple calculations using an activation function (tanh or sigmoid) and a set of weights. A diagram of the structure of an LSTM unit can be found in figure 3.

The GRU model [38][39] is similar to the LSTM unit but simpler. The difference is that the GRU uses a single gate for the forgetting and updating parts of the unit. The gates that the GRU uses are called the update and reset gates. The performance of the LSTM and GRU units

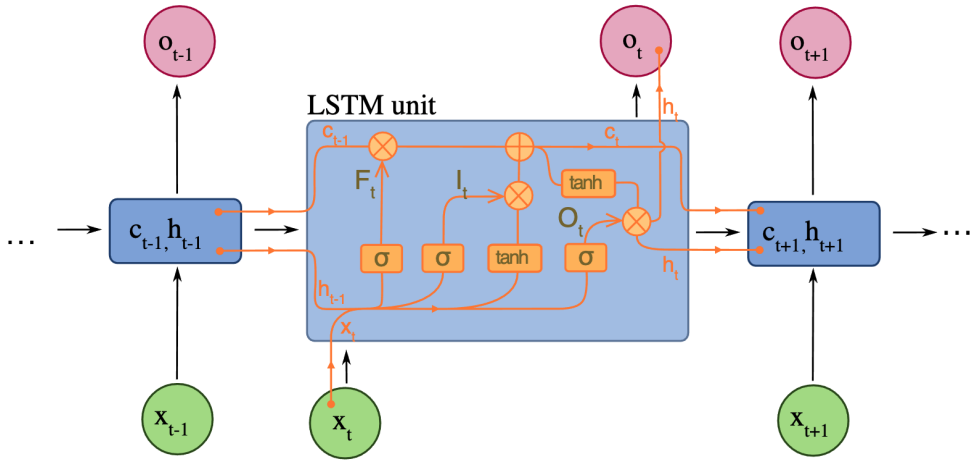



Figure 3: Graphical representation of a LSTM unit, $o_t$ is the output in time step $t$, $F_t$, $I_t$, and $O_t$ are the forget, update and output parts of the unit, respectively. Taken from [40].

are comparable, but the GRU is computationally more efficient [39]. A more detailed explanation of the LSTM and GRU models can be found in Goodfellow (2016) [18].

### 2.3.3 Convolutional neural network

Another type of deep neural network is the convolutional neural network (CNN), an architecture that is specialized in data that contains spatial information. This information can be in a single dimension, like a time series, but also in 2D, like images. Convolutional neural networks have shown impressive performance on computer vision tasks, such as image classification. The convolutional neural network makes use of several mathematical techniques to extract information from the data efficiently, such as convolution, pooling, and an activation function.

The CNN uses the convolution operation to extract information from its input. In this convolution layer of the network, a so-called filter (or kernel) is used that slides over the input data. How this operation works is best explained via an example. For example, the data is a black-and-white image of 30 by 30 pixels and the filter has a size of 3 by 3. All the pixels in the image have a (normalized) value, ranging from -1 for black to 1 for white. The filter has a total of 9 weights (3 by 3), and the filter will slide over the image starting at the top-left and ending at the bottom right. Every step the filter takes, all weights are multiplied with the pixel value underneath it, resulting in 9 values which are summed together (convolution). When the filter has reached the end of the image, a matrix of all these sums is created, called a feature map. If the filter used has proper weights, features can be extracted from the original image. If the weights of the filter would have been positive values for the middle column and zero for

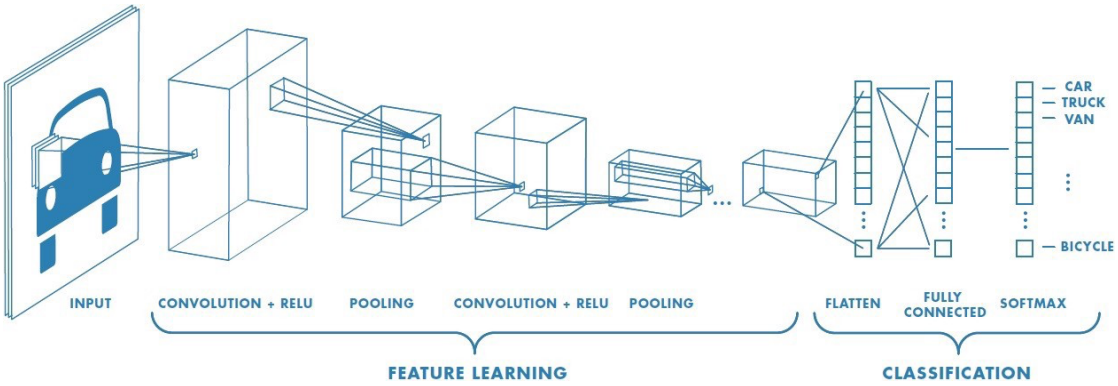



Figure 4: Graphical representation of a convolutional neural network. Taken from [41].

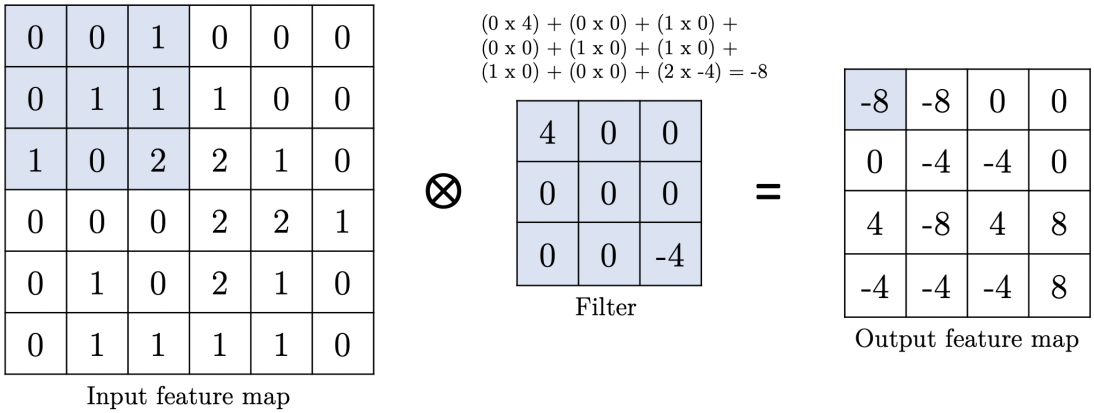



Figure 5: The convolution operation

the left and right column, it was able to detect vertical edges in the image. Multiple filters are used to discover different features from the input, thus generating multiple feature maps. The weights in the filters are learned during training, enabling the network to discover features that might not be directly visible by humans. The learned weights are shared and reused for efficient computation.

The feature maps discovered via the use of these filters are stacked together to a single 3D feature map. Before the data is passed through to the next layer, a non-linearity is introduced by using an activation function that transforms the features. For CNNs, the rectified linear unit (ReLU) is the most used activation function, which basically replaces all negative values on the feature maps with zero. Then, the same operation can be applied again with new (3D) filters, creating new feature maps and extracting new features.

After having multiple convolution layers in the network, the stacked feature maps grow signif-

icantly in size. Due to these large feature maps, many weights have to be learned and optimized during the training phase, which is computationally inefficient. Therefore, a pooling layer is often added. The pooling layer also makes use of a window that slides over the stacked feature maps, but instead of a convolution operation, it takes the maximum or average value of the features directly underneath the window and by doing so, it creates smaller feature maps. Pooling has two advantages, firstly, it reduces the dimensions which makes it more computationally efficient. Secondly, it makes the model invariant to small changes in the input feature map - when the input is slightly changed, the output of the pooling remains the same. This is important in, for example, object recognition because it does not matter whether an object is more to the left or more to the right in the image, it is still the same object.

After several convolution and pooling layers, the resulting feature map is flattened, creating a vector of features. These features can then be fed into a few fully-connected feedforward layers, similar to a "traditional" artificial neural network. These feedforward layers are then connected to an output layer that will return a prediction. This prediction can then be used (trained network) or compared against a true value via a loss function. In the latter case, the weights in the network can then be updated via backpropagation and a form of gradient descent.

## 2.4 Developmental age

In the previous subsections, we have discussed the foundations of EEG, traditional machine learning, and deep learning. We want to use these techniques to determine the developmental age of an infant. Determining the chronological age does not require any sophisticated techniques, it can simply be calculated when the date of birth is known. However, the developmental, maturational, or brain age of a subject is something that has to be determined via other means. Therefore, in this subsection, we will discuss what developmental age is and how it can be determined.

Several techniques can be used to estimate the age of a subject, the means we will use are discussed in section 4. When a prediction using EEG data is made, this is called the brain age, which can also be seen as the developmental age of a subject. The difference between brain age and chronological age is called the brain age gap. The brain age gap can be seen as a measure of whether the subject's brain seems younger or older than its actual chronological age. This brain age gap estimation technique can be used as a tool for determining the development of an infant.

In some studies, like [42], this technique is used to determine the brain age of a subject with a certain disease. The model is then trained on a healthy baseline group, and this model is subsequently used to predict the age of the other group. The brain age estimation on the healthy subjects is considered as being an accurate estimation of the brain age. However, in the case of

infant developmental age estimation, there is no baseline group for which we can consider the estimates as accurate.

It is possible to use a single subject group (instead of a healthy baseline group and another group) where chronological age is a proxy for brain age (or developmental age). However, when using only a single subject group, there has to be some kind of validation that the brain age gap arises from actual developmental age differences, rather than poor model performance in estimating chronological age. The authors of [5] suggest that this can be validated by using a longitudinal study with the same subjects. When predicting the brain age gap of an infant, the gap has to remain relatively stable over multiple points in time for individual subjects. The reasoning behind this is that developmental age differences are relatively stable - it is unreasonable to have a developmental age estimation of five months below the chronological age at an age of 20 months, and then have an estimation of seven months above the chronological age a few months later. Thus, when a model is correctly predicting the developmental age of a subject, the brain age gaps for individual subjects should be stable over time, which is a condition that can be verified. Comparing the developmental age estimation to other proxies can be used to validate the models as well.

# 3 Related literature

In this section, we will summarize previous research on the topics of this thesis and present the state-of-the-art in the field.

## 3.1 (Brain) age and EEGs

The relationship between age and EEG signals has been studied as early as the late 1930s when Lindsley [43] found in a longitudinal study that when children age, more waves with higher frequencies can be found and fewer waves with lower frequencies in some brain regions. During the years that followed, there have been many studies researching the effect of age on the EEG signals, where the signals are often split into different frequency bands.

Multiple studies have seen the effect of age on the EEG signals due to cerebral changes in subjects [44][45][46], specifically in children [47][48][49][50][51][52] and young children as well [53]. These studies suggest that EEG features (delta, theta, alpha, beta, and gamma bands) change as a function of age. Those changes over time can be used for machine learning and deep learning models to determine the age of a subject.

As mentioned in subsection 2.4, the difference between the predicted brain age and the chronological age is called the brain age gap. Franke and colleagues [7] predicted brain age using MRI data and hypothesize that this gap can be seen as a biomarker for brain maturation. This would mean that an estimate of the age lower than the chronological age indicates that the subject has a lower brain maturational level or developmental age. Vandenbosch and colleagues [5] argued that this is only true when the brain age gap (prediction error) is stable over time, and the gap is not due to model misspecification or measurement noise.

## 3.2 Machine learning, EEG, and related research

Before the rise of deep learning, traditional machine learning (ML) approaches were used for classification and regression problems. Nowadays, researchers often attempt to improve performance by the use of deep learning, but it is often still the case that traditional ML methods are on par with or even better performing than deep learning techniques. Therefore, we will also make use of traditional ML approaches to compare against the performance of our deep learning models.

Hosseini and colleagues [54] have reviewed machine learning research for EEG signal processing with bioengineering applications from 1988 until 2018. The downside of this study was that it was mainly focused on classification problems, rather than regression problems (like ours). The researchers found that all dominant methods of ML have been applied to EEG classification

and that there is not a golden standard for ML research with EEG data. For some forms of machine learning techniques for classification, the conclusions can (to some extent) be translated to a regression problem like ours. Such conclusions point out that dimensionality reduction and selection are essential when applying ML techniques to EEG data.

A classical way of preprocessing EEG data to make it more useful (and to extract features) is the Fourier transformation. Hosseini and colleagues [54] find that the wavelet transformation and the autoregressive model are ways to extract features out of EEG data that have had pivotal roles in EEG machine learning research. Also, they find that data preparation methods as principal component analysis (PCA), independent component analysis (ICA), and linear discriminant analysis (LDA) can be used to increase classification accuracy. Lastly, they see that ensemble methods and combining classifiers have shown good performance.

A study that attempts to predict the brain age (gap) of subjects (mean age: 34.8 years) and makes use of these ensemble methods is done by Al Zoubi and colleagues [6]. The researchers focused on the interpretability of the model, and therefore they have not used the best performing feature extraction methods per se, but used features that were more easily interpretable afterward. They separated the features into five groups, amplitude, peak-to-peak, spectral power, connectivity, and fractal dimension, and calculated subsets of features within these groups, statistics like total power, mean, standard deviation, skewness, and many more. After the process of feature reduction (i.e. removing features that highly correlate with others or have low variation among subjects), multiple regression algorithms were used. The algorithms used were: elastic net, support vector regression, random forest, extreme gradient boosting tree, and Gaussian process with a polynomial kernel. After training the models with nested cross-validation, they calculated a weighted average estimate for the subject's age. Using this stack-ensemble approach, they were able to predict the age of the subjects with an MAE = 6.87(0.69) years, and an RSME = 8.46(0.59) years. However, by only using the support vector regression approach with a radial kernel, they already achieved an MAE = 7.01(0.68) and RSME = 8.7(0.63) years.

Similarly, Vandenbosch and colleagues [5] attempt to predict the age of children based on EEG data and also investigated whether the error (underestimation or overestimation) was stable over a longer period, indicating the brain maturational level of the subject. The subjects came from two data sets (N=702, and N=835), with the subject's ages ranging from five to eighteen years old. They removed artifacts from the data using ICA and user fast Fourier transformation to split the data into different bands. The input features used were the power in 1 Hz wide bins from 1 to 24 Hz, correcting for gender differences. The researchers trained three different models: a random forest, support vector machine, and relevance vector machine. The SVM was discarded as the performance was lower on all criteria. For the random forest, they found an

MAE of 1.22 years, and for the relevance vector machine an MAE of 1.46 years. Interestingly, when looking at the age prediction error (brain age gap), they found a moderate to high stability (RF: $0.43 < r < 0.74$; RVM: $0.53 < r < 0.67$), depending on the age group the subjects belonged to. This means that the variance is not fully explainable by noise and/or the model not being fully accurate. Though it still has to be proven, this brain age gap might be a good candidate for being a predictor for individual developmental differences, according to the authors.

Traditional machine learning techniques generally do not use raw EEG data for classification or regression but start with a feature extraction process. Feature extraction could be simple, like in [5] where the spectral power in 1 Hz bins from 1 to 24 Hz was used. Other studies use similar features [55][56][57][58][6], but combine it with additional features. These additional features include, but are not limited to: mean power, squared power, root mean squared power, autoregressive coefficients, Hjorth parameters (activity, mobility, complexity), relative power, power of different bands (alpha, beta, delta, theta, and gamma), power minus power of a certain band, zero crossings, variance, standard deviation, kurtosis, skewness, approximate entropy, edge frequency, and peak-to-peak statistics.

Many different features can be extracted from the EEG data, and also different preprocessing steps can be taken to prepare the data for feature extraction (Fourier, wavelet transform). Therefore, feature reduction methods are used to reduce the dimensionality of the models. Feature reduction methods include linear discriminant analysis, PCA, ICA, looking at the correlation between features, and removing features that are low in variation between subjects. Furthermore, for making predictions using EEG data with ML techniques, the data is often denoised and artifacts are removed. Methods that we have found that were used are a Savitzky-Golay filter [57], removal of certain absolute amplitudes, manual inspection, ICA, and Autoreject [15].

Lastly, according to Roy and colleagues [14] which review deep learning methods for EEG, the state-of-the-art machine learning baseline is the Riemannian geometry-based support vector machine. These have been used as baselines to compare against deep learning models [59][60], and were introduced by Barachant and colleagues [61].

## 3.3 Deep learning, EEG, and related research

In the last few years, the use of deep learning techniques has increased a lot in many different fields of research. However, when looking at our subject specifically - predicting (developmental) age with EEG data - there is only single a study.

Kaushik and colleagues [11] have developed an EEG-based bidirectional LSTM-LSTM neural network for age (and gender) prediction. They used the EEG recordings of 60 subjects, ranging from 6 to 55 years, both males and females (35 vs. 25), and divided the data set into six age

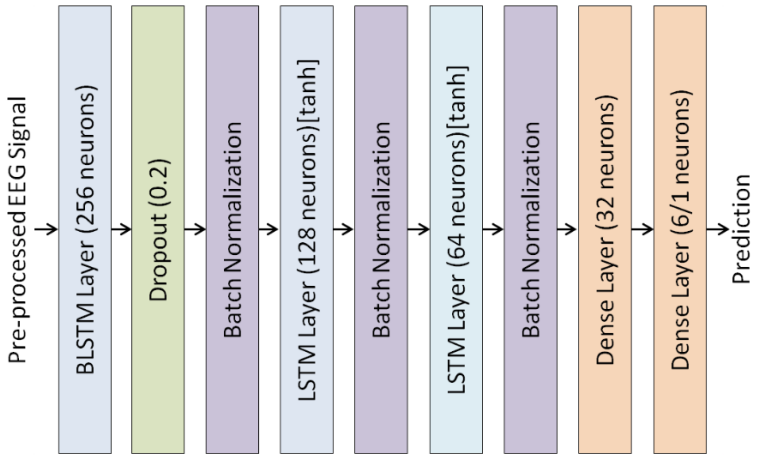



Figure 6: The architecture of the BLSTM-LSTM model by Kaushik and colleagues, with a single BLSTM and two LSTM layers [11].

groups of ten individuals each. These age groups were considered to be the six different classes. The data was preprocessed via a discrete wavelet transformation (DWT), splitting the data into alpha, beta, theta, delta, and gamma brain waves. They trained various LSTM and BLSTM models on the data set, the final model with the best performance was a combined BLSTM-LSTM (figure 6). They achieved the highest classification accuracy of 93.7% when using this deep BLSTM-LSTM with the beta brain waves. The accuracy of the raw EEG data was 83.26%.

Even though the accuracy of Kaushik and colleagues [11] seems quite high, there are some considerable limitations. Firstly, they split the subject into age groups of 10 individuals and predicted to which group someone should belong, rather than actually predicting the subject's age. Predicting the actual age would change the problem from an age group classification problem to an age prediction regression problem. Secondly, they have used a feature extraction technique (DWT) to make the model perform better. This is hence not making use of a key strength of deep learning, namely that the models are often capable of learning different representations (and therefore extract features) using the raw data. Therefore, it would be interesting to see whether it is possible to achieve similar or better performance on raw data compared to the data that is been transformed via feature extraction techniques.

Luckily, though there has not been more deep learning research on our research topic, there has been quite some research on EEGs and deep learning applied to other problems. Roy and colleagues [14] reviewed 154 studies that apply deep learning to EEG data, published between the start of 2010 and mid-2018, making several observations and formulating several recommendations for future EEG research. To begin, one of the main motivations they find for the use of

deep learning techniques is that they can be used on raw data, without any feature extraction, simplifying the processing pipelines. This is in line with the second comment on the Kaushik and colleagues [11] study mentioned above.

Furthermore, deep learning techniques tend to require more (training) data than traditional machine learning techniques. Roy and colleagues [14], however, argue that the amount of data necessary is already available, and otherwise, data augmentation techniques can be used to improve performance. Data augmentation is a technique to artificially generate more data, to improve performance, stability, and the generalization of the model. Data augmentation techniques include the use of overlapping sliding windows, adding Gaussian (white) noise to the data, the use of a generative adversarial network (GAN), swapping left and right side electrodes, or use data that is discarded during down-sampling. Another means to improve performance when little data is available is transfer learning, using a modified model trained on a different data set and train (parts of) the model again on the current data set, optimizing it for the task at hand.

Also, regarding the EEG preprocessing stage, they observe that more often raw EEG data is being used, rather than extracted features. They report that in some studies, the performance on data that is not preprocessed is higher than on preprocessed data. However, they did not notice a clear preference for either raw data or features that are extracted from the data. Also, artifact removal is not explicitly mentioned in most papers, suggesting that this is not required as a preprocessing step. Therefore, they recommend using raw EEG data as a starting point for a new study.

When looking at the architectures used for these EEG problems, the authors expected RNNs to be the dominant architecture due to the temporal structure of the data. Yet, they found that more than half of the architectures were CNNs. Unfortunately, they were not able to draw conclusions on which architecture was best, and they suggest that studies evaluating architecture choice should be done to learn more. They also observed that most models had a depth of fewer than five layers, and that performance often decreased when using more layers. Nonetheless, they hypothesize that hyperparameter tuning is key to increase the performance of deeper models. Hyperparameter tuning was done in only 20% of the studies, and therefore a major area of improvement for most studies.

After that, the authors have looked at how studies performed some kind of model inspection. As mentioned in the introduction, deep learning models are often seen as black boxes, and if a model is going to be used in a clinical environment, it should be clear how certain a model is and why it is making a decision. Model inspection techniques that were used consist out of weights analysis, activation analysis, input-perturbation network-prediction correlation maps, generating

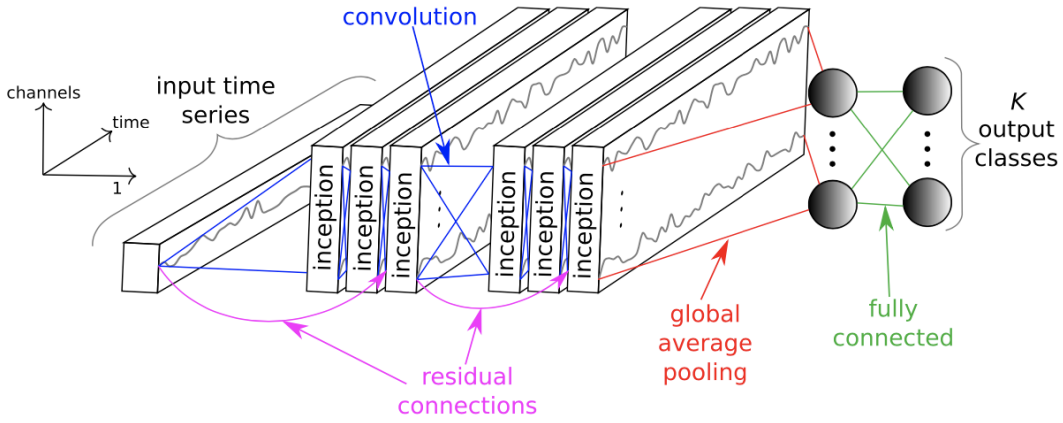



Figure 7: InceptionTime network by Fawaz and colleagues for classification [63]

input to maximize activation, and occlusion of input. The authors noticed that artifacts in the data often impact the decision-making process of the model. This might be a reason to use an artifact removal technique, even though it was mentioned before that most studies did not do so.

Lastly, the authors observe that there are several improvements to be made regarding how results are being reported. Sometimes the baselines were very simple, instead of making use of state-of-the-art baselines that have been used in similar studies. Also, results can often not be replicated due to the lack of reproducibility standards. Therefore, they recommend authors to provide as much information as possible on the baseline models used and how to replicate the results of the study.

Another indication that predicting age with deep learning based on neuroimaging scans is definitely possible is a study by Cole and colleagues [62], where they use (raw and processed) MRI data and a CNN to predict the age of the subjects. They can predict the age (in years) on raw data with a mean absolute error (MAE) of 4.65 years. Their data set consisted out of 2001 individuals (male vs. female = 1016 vs. 985, mean age = 36.95±18.12, age ranges from 18 to 90 years). As mentioned in subsection 2.1, MRI scanners are more expensive than EEG scanners, and therefore it would be beneficial (cost-wise) if comparable or better results can be achieved using EEG data.

As many different deep learning architectures can be used for predicting developmental age with EEG data, it is interesting to consider a study that has reviewed different architectures on time series data classification. Fawaz and colleagues [64] have trained 8730 deep learning models on 97 different time series data sets, comparing the performance of state-of-the-art (non-ensemble) architectures. The approaches they have compared are the fully-connected neural network (called multi layer perceptron in their review) [65], (fully) convolutional neural network

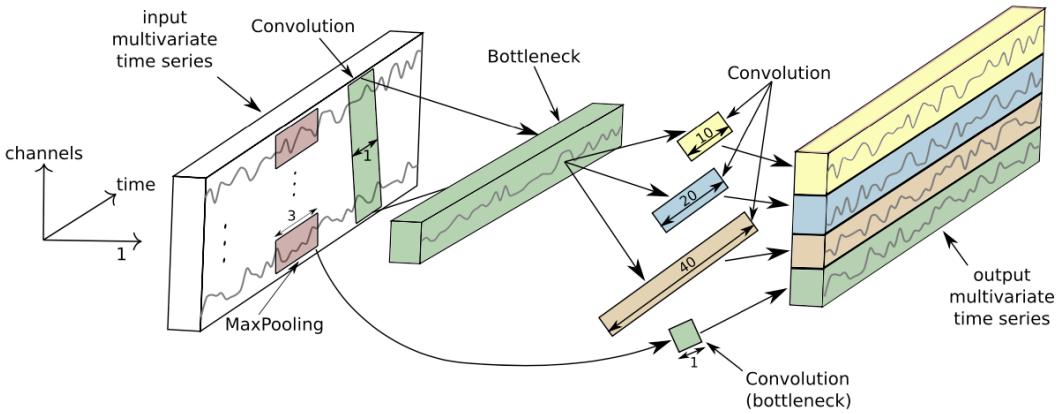



Figure 8: Inception module by Fawaz and colleagues, with a bottleneck layer size $m = 1$ [63]

[65], residual network (ResNet) [65], encoder [66], time convolutional neural network [67], multi-scale convolutional neural network [68], Time Le-Net [69], multi-channel deep convolutional neural network [70][71], and time warping invariant echo state network [72].

For our research, the first five architectures are most interesting. Firstly, these five can be relatively easily be re-implemented and modified as a regression model for our EEG data set without having to re-implement a data generator for all of them. Secondly and coincidentally, they happen to be the best performing according to the authors, and therefore it would be interesting to use these architectures for the current study.

Next to the review, Fawaz and colleagues also developed another architecture called InceptionTime [63], shown in figure 7, which is an ensemble of convolutional neural networks. This model is inspired by the success of Inception-based networks for various computer vision tasks. This architecture makes use of so-called inception modules (figure 8) and residual connections and can be adapted for regression problems as well. In section 4, the architecture is explained in-depth. The authors have compared the model's performance to the ensemble algorithm HIVE-COTE [73], the state-of-the-art algorithm for time series classification, and the performance is on par. Additionally, it is scalable to larger data sets, making it an interesting architecture to apply to our EEG data set.

All things considered, even though there are not many studies on age prediction with EEG data using deep learning techniques, there are many recommendations made based on other (EEG) deep learning research that can be used to improve the quality of our study.

# 4 Methods

In this section, we outline the methods used for this thesis research. First, we look at the data set used, the different prepossessing and feature extraction steps for both traditional machine learning techniques and deep learning techniques. Then, we will discuss the traditional machine learning models we use and the deep learning architectures we have considered. After that, we will explain how we predict brain age. Lastly, we will discuss how the models are assessed and validated and how we attempt to make them more explainable. All the code for preprocessing, the models, and model validation is available on GitHub (https://github.com/epodium/EEG_age_prediction).

## 4.1 Data

The data set used for this study comes from the Dutch Dyslexia Programme (DDP) [10] which started in the late nineties and of which the first part was completed in 2002. This longitudinal study consisted out of 180 children with a familial risk of dyslexia (FR) and a comparison group of 120 children without FR (noFR). The subjects were followed from the age of two months up to nine years, and they performed three kinds of measurements: (1) neurophysiological measures (EEG/ERP), (2) behavioral assessments, and (3) questionnaires. Between the age of two months and 47 months, the researchers measured event-related potentials (ERP) every half year. These measurements will be the primary data we use for this study.

Furthermore, the researchers have collected some other data points that can be used as proxies for development to validate our models. The parents filled out questionnaires about the vocabulary of the children, monitoring the expressive and receptive language of the child, which we could use as a proxy. Also, when the children reached the age of 7 to 9 years old, reading tests were administered to determine whether the FR children actually had dyslexia or not, splitting the data into three groups, FR with and without dyslexia and the noFR group. This data can be used to compare predicted developmental age against dyslexia.

For this study specifically, a subset of the ERP data set was used. We used the data of children between the ages of 11 and 47 months with 6-month intervals. The data of younger children was excluded, as the recordings of these children contained a lot of noise and artifacts. This is most likely caused by the children moving substantially during recording.

As mentioned, the children were exposed to stimuli during the ERP recordings. These acoustic stimuli were on a continuum in nine steps from the Dutch words “bak” (bin) /bAk/ to “dak” (roof) /dAk/. The words on the “bak” side of the continuum were used as standard in the original study, whereas the “dak” side of the continuum was used as deviant. For our study, we have only used the “bak” side of the continuum. We chose to exclude the other stimuli, with the

aim that all the stimuli used sounded almost the same without discarding too much of the data (i.e. almost what would have happened when choosing a single stimulus). These chosen stimuli were coded with the event IDs 2, 3, 4, 5, 12, 13, 14, and 15. All other events were ignored. Please refer to the original study for a more extensive explanation of the stimuli and the data collection process [10].

After removing empty recordings, the data set consisted out of data for 304 subjects. Not all subjects have recordings for all ages on the 6-month intervals. The number of subjects per age are 166 (11 months), 109 (17 months), 255 (23 months), 230 (29 months), 160 (35 months), 237 (41 months), and 32 (47 months).

For the recordings, a 10-20 electrode montage with either 62 or 30 channels was used. When the recording used 62 channels, we only used the 30 channels that were also used for the 30 channel setup. The names of the 30 channels used are `O2, O1, OZ, PZ, P4, CP4, P8, C4, TP8, T8, P7, P3, CP3, CPZ, CZ, FC4, FT8, TP7, C3, FCZ, FZ, F4, F8, T7, FT7, FC3, F3, FP2, F7, FP1`. The sampling rate was 500 Hz, and we used 1-second epochs, 200 milliseconds before the stimulus and 800 milliseconds after. Therefore, the shape of a single epoch was 501 samples for 30 channels: $(501 \times 30)$.

During a recording, the subjects hear the stimuli many times, and therefore many epochs per subject are collected. The events of a single recording are saved in a single file, resulting in 2093 files. All these files combined contain 1,368,001 epochs. The number of epochs per age are 238,179 (11 months), 125,632 (17 months), 281,091 (23 months), 251,144 (29 months), 190,245 (35 months), 250,638 (41 months), and 31,072 (47 months).

The (pre)processing of the data was done using the MNE-Python package [74]. This includes the extraction of the epochs, a low-pass (0.5 Hz) and high-pass (40 Hz) filter (firwin method), and the interpolation of bad channels. Channels were considered bad when their standard deviation was five times higher or lower than the average standard deviation in more than 20% of the epochs in a file. This is also the case when the min-max difference is five times higher or lower than the average min-max difference. If there were more than two bad channels, the file was ignored. Otherwise, the bad channels were interpolated (spherical spline interpolation) via MNE-Python, if necessary. We did not use any other specific artifact removal techniques, as [14] suggests that deep learning techniques might be a way to avoid doing this without harming performance.

As a target variable, that is to be predicted by the models, we have the ages of almost all the children in days, months, and years. If this information was missing, we used the "age group" to fill out these missing values. In the original data set, the files are saved in a separate folder per age group or category, i.e. all the files for approximately 11-month old children are in the same folder. The age group is the age the child approximately should have been when the recording

took place and therefore it is reasonable to assume that the actual age matches the age group quite well. For training and predicting, the age in months is used as the target variable.

Further data preprocessing steps were dependent on the type of model used, either a traditional machine learning model or a deep learning model. Therefore, these additional preprocessing steps are discussed in the following subsections.

## 4.2 Preprocessing for traditional machine learning models

As mentioned in 3.2, traditional machine learning techniques generally do not use raw EEG data but start with a feature extraction process. As there does not seem to be a standard feature extraction procedure, we have extracted most of the features we found in other research and then proceeded with a feature reduction step.

All the features were extracted per channel. The features (times 30 channels) we originally extracted were the mean, root mean squared, hjorth mobility, hjorth complexity, variance, standard deviation, kurtosis, skewness, approximate entropy, zero crossings, band energy, spectral edge frequency, and peak-to-peak amplitude. This resulted in a total of 510 features per epoch, almost 30 times fewer than the raw data. These features were extracted using the MNE-Features Python library [75].

Furthermore, we have applied a simple feature reduction step by looking at the correlation between features. This was done to remove features that are very similar, and therefore redundant. The standard deviation and peak-to-peak amplitude features were removed because they had a high ($> 0.90$) correlation with each other, root mean squared, and variance. This reduces the number of features to 450 per epoch.

For most of the traditional machine learning models, the data set was split into a training (85%) and test (15%) set. For the simple neural network model, a validation set (15%) was used as well, which was taken from the training set. Samples from a single subject, no matter in which age group, were not allowed to be in multiple sets.

## 4.3 Preprocessing for deep learning models

Deep learning models are capable of extracting features out of large data sets, without the help of (manual) feature extraction steps. As suggested in [14], this is also the case for EEG data. Therefore, there are no additional feature extraction steps for the deep learning models. The processed data as described in section 4.1 is used.

However, due to the size of the data set, we had to make use of so-called data generators instead of loading the full data set into memory. This allowed us to add apply noise reduction

and data augmentation techniques.

The subjects were divided into three groups: train (70%), validation (15%), and test (15%). During training, all the files of the subjects in the training set are iterated over 30 times per training epoch. When data is sampled from a file, 30 random EEG epochs are chosen from this file and these are averaged. When these 30 EEG epochs are averaged, the resulting epoch contains less noise. At the same time, since the number of possible combinations of EEG epochs is very large (varying per subject) and therefore not all combinations will be seen during the training process, this serves as a data bootstrapping step. Consequently, the model will be trained on constantly varying (averaged) EEG epochs. Next to this, we also added some Gaussian noise ($\mu = 0, \sigma = 0.01$) to the final EEG epoch for data augmentation.

## 4.4 Traditional machine learning models

The traditional machine learning models presented in this thesis research have all been implemented in Python, and make use of the Scikit-Learn framework [76]. The fully-connected neural network has been implemented using Tensorflow with Keras [77]. The code can be found on GitHub. In the following parts of this subsection, we will briefly discuss the models and their (hyper)parameters.

### 4.4.1 Random forest

As the random forest regressor (`RandomForestRegressor`) we used the implementation from Scikit-Learn, which comes with only a few parameters that can be changed to have a very significant positive impact on performance. We performed a grid search on two hyperparameters: (1) the maximum number of features to consider in a split (`max_features`) and (2) the minimum sample leaf size (`min_sample_leaf`).

In the final random forest, we used 100 trees, the maximum number of features was 250 (out of 450), and the minimum sample leaf size was 10. All the other parameters have been set to the default values for the `RandomForestRegressor` of the Scikit-Learn framework.

### 4.4.2 Linear support vector regressor

The linear support vector regressor was implemented because it can handle large data sets, which often is an issue with (non-linear) support vector regressors. The `LinearSVR` of Scikit-Learn was used, and the data was standardized by using the `StandardScalar` functionality.

For the parameter search, we decided to randomly divide the training set into 100 subsets and use one of these subsets, to make it computationally feasible to perform a parameter search

for the model. We then first performed a randomized search in a large parameter space with only one subset of training data, to find reasonable parameter magnitudes. Then, we used 10 subsets of the data to perform another randomized search to narrow down the parameter space further. Lastly, we performed a grid search on these 10 subsets to find the final hyperparameters and we fit the model on the full training set with these parameters. We did not do a grid search on the whole data set itself due to the extensive computational cost of doing this.

The parameters searches were performed on two hyperparameters: (1) the regularization parameter (`C`) and (2) the epsilon parameter. In the final linear support vector regressor, we used a regularization parameter of $C = 0.45$, and an $epsilon = 2.5$. The maximum number of iterations of the support vector regressor during fitting was increased to 50,000, making sure it was able to converge. All the other parameters have been set to the default values for the `LinearSVR` of the Scikit-Learn framework.

### 4.4.3 Support vector regressor

For the support vector regressor, a somewhat different approach was used. Because the fit time complexity is more than quadratic with the number of samples, we again decided to randomly divide the training set into 100 subsets and only use one of these subsets for the parameter search. We used 10 subsets to fit the final support vector regressor to make it computationally feasible.

The `SVR` of Scikit-Learn was used, and the data was standardized by using the `StandardScalar` functionality. We then performed a randomized search in the parameter space to find reasonable parameter magnitudes, followed by a grid search in the parameter space. The parameters searches were performed on five hyperparameters: (1) the regularization parameter (`C`), (2) the epsilon parameter, (3) the gamma parameter, (4) the kernel used, and only for the polynomial kernel also (5) the number of degrees.

In the final support vector regressor, the hyperparameters we used were $C = 20$, $epsilon = 2$, and $gamma = scale$ with a radial basis function kernel. The gamma *scale* parameter means that it is set to one divided by the number of features (450) multiplied by the variance of all the features. The maximum number of iterations of the support vector regressor during fitting was increased to 50,000, making sure it was able to converge. All the other parameters have been set to the default values for the `SVR` of the Scikit-Learn framework.

### 4.4.4 Relevance vector regressor

We used the relevance vector regressor from the sklearn-rvm library (https://github.com/Mind-the-Pineapple/sklearn-rvm). Due to the same reason as the support vector regressor, we used

a subset (1/100th) of the training data for fitting this model. The `EMRVR` of the library was used, and the data was first standardized by using the `StandardScalar` functionality. Due to the computational complexity of fitting this model, we did not perform a randomized or grid search. As support vector regression and relevance vector regression are somewhat similar, we decided to train the model on the best parameters found for the support vector regressor.

In the final relevance vector regressor, the hyperparameters we used were $C = 20$, $epsilon = 1.5$, and $gamma = scale$ with a radial basis function kernel. The gamma *scale* parameter means that it is set to one divided by the number of features (450) multiplied by the variance of all the features. All the other parameters have been set to the default values for the `EMRVR` of the library.

### 4.4.5 SGD regressor

We have also explored the use of an SGD regressor, as recommended by the Scikit-Learn framework's documentation for large data sets. After a randomized parameter search using `RandomizedSearchCV` to find reasonable parameters, and a grid search using `GridSearchCV`, we found that the best SGD regressor was making use of the epsilon-insensitive loss. According to the documentation, an SGD regressor with this loss is essentially a support vector regressor. As the performance of the SGD regressor was lower than the SVR itself, we did not pursue exploring this model further.

### 4.4.6 Fully-connected neural network

The last traditional machine learning model we have explored is a fully-connected neural network. We explored several architectures and eventually decided to use the architecture of the fully-connected neural network that can be found in the next subsection, please refer to the next section for the exact architecture. This is not necessarily a traditional method like the previous methods. However, by using the traditional machine learning feature extraction pipeline, we can make a comparison between a neural network with extracted features and one trained on raw data.

## 4.5 Deep learning models

The deep learning models presented in this thesis research have all been implemented in Python, and make use of the Tensorflow with Keras library [77]. All the code can be found on GitHub. The first five architectures are modified versions of models that can be found in Fawaz and colleagues' [64] review for deep learning in time series classification. The other three models in this review

did not work with our data generator strategy. However, they were also the worst-performing in the review, therefore we expect they would not be the best for our problem either. Furthermore, we explored the InceptionTime architecture [63] with additional architecture adjustments and the only other deep learning model we were able to find for EEG-based age prediction, the BLSTM-LSTM model [11].

For all models, we have performed a search for hyperparameters. We have also tested minor changes to the model's architectures, e.g. adding or removing layers, changing the number of filters, adding an activation function to a layer that did not have one. These changes were all tested on a reduced data set. To construct this reduced data set, the original data set size was reduced by a factor of ten by averaging chunks (without replacement) of 10 EEG epochs, resulting in a smaller data set. Later, during the model training stage, not 30 (for the original data set) but only three EEG epochs were averaged to create a final averaged EEG epoch, see subsection 4.3 for the original pipeline. Using this reduced data set resulted in faster training, but also lower performance and was therefore only used for choosing the best architecture, but not for the final model training. In the following parts of this subsection, we will briefly discuss the final models' architectures, hyperparameters, and how we have optimized the models.

### 4.5.1 Fully-connected neural network

The fully-connected neural network is the simplest architecture we have implemented for the different deep learning models. This architecture was initially proposed by Wang and colleagues [65] and is called Multi Layer Perceptron (MLP) by Fawaz and colleagues [64]. The model contains four layers, all fully connected to the output of the previous layer. The original model's output layer was a softmax classifier, but we have removed this. For our regression purposes, we changed it to a dense layer with a single node, giving a single value as output. The three hidden layers consist out of 500 nodes, with a ReLU activation function. All layers (the output layer as well) are preceded by a dropout layer. The dropout rates are 0.1, 0.2, 0.2 and 0.3, respectively.

The model was trained using the Adadelta optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not decrease for 200 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 5000 epochs unless the validation loss did not improve for 1000 epochs, then training was stopped early. Only the best model based on the validation loss was saved.

### 4.5.2 Convolutional neural network

The second architecture we have implemented is the (fully) convolutional neural network. This architecture was also initially proposed by Wang and colleagues [65]. The model has three convolutional blocks containing a convolution operation, batch normalization, and ReLU activation. The output of the last convolutional block is fed into a global average pooling (GAP) layer, which takes the average of all the time steps in all the channels. This is then passed into a softmax classifier. We changed this part (the GAP layer and softmax classifier) of the network to an average pooling layer with a pool size of 50, a dense layer with 128 nodes and ReLU activation, and a final single-node dense layer for our regression purposes.

All the convolution operations have a stride of 1 and a zero-padding, making sure the input and output lengths are of the same length. The first convolution operation has 128 filters with a kernel size (filter length) of 8. The second convolution operation has 256 filters with a kernel size of 5. The last convolution operation has 128 filters and a kernel size of 3. Unlike the feedforward neural network, no form of regularization was used in this architecture.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not increase for 50 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 2000 epochs unless the validation loss did not improve for 250 epochs, then the training was stopped early. Only the best model based on the validation loss was saved.

### 4.5.3 ResNet

The third architecture we have implemented is a residual network (ResNet). This is the last architecture proposed by Wang and colleagues [65]. The architecture is made up out of 9 convolutional layers, a global average pooling (GAP) layer, and an output layer. Unlike the original model, we have changed the output layer from a softmax classifier to a single-node dense layer. Different than a "regular" convolutional neural network, the ResNet has shortcut connections between the input of a residual block to the output of this block.

The convolutional layers are split into three residual blocks. These residual blocks consist out of three convolutional layers. The activation outputs of the first two layers are fed as input into the next convolutional layer, the output of the third layer, and a shortcut connection from the input of the residual block together compose the output of the residual block. All convolution operations in the first residual block have 128 (instead of 64 in the original model) filters, the second and third block convolution operations have 128 (instead of 256 in the original model) filters. In each residual block, the first, second, and third convolution operations have a kernel

size (filter length) of 8, 5, and 3, respectively. The convolution operation is followed by batch normalization and a ReLU operation in the convolution layers.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not increase for 50 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 1500 epochs unless the validation loss did not improve for 250 epochs, then the training was stopped early. Only the best model based on the validation loss was saved.

### 4.5.4 Encoder

The next architecture we have implemented is an Encoder network, a hybrid deep convolutional neural network inspired by the second architecture we have implemented [65]. This model was originally proposed by Serrà and colleagues [66]. One of the differences is that this model makes use of an attention layer instead of a global average pooling layer. Like all other models we implemented, we changed the output softmax classifier to a single-node dense layer for our regression problem.

The model first has three convolution layers with 128, 256, and 512 filters, respectively. These kernel sizes (filter lengths) for these three layers are 5, 11, and 21, respectively. All the layers have a stride of 1 and zero padding, similar to the (fully) convolutional neural network. In each of the convolution layers, the convolution operation is followed by instance normalization [78] and a parametric rectified linear unit (PReLU) [79] activation function. Lastly, all the convolution layers then have a dropout layer with a rate of 0.2. The first two convolution layers also have a max pooling of size 2 before its output is fed into the next convolution layer. The last convolution layer does not have this and instead its output is fed into an attention mechanism [80]. In this attention mechanism, the output of the third convolution layer is multiplied with a copy of itself that has gone through a softmax operation. This copy that went through a softmax operation enables learning a weight (importance) for all of the time steps in the time series. Finally, before the output layer, there is a last dense layer with 256 nodes, followed by instance normalization and flattening operations.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 (instead of 0.00001 in the original model) and the learning rate was halved when the loss on the validation set did not increase for 50 epochs. The minimum learning rate was 0.00001. The batch size was 10. The model was trained for 1000 (instead of 100 epochs) unless the validation loss did not improve for 300 epochs, then the training was stopped early. Only the best model based on the validation

loss was saved.

#### 4.5.5 Time-CNN

The fifth architecture we have implemented, and the last one we implemented of the architectures found in the Fawaz and colleagues deep learning review [64], is the Time-CNN. This model was originally proposed by Zhao and colleagues [67]. Interestingly, this model originally does not have a softmax classifier as output, but a single-node dense layer with a sigmoid activation function. We removed the sigmoid activation for our regression purposes. Furthermore, the model has two convolution layers. The first layer has a convolution operation with 6 filters, a kernel size (filter length) of 7, and a sigmoid activation function, followed by an average pooling layer with a pool size of 3. The second layer is the same but uses 12 filters instead of 6. The output of the second layer is then flattened and fed into the output layer.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not increase for 50 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 2000 epochs unless the validation loss did not improve for 250 epochs, then the training was stopped early. Only the best model based on the validation loss was saved.

#### 4.5.6 InceptionTime

The sixth deep learning architecture we have implemented, is the InceptionTime architecture, as proposed by Fawaz and colleagues [63]. The architecture of the InceptionTime model has some similarities to the ResNet architecture, but instead of three residual blocks, it makes use of two inception blocks. These inception blocks are made up out of three inception modules (instead of convolution layers). Similar to the ResNet, there is a shortcut between the input of the block and the last module/layer of the block. After the second inception block, the original InceptionTime model uses a global average pooling (GAP) layer which is then fed into a softmax classifier. We changed this part of the network to an average pooling layer with a pool size of 50, a dense layer with 128 nodes and ReLU activation, and a final single-node dense layer for our regression purposes. We replaced this final part because the original model's performance was sub-optimal and we expected this was due to time-information loss in the global average pooling layer.

The inception modules first have a so-called bottleneck layer, which is a convolution operation with 32 filters, a kernel size (filter length) of 1, and a stride of 1. This is normally used for reducing the dimensions of the input, in our case, it increases the number of dimensions from 30 to 32. Secondly, the module uses multiple convolution operations using 32 filters of different kernel sizes

(filter lengths) on the output of the bottleneck layer. In our case, we have used three kernel sizes: 40, 20, and 10. Next to this, a max pooling operation with a pool size of 3 is performed on the original input of the module, followed by a bottleneck layer. The outputs of the convolution and the max pooling parts are concatenated, batch normalization is applied and a ReLU activation is used to construct the output of the inception module.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not increase for 20 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 1500 epochs unless the validation loss did not improve for 100 epochs, then the training was stopped early. Only the best model based on the validation loss was saved.

### 4.5.7 BLSTM-LSTM

The last deep learning architecture we have implemented is based on the BLSTM-LSTM model from Kaushik and colleagues [11]. This model is the only one we found that was specifically designed to classify subjects' gender and age based on EEG data. We had to modify the architecture's output because we have a regression rather than a classification problem. The model we used started with a bidirectional LSTM with 256 nodes and a hyperbolic tangent (tanh) activation function, returning the last sequence and concatenating the outputs of both directions. This was followed by a dropout layer with a rate of 0.2 and batch normalization. After this, two blocks of LSTM and batch normalization followed, again with hyperbolic tangent activation and 128 and 64 nodes for the first and second block, respectively. Both blocks returned the sequences, whereas this only was done for the first one of the two blocks in the original model. The output of this was fed into a dense layer with 32 nodes and no activation function. This last layer connected to the output layer which was a single-node dense layer, providing a single value as output.

The model was trained using the Adam optimizer on the mean squared error loss. The weights were Glorot uniform initialized [28]. The initial learning rate was 0.01 and the learning rate was halved when the loss on the validation set did not increase for 20 epochs. The minimum learning rate was 0.0001. The batch size was 10. The model was trained for 1500 epochs unless the validation loss did not improve for 200 epochs, then the training was stopped early. Only the best model based on the validation loss was saved.

## 4.6 Brain age prediction

As discussed in the subsections 4.2 and 4.3, the preprocessing for traditional machine learning and deep learning is different. For the traditional models, features are extracted from the raw EEG data and this is used as input for the models. All of these samples with extracted features have a target, which is the subjects' actual age in months, associated with them. For the deep learning models, the data fed into the model is coming from the data generator, which is randomly sampling 30 EEG epochs from the subject at a specific age and averaging them to a single EEG epoch. Also, these samples have a similar target associated with them (age in months).

For predicting brain age we do not have a true value. Therefore, we use the chronological age (the target) as a proxy. The prediction error between the predicted age and the chronological age can then be interpreted as higher or lower brain maturational level or developmental age. In subsection 4.7, we discuss how the model is validated to verify whether using this as a proxy is correct or not.

The actual prediction is different for the traditional models and deep learning models as well. For the traditional models, the models make predictions for all the samples (extracted features from EEG epochs) of a single subject at a specific age. For all these predictions, the median is taken as the final brain age prediction for this subject at this age.

For the deep learning models, all the samples (raw EEG epochs) of a subject at a specific age are randomly divided into non-overlapping subsets of 30 epochs. The EEG epochs in these subsets are averaged and fed into the deep learning model for prediction. The median of all the predictions on these averaged subsets is taken as the final brain age prediction for the current subject at this specific age.

We have also considered different strategies for brain age prediction. For traditional machine learning methods, we considered averaging of features or EEG epochs like we had done with the deep learning methods. However, this would most likely have resulted in the loss of frequency information in the data, which is extracted in multiple features in the traditional methods preprocessing pipeline.

For the deep learning methods, we have tried averaging all EEG epochs and making a single prediction, and we have tried making predictions for all EEG epochs separately and calculating the mean or mean of this. Both methods led to (much) lower performance and were therefore not explored further.

## 4.7 Performance measures and model validation

All the models that were implemented have been assessed similarly. Firstly, the performance measures root mean squared error (RMSE), mean absolute error (MAE), and R-squared were calculated by comparing the subjects' age predictions to their chronological ages. Furthermore, the models were assessed by looking at the prediction error stability. We attempted to validate the models by comparing the model's predictions with the vocabulary size. Lastly, a comparison of the model's prediction with the subject's dyslexia predisposition was made. These methods are discussed in the following parts of this section.

### 4.7.1 Error stability

In our models, we are using the chronological age of a subject as the target and this is used as a proxy for developmental age. As already mentioned in the background and related literature sections (sections 2 and 3), there is a need for validating this proxy. The authors of [5] suggest that this can be validated by using a longitudinal study with the same subjects. When predicting the brain age of an infant, the brain age gap (prediction error) is expected to remain relatively stable over multiple points in time for individual subjects. The reasoning behind this is that developmental differences (lower and higher development) are relatively stable. This being true might indicate that the error is a predictor for individual development [5][7] and the brain age gap arises from actual developmental age differences. If this is not the case, then the brain age gap (prediction error) is more likely to arise from model misspecification or measurement noise and the model just has poor performance in estimating chronological age.

The error stability was determined by calculating the correlation between the prediction errors (brain age gap) of subjects in the test set at baseline time $t$ compared to their own prediction error at follow-up time $t+1$. The prediction error was calculated by subtracting the chronological age from the predicted age.

A positive correlation would indicate a stable error, meaning a higher or lower predicted age (than chronological age) at time $t$ would also show a higher or lower prediction at follow-up time $t + 1$, respectively.

### 4.7.2 Vocabulary size

A means to validate the models is by comparing the active (expressive) and passive (receptive) vocabulary sizes of the subjects at specific ages to the brain age gap at that age. The researchers of the DDP [10] asked the parents of the subjects to fill out questionnaires about the vocabulary of the children, monitoring the expressive and receptive language of the child. The receptive

vocabulary is defined as the number of words a subject understands, the expressive vocabulary is the number of words a subject understands and says. In our models, we assumed that chronological age can be used as a proxy for developmental age. Vocabulary size can also be considered as another proxy for a part of the development in children. Therefore, to see whether our models indeed capture the developmental age of the children, the brain age gaps can be compared to this other proxy.

We have calculated this relationship on an "age group" basis, meaning that we have separately calculated the relationship between the brain age gap and the vocabulary size when the children were approximately 17 months, 23 months, 29 months, and 35 months old. We then calculated the correlation between the active and passive vocabulary compared to the brain age gap, respectively. A positive correlation indicates that there is a relationship between (signed) brain age gap and vocabulary size.

### 4.7.3 Dyslexia

The last method we use to assess the models is by looking at the data on dyslexia we have for our subjects. As the data collection was done within the context of the Dutch Dyslexia Programme research [10], we know for many subjects whether they have been diagnosed with dyslexia halfway group five (third grade) of elementary school when the subjects were about 7-9 years of age.

This dyslexia information can be used in two ways. Firstly, it can be used as a means to gain more supporting evidence of the model's correctness. We do not know whether there is a relationship between dyslexia predisposition and developmental age prediction. If dyslexia is considered a condition related to the development of a child, then there might be a relationship between being diagnosed with dyslexia and developmental age. This can then be demonstrated by comparing distributions of developmental age predictions for children with dyslexia versus those without dyslexia. If these significantly differ, this is supporting evidence that the models can differentiate children with higher and lower development.

Secondly, this can be used for interpretation in the opposite way. If these distributions indeed differ significantly, then the developmental age prediction could assist in finding dyslexia predisposition.

The comparison was done by first selecting the subjects of which we had dyslexia information. Then, they were separated into two groups, a group of subjects diagnosed with dyslexia and a group without. For both groups, we calculated the brain age gaps (prediction errors), resulting in two distributions. To see whether these distributions significantly differ, we have performed a one-sided Kolmogorov-Smirnov test. This procedure was followed for each of the age groups.

## 4.8 Cross-validated best model

From a deep/machine learning or artificial intelligence perspective, models are trained and then compared to determine which model is best. This can be achieved by splitting the data into different sets and then validate the model on the test set. However, from the perspective of researching the developmental age of the subjects, this introduces a problem. In the end, we would prefer to have developmental age predictions for all the subjects and not only the subjects in the test set.

Ideally, all the models are k-fold cross-validated, enabling making predictions for all subjects with all models. Unfortunately, this is problematic, especially for the deep learning models, because this would require training all the deep learning models 7 times (in the case of 7-fold cross-validation). Training this many models is computationally unfeasible, and therefore we searched for the best-performing model using the methodology described in the subsections above.

Finally, we have picked the best-performing model and retrained this model on the whole data set using 7-fold cross-validation. This means we have split the subjects into 7 equally-sized groups. We then trained seven models using the EEG data of five folds as the training set, one fold as the validation set, and one fold as the test set. Every model used different train, validation, and test sets. Each model makes developmental age predictions for the excluded test set. By using this technique, we can make developmental age predictions for the whole data set, without introducing bias towards the test fold.

We perform all analyses mentioned in the previous subsections to this cross-validated model as well. We must note that we cannot be certain that the chosen model is the best cross-validated model of all, as we only performed this for a single model that was the best-performing one using the train-validation-test split technique. However, as it is never feasible to explore all possible architectures and due to computational constraints, finding the best model for certain is not possible.

## 4.9 Explainability

The downside of deep learning models is often that they are considered "black boxes". We have tried to make the models more explainable by looking at how certain a model is of its prediction. We also inspected the input features of the models, to see which input features contributed the most to the predictions.

### 4.9.1 Prediction certainty for traditional machine learning models

Calculating the prediction certainty for the traditional machine learning models was done via a bootstrapping technique and by looking at separate predictions.

For the bootstrapping technique, the age prediction of a single subject was performed 100 times. Normally, for the final prediction, the median is taken from the separate age predictions for all data points of the subject at a specific age. In the bootstrapped case, not all data points are used, but an equal number of data points as for the original prediction is sampled (with replacement) from the original data points. Predictions are made on these sampled data sets. This returns a distribution of predictions, of which the standard deviation was calculated. The mean standard deviation for all subjects is the model's prediction uncertainty.

For the technique with the separate predictions, instead of taking the median of all predictions for the final prediction, the standard deviation of the distribution of single predictions was calculated. The mean standard deviation for all subjects is the model's prediction uncertainty.

### 4.9.2 Prediction certainty for deep learning models

In the case of the deep learning techniques, most prediction certainty techniques like bootstrapping the data for all models were computationally unfeasible. However, we used another technique only for the best model (due to the same computational limitations), called Monte Carlo Dropout [81]. Normally, when a deep learning model has dropout layers, these are disabled during the process of making predictions. When using Monte Carlo Dropout, these dropout layers are enabled during the prediction phase. By doing this, neurons in the network are randomly switched off. If the input data is sufficiently similar to the training data, the model will likely have developed redundancies and this randomly switching off of neurons should not influence the final prediction a lot.

We have predicted the age of all subjects 50 times using Monte Carlo dropout, resulting in a distribution of predicted ages for each subject. Then, we calculated the standard deviations of these distributions. Finally, we were able to take the mean of all these standard deviations to get the prediction uncertainty of the model.

### 4.9.3 Weights inspection

To better understand the input features that contribute to the final prediction of the models, we have visualized the input features for the fully-connected neural network using the raw EEG data as input. This model was chosen as this network has a direct connection between all the input features (30 channels, 501 samples) and the first dense layer (500 nodes). This makes it

easier to directly look at the weights between these connections compared to a convolutional neural network that makes use of filters.

We looked at the weights of the trained model, which we first had to unflatten to get the original input shape of 30 channels and 501 samples per channel (500 Hz, 1 second). For each input feature, we have taken the mean weight value for the feature's connection to all the (500) nodes in the first dense layer. To assess the importance of a weight, its sign is not relevant, only its magnitude. Moreover, the sign could lead to misinterpretations of the weight, as a negative weight doesn't necessarily say something about the final prediction being lower or higher. Therefore, we calculated the absolute values of all of the weight values.

# 5 Results

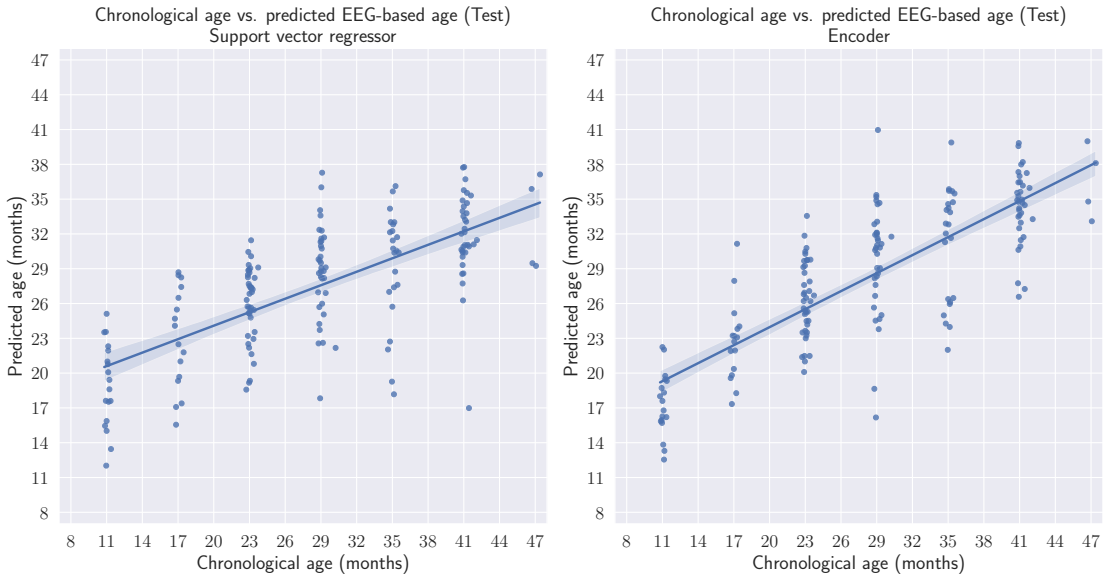
In this section, we present the results of the models we have developed for this thesis research. As discussed in the methods section, we performed parameter searches for most of the models. The results of only the best of each of the models are presented below. Firstly, we present the performance measures of the models. Furthermore, we will present the prediction error stability, vocabulary size compared to prediction error, and dyslexia compared to prediction error. Additionally, we try to make the models a little more explainable by looking at model prediction certainty and by visualizing the weights of the fully-connected neural network. Most results were calculated on both the test set and on the training set. The training set for the traditional machine learning approaches was larger than the deep learning approaches, as no separate validation set was used for these approaches. The results are presented in tables and graphs. The graphs are only shown for the best-performing model for the traditional machine learning and deep learning approaches, based on the lowest MAE. The graphs for all other models can be found in the supplementary materials section. Please refer to the methods section, specifically subsection 4.7, for a detailed explanation of how the results were gathered. Lastly, we will present the results of the best model that we have retrained by using cross-validation, enabling us to present results on the data set as a whole.

## 5.1 Performance measures

In this subsection, we present the performance of the models by using three measures: (1) the root mean squared error, (2) mean absolute error, and (3) R-squared. The graphs for the support vector regressor and Encoder model on the test set can be found in figures 9a and 9b, respectively. The results for the traditional machine learning approaches can be found in table 3 and the results for deep learning approaches can be found in table 4.

The subjects in the test set were the same for both deep learning and traditional machine learning approaches. We have also included two dummy regressors in the traditional machine learning results table, one that always predicts the mean and one that always predicts the median.

For the traditional machine learning approaches, the support vector regressor (SVR) shows the lowest mean absolute error on the test set. For the deep learning approaches, the Encoder model has the lowest mean absolute error on the test set. Both graphs plotting the chronological (true) age of the subjects against the predicted ages for these two best models show that the younger (e.g. 11 months) subjects' prediction tends to be higher than the chronological age, whereas the opposite is true for the older (e.g. 47 months) subjects.




(a) Support vector regressor

(b) Encoder model

Figure 9: The predictions of the support vector regressor and Encoder model compared to the true chronological ages of the subject on the test set.

| | Test | | | Train | | |
|---|---|---|---|---|---|---|
| Model | MAE | RMSE | $R^2$ | MAE | RMSE | $R^2$ |
| Random forest | 6.56 | 7.90 | 0.36 | 3.29 | 4.01 | 0.85 |
| Linear support vector regressor | 6.87 | 8.30 | 0.29 | 6.41 | 7.78 | 0.44 |
| Support vector regressor | 5.93 | 7.31 | 0.45 | 4.93 | 6.14 | 0.65 |
| Relevance vector regressor | 6.41 | 7.79 | 0.38 | 6.10 | 7.31 | 0.51 |
| Fully-connected neural network | 6.03 | 7.54 | 0.42 | 4.24 | 5.28 | 0.74 |
| Dummy regressor (mean) | 8.40 | 9.98 | -0.02 | 8.87 | 10.44 | -0.01 |
| Dummy regressor (median) | 8.13 | 9.90 | 0.00 | 8.68 | 10.46 | -0.01 |

Table 3: The performance measure results (MAE and RMSE in months) for the traditional machine learning approaches and dummy regressors.

| | Test | | | Train | | | Validation | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | MAE | RMSE | $R^2$ | MAE | RMSE | $R^2$ | MAE | RMSE | $R^2$ |
| Fully-connected neural network | 6.26 | 7.81 | 0.375 | 4.59 | 5.91 | 0.675 | 5.26 | 6.90 | 0.572 |
| Convolutional neural network | 5.71 | 7.10 | 0.484 | 4.52 | 5.67 | 0.702 | 5.04 | 6.59 | 0.610 |
| ResNet | 5.78 | 7.05 | 0.490 | 4.21 | 5.33 | 0.736 | 4.61 | 6.05 | 0.670 |
| Encoder | 5.06 | 6.15 | 0.613 | 3.85 | 4.77 | 0.788 | 4.60 | 5.87 | 0.689 |
| Time-CNN | 6.09 | 7.43 | 0.434 | 5.19 | 6.41 | 0.618 | 5.32 | 6.59 | 0.609 |
| InceptionTime | 5.63 | 7.04 | 0.492 | 4.25 | 5.41 | 0.728 | 4.68 | 6.06 | 0.669 |
| BLSTM-LSTM | 7.51 | 9.06 | 0.160 | 7.63 | 9.06 | 0.237 | 7.56 | 9.17 | 0.243 |

Table 4: The performance measure results (MAE and RMSE in months) for the deep learning models.

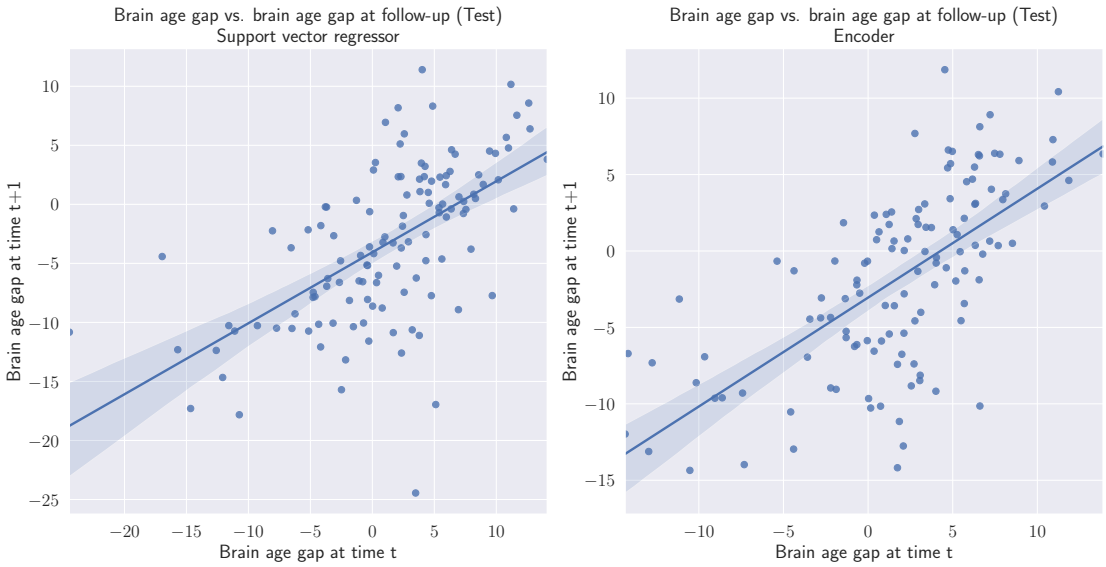
## 5.2 Error stability

In this subsection, we present the error stability of the different models. In figure 10 the brain age gaps compared to the follow-up measurements for the support vector regressor and Encoder model are shown. The results for the traditional machine learning approaches can be found in table 5 and the results for deep learning approaches can be found in table 6.

The error stability (or brain age gap) is expressed in the Pearson's correlation coefficient and measures the correlation between the prediction error at time $t$ compared to the follow-up measurement at the $t+1$. A (high) positive correlation indicates a stable error. All models show a relatively high positive correlation. All correlations are significant (alpha of 0.05).

| Model | Correlation (Test) | Correlation (Train) |
|---|---|---|
| Random forest | 0.660 | 0.749 |
| Linear support vector machine | 0.655 | 0.758 |
| Support vector machine | 0.602 | 0.716 |
| Relevance vector machine | 0.671 | 0.782 |
| Fully-connected neural network | 0.608 | 0.702 |

Table 5: The Pearson's correlation coefficient between the brain age gap (prediction error) at time $t$ compared to the follow-up at time $t+1$ for the traditional machine learning approaches.




(a) Support vector regressor

(b) Encoder model

Figure 10: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the support vector regressor and Encoder model on the test set.

| Model | Correlation (Test) | Correlation (Train) | Correlation (Validation) |
|---|---|---|---|
| Fully-connected neural network | 0.619 | 0.602 | 0.611 |
| Convolutional neural network | 0.534 | 0.551 | 0.480 |
| ResNet | 0.604 | 0.560 | 0.519 |
| Encoder | 0.663 | 0.561 | 0.553 |
| Time-CNN | 0.579 | 0.635 | 0.559 |
| InceptionTime | 0.590 | 0.544 | 0.505 |
| BLSTM-LSTM | 0.835 | 0.877 | 0.852 |

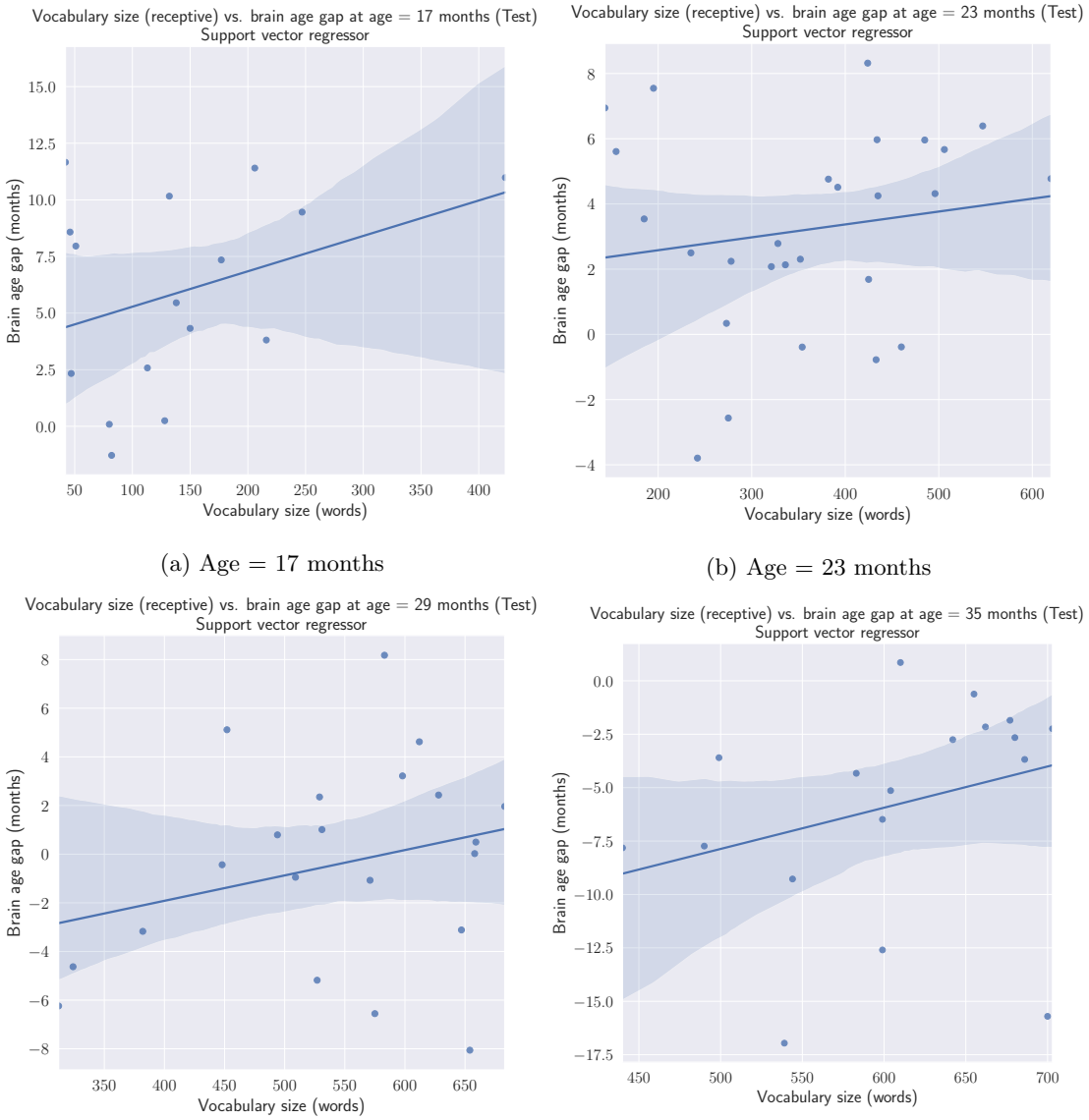
Table 6: The Pearson's correlation coefficient between the brain age gap (prediction error) at time $t$ compared to the follow-up at time $t+1$ for the deep learning approaches.

## 5.3 Vocabulary size

In this subsection, we compare the brain age gap (prediction error) with the vocabulary size of subjects at the ages of 17, 23, 29, and 35 months. The results for the traditional machine learning approaches can be found in tables 7 and 8, and the results for deep learning approaches can be found in tables 9 and 10. Figures 11, 12, 13, and 14 show the graphs for SVR and Encoder model on the test set.

We calculated the correlation between the vocabulary size (active/expressive and passive/receptive) and the brain age gap. It is important to note that we did not have vocabulary information on all subjects and for all ages. For the test set, we had vocabulary size data for 16 (out of a total of 16), 27 (39), 21 (35), and 18 (23) subjects for the ages of 17, 23, 29, and 35 respectively.

The training sets for the traditional machine learning models and deep learning models plus the fully-connected neural network of the traditional models have different sizes. The traditional machine learning models did not have a separate validation set (except the fully-connected neural network), the deep learning models used a validation set of 15%. For the traditional machine learning models training set we had vocabulary information of 70 (out of a total of 93), 147 (216), 134 (195), and 92 (137) subjects, respectively. For the deep learning models and fully-connected traditional model training set we had information of 57 (out of a total of 74), 118 (181), 107 (159), and 77 (114) subjects, respectively. For the validation set of the deep learning models, we had information of 13 (out of a total of 19), 29 (35), 27 (36), and 15 (23) subjects, respectively.



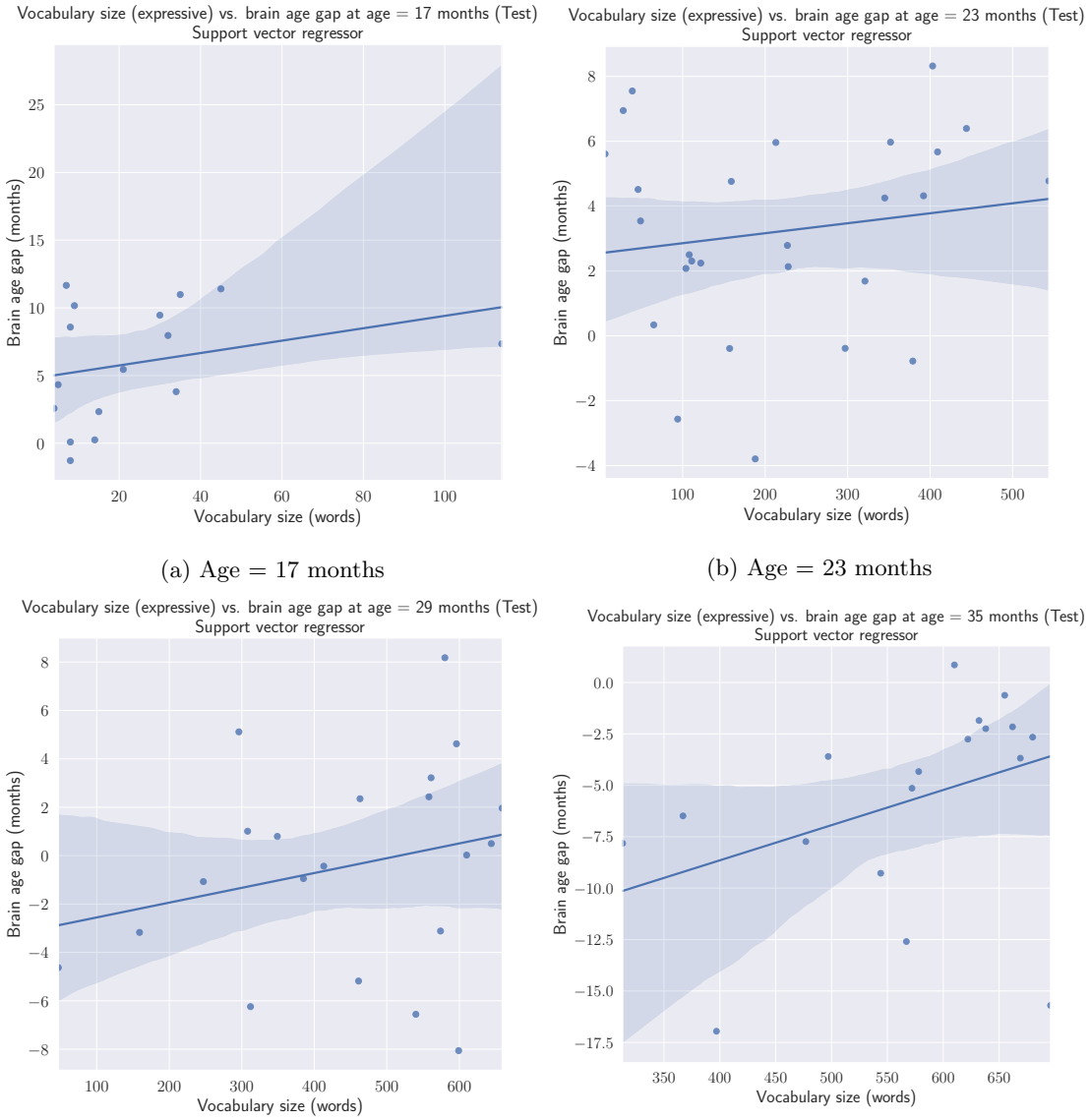

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 11: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the test set.



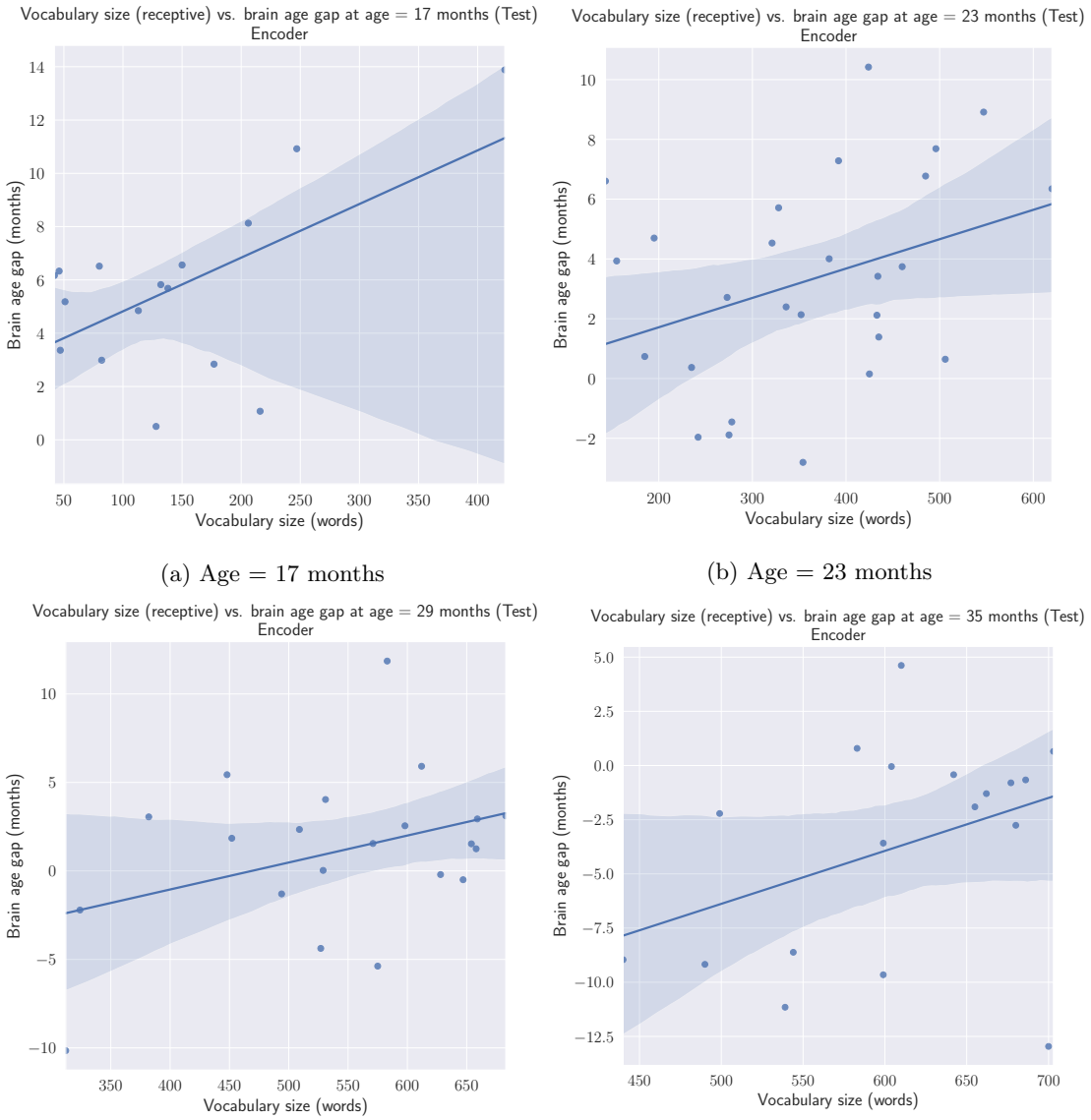

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 12: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the test set.



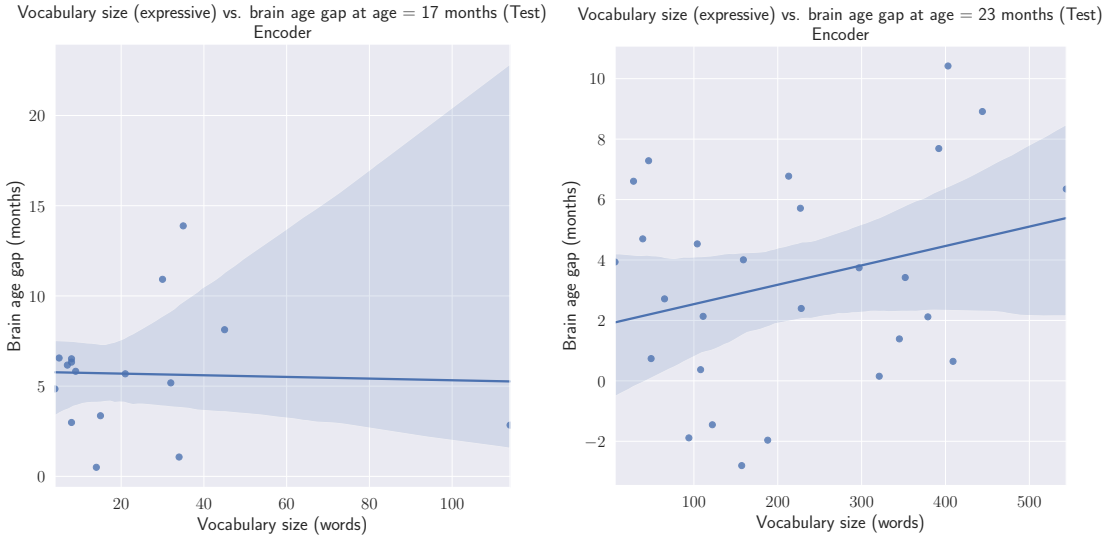

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

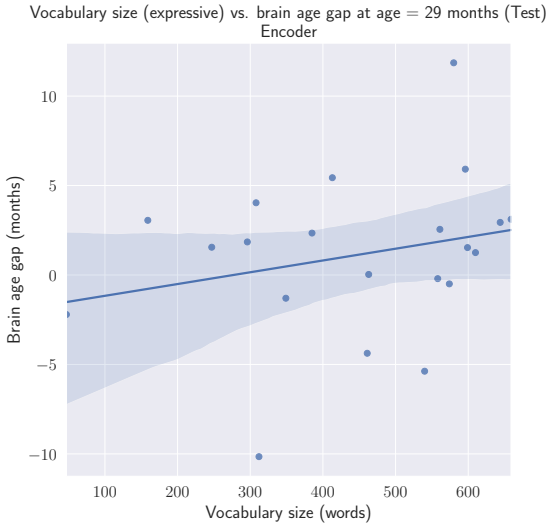
Figure 13: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the test set.



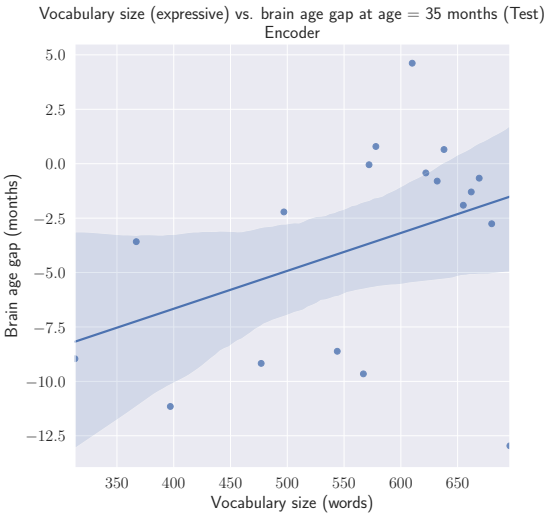

(a) Age = 17 months




(b) Age = 23 months




(c) Age = 29 months




(d) Age = 35 months

Figure 14: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the test set.

| Test | | | | |
|---|---|---|---|---|
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Random forest | 0.477 (0.062) | 0.169 (0.399) | 0.123 (0.595) | 0.278 (0.264) |
| Linear support vector machine | 0.292 (0.273) | 0.100 (0.618) | 0.198 (0.390) | 0.275 (0.269) |
| Support vector machine | 0.356 (0.176) | 0.159 (0.428) | 0.269 (0.238) | 0.298 (0.229) |
| Relevance vector machine | 0.433 (0.094) | 0.123 (0.540) | 0.203 (0.378) | 0.302 (0.224) |
| Fully-connected neural network | 0.431 (0.096) | 0.153 (0.445) | 0.197 (0.393) | 0.316 (0.202) |
| Train | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Random forest | 0.107 (0.376) | -0.007 (0.931) | 0.000 (0.998) | 0.197 (0.060) |
| Linear support vector machine | -0.016 (0.898) | -0.101 (0.223) | 0.047 (0.588) | 0.133 (0.205) |
| Support vector machine | 0.071 (0.559) | -0.004 (0.960) | 0.082 (0.348) | 0.172 (0.102) |
| Relevance vector machine | 0.065 (0.590) | -0.038 (0.644) | 0.073 (0.402) | 0.163 (0.121) |
| Fully-connected neural network | 0.047 (0.728) | 0.023 (0.808) | -0.076 (0.439) | 0.137 (0.236) |

Table 7: The Pearson's correlation coefficient (p-value, two-sided) between the brain age gap (prediction error) and the receptive vocabulary at the ages of 17, 23, 29, and 35 months for traditional machine learning approaches.

| Test | | | | |
|---|---|---|---|---|
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Random forest | 0.370 (0.158) | 0.164 (0.412) | 0.013 (0.957) | 0.334 (0.175) |
| Linear support vector machine | 0.368 (0.160) | 0.114 (0.570) | 0.133 (0.565) | 0.343 (0.164) |
| Support vector machine | 0.287 (0.281) | 0.152 (0.448) | 0.248 (0.279) | 0.384 (0.116) |
| Relevance vector machine | 0.317 (0.231) | 0.109 (0.588) | 0.153 (0.507) | 0.345 (0.160) |
| Fully-connected neural network | 0.197 (0.466) | 0.129 (0.522) | 0.159 (0.491) | 0.415 (0.087) |
| Train | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Random forest | 0.095 (0.433) | 0.078 (0.346) | 0.025 (0.775) | 0.209 (0.045) |
| Linear support vector machine | 0.010 (0.938) | 0.005 (0.953) | 0.030 (0.728) | 0.086 (0.416) |
| Support vector machine | 0.071 (0.560) | 0.084 (0.309) | 0.090. (0.301) | 0.152 (0.149) |
| Relevance vector machine | 0.068 (0.573) | 0.049 (0.555) | 0.070 (0.422) | 0.141 (0.179) |
| Fully-connected neural network | 0.008 (0.953) | 0.109 (0.240) | -0.055 (0.577) | 0.116 (0.313) |

Table 8: The Pearson's correlation coefficient (p-value, two-sided) between the brain age gap (prediction error) and the expressive vocabulary at the ages of 17, 23, 29, and 35 months for traditional machine learning approaches.

| Test | | | | |
|---|---|---|---|---|
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.564 (0.023) | 0.269 (0.175) | 0.171 (0.460) | 0.248 (0.320) |
| Convolutional neural network | 0.659 (0.005) | 0.326 (0.097) | 0.237 (0.300) | 0.330 (0.181) |
| ResNet | 0.542 (0.030) | 0.336 (0.087) | 0.255 (0.265) | 0.391 (0.108) |
| Encoder | 0.585 (0.017) | 0.349 (0.075) | 0.369 (0.100) | 0.383 (0.116) |
| Time-CNN | 0.548 (0.028) | 0.356 (0.068) | 0.210 (0.361) | 0.313 (0.206) |
| InceptionTime | 0.705 (0.002) | 0.343 (0.080) | 0.346 (0.124) | 0.350 (0.155) |
| BLSTM-LSTM | 0.201 (0.456) | 0.368 (0.059) | -0.058 (0.804) | 0.272 (0.275) |
| Train | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.146 (0.278) | 0.115 (0.216) | -0.017 (0.858) | 0.211 (0.066) |
| Convolutional neural network | 0.020 (0.883) | 0.104 (0.265) | -0.020 (0.835) | 0.140 (0.224) |
| ResNet | 0.019 (0.887) | 0.105 (0.256) | 0.006 (0.949) | 0.098 (0.396) |
| Encoder | 0.144 (0.285) | 0.083 (0.373) | 0.071 (0.467) | 0.260 (0.023) |
| Time-CNN | 0.084 (0.535) | 0.041 (0.660) | -0.053 (0.588) | 0.226 (0.048) |
| InceptionTime | 0.076 (0.575) | 0.032 (0.735) | -0.006 (0.947) | 0.205 (0.074) |
| BLSTM-LSTM | 0.039 (0.774) | 0.144 (0.119) | -0.047 (0.629) | 0.223 (0.052) |
| Validation | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.515 (0.072) | 0.336 (0.074) | 0.353 (0.071) | -0.118 (0.676) |
| Convolutional neural network | 0.343 (0.251) | 0.328 (0.082) | 0.409 (0.034) | 0.165 (0.558) |
| ResNet | 0.057 (0.852) | 0.260 (0.174) | 0.524 (0.005) | 0.108 (0.702) |
| Encoder | 0.054 (0.861) | 0.139 (0.472) | 0.342 (0.081) | 0.072 (0.798) |
| Time-CNN | 0.111 (0.717) | 0.322 (0.088) | 0.348 (0.075) | 0.121 (0.666) |
| InceptionTime | -0.019 (0.950) | 0.233 (0.224) | 0.488 (0.010) | 0.204 (0.466) |
| BLSTM-LSTM | -0.02 (0.949) | 0.421 (0.023) | 0.186 (0.353) | 0.261 (0.348) |

Table 9: The Pearson's correlation coefficient (p-value, two-sided) between the brain age gap (prediction error) and the receptive vocabulary at the ages of 17, 23, 29, and 35 months for deep learning approaches.

| Test | | | | |
|---|---|---|---|---|
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.208 (0.439) | 0.208 (0.297) | 0.083 (0.721) | 0.265 (0.287) |
| Convolutional neural network | 0.145 (0.593) | 0.251 (0.207) | 0.114 (0.621) | 0.334 (0.176) |
| ResNet | 0.086 (0.750) | 0.233 (0.242) | 0.184 (0.426) | 0.397 (0.103) |
| Encoder | -0.037 (0.892) | 0.280 (0.157) | 0.250 (0.274) | 0.396 (0.104) |
| Time-CNN | 0.066 (0.808) | 0.315 (0.109) | 0.154 (0.506) | 0.322 (0.193) |
| InceptionTime | 0.105 (0.700) | 0.300 (0.129) | 0.221 (0.335) | 0.403 (0.098) |
| BLSTM-LSTM | 0.259 (0.333) | 0.286 (0.148) | -0.086 (0.710) | 0.443 (0.065) |
| Train | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.132 (0.328) | 0.224 (0.015) | -0.012 (0.905) | 0.172 (0.136) |
| Convolutional neural network | -0.015 (0.910) | 0.204 (0.027) | 0.023 (0.816) | 0.098 (0.397) |
| ResNet | 0.039 (0.771) | 0.195 (0.034) | 0.026 (0.788) | 0.071 (0.541) |
| Encoder | 0.035 (0.797) | 0.177 (0.056) | 0.092 (0.344) | 0.209 (0.068) |
| Time-CNN | 0.150 (0.265) | 0.136 (0.141) | -0.029 (0.766) | 0.193 (0.092) |
| InceptionTime | 0.142 (0.291) | 0.147 (0.112) | 0.034 (0.725) | 0.165 (0.152) |
| BLSTM-LSTM | 0.043 (0.750) | 0.222 (0.016) | -0.051 (0.603) | 0.177 (0.124) |
| Validation | | | | |
| | Age in months | | | |
| Model | 17 | 23 | 29 | 35 |
| Fully-connected neural network | 0.414 (0.160) | 0.438 (0.017) | 0.180 (0.368) | 0.116 (0.680) |
| Convolutional neural network | -0.015 (0.962) | 0.420 (0.023) | 0.273 (0.168) | 0.173 (0.539) |
| ResNet | -0.089 (0.773) | 0.368 (0.049) | 0.410 (0.034) | 0.192 (0.494) |
| Encoder | 0.008 (0.980) | 0.303 (0.110) | 0.224 (0.260) | 0.228 (0.415) |
| Time-CNN | 0.153 (0.618) | 0.395 (0.034) | 0.236 (0.236) | 0.128 (0.650) |
| InceptionTime | -0.117 (0.704) | 0.352 (0.061) | 0.367 (0.059) | 0.167 (0.552) |
| BLSTM-LSTM | 0.016 (0.959) | 0.547 (0.002) | -0.044 (0.829) | 0.199 (0.476) |

Table 10: The Pearson's correlation coefficient (p-value, two-sided) between the brain age gap (prediction error) and the expressive vocabulary at the ages of 17, 23, 29, and 35 months for deep learning approaches.

## 5.4 Dyslexia

In this subsection, we compare the brain age gap (prediction error) with the dyslexia information we have on the subjects. The results for the traditional machine learning approaches can be found in tables 11 and 13, and the results for deep learning approaches can be found in tables 12 and 14.

Within each age group, the subjects are split into a dyslexic and non-dyslexic group. Then, a one-sided Kolmogorov-Smirnov test was performed to determine whether the dyslexic group's signed brain age gap distribution was significantly lower (alpha of 0.05) than the non-dyslexic group's brain age gap distribution. It is important to note that we did not have dyslexia information on all subjects.

For the test set, we had information of 41 subjects out of 46, of which 13 were diagnosed with dyslexia. Due to the small number of dyslexic subjects, we found none in the group with 47-month-old subjects. The training sets for the traditional machine learning models and deep learning models plus the fully-connected neural network of the traditional models have different sizes. The traditional machine learning models did not have a separate validation set (except the fully-connected neural network), the deep learning models used a validation set of 15%. For the traditional machine learning models training set we had dyslexia information of 236 subjects out of 258, of which 95 were diagnosed with dyslexia. For the deep learning models and fully-connected traditional model, we had dyslexia information of 194 subjects out of 212, of which 80 were diagnosed with dyslexia. For the validation set of the deep learning models, we had dyslexia information of 42 subjects out of 46, of which 15 were diagnosed with dyslexia.

| Test | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 0.0 (1.000) | 0.109 (0.833) | 0.042 (0.960) | 0.208 (0.456) | 0.467 (0.125) | 0.112 (0.809) | X |
| Linear support vector regressor | 0.0 (1.000) | 0.200 (0.673) | 0.083 (0.886) | 0.091 (0.846) | 0.300 (0.398) | 0.056 (0.913) | X |
| Support vector regressor | 0.0 (1.000) | 0.109 (0.833) | 0.125 (0.760) | 0.043 (0.946) | 0.333 (0.342) | 0.006 (0.967) | X |
| Relevance vector regressor | 0.0 (1.000) | 0.109 (0.833) | 0.125 (0.760) | 0.108 (0.757) | 0.367 (0.262) | 0.112 (0.809) | X |
| Fully-connected neural network | 0.0 (1.000) | 0.109 (0.833) | 0.083 (0.886) | 0.177 (0.560) | 0.433 (0.163) | 0.106 (0.817) | X |
| Train | | | | | | | |
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 0.130 (0.302) | 0.029 (0.948) | 0.096 (0.383) | 0.010 (0.986) | 0.094 (0.533) | 0.052 (0.740) | 0.500 (0.041) |
| Linear support vector regressor | 0.102 (0.469) | 0.029 (0.948) | 0.121 (0.227) | 0.000 (1.000) | 0.060 (0.756) | 0.031 (0.890) | 0.143 (0.729) |
| Support vector regressor | 0.012 (0.370) | 0.029 (0.936) | 0.116 (0.247) | 0.000 (1.000) | 0.055 (0.791) | 0.016 (0.963) | 0.214 (0.533) |
| Relevance vector regressor | 0.023 (0.947) | 0.029 (0.931) | 0.116 (0.247) | 0.017 (0.962) | 0.051 (0.817) | 0.018 (0.951) | 0.214 (0.533) |
| Fully-connected neural network | 0.081 (0.660) | 0.000 (1.000) | 0.150 (0.145) | 0.034 (0.894) | 0.069 (0.738) | 0.096 (0.453) | 0.300 (0.378) |

Table 11: The one-sided Kolmogorov-Smirnov test statistic (p-value) for the dyslexic group brain age gap distribution compared to the non-dyslexic group brain age gap distribution for the traditional machine learning approaches.

| Test | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Fully-connected neural network | 0.083 (0.923) | 0.545 (0.088) | 0.167 (0.627) | 0.052 (0.904) | 0.133 (0.823) | 0.000 (1.000) | X |
| Convolutional neural network | 0.083 (0.923) | 0.345 (0.362) | 0.083 (0.886) | 0.126 (0.723) | 0.267 (0.494) | 0.000 (1.000) | X |
| ResNet | 0.167 (0.813) | 0.182 (0.736) | 0.125 (0.760) | 0.147 (0.651) | 0.233 (0.547) | 0.062 (0.902) | X |
| Encoder | 0.250 (0.663) | 0.272 (0.502) | 0.208 (0.483) | 0.100 (0.784) | 0.133 (0.813) | 0.031 (0.956) | X |
| Time-CNN | 0.167 (0.813) | 0.364 (0.323) | 0.083 (0.886) | 0.074 (0.873) | 0.133 (0.813) | 0.000 (1.000) | X |
| InceptionTime | 0.083 (0.923) | 0.091 (0.896) | 0.167 (0.627) | 0.100 (0.784) | 0.200 (0.646) | 0.118 (0.797) | X |
| BLSTM-LSTM | 0.417 (0.327) | 0.364 (0.323) | 0.125 (0.760) | 0.195 (0.502) | 0.200 (0.646) | 0.043 (0.950) | X |
| Train | | | | | | | |
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Fully-connected neural network | 0.042 (0.881) | 0.074 (0.789) | 0.059 (0.720) | 0.088 (0.541) | 0.161 (0.229) | 0.005 (0.990) | 0.200 (0.622) |
| Convolutional neural network | 0.015 (0.970) | 0.074 (0.789) | 0.049 (0.791) | 0.011 (0.984) | 0.125 (0.402) | 0.019 (0.955) | 0.000 (1.000) |
| ResNet | 0.015 (0.979) | 0.074 (0.789) | 0.020 (0.952) | 0.034 (0.899) | 0.132 (0.363) | 0.044 (0.828) | 0.000 (1.000) |
| Encoder | 0.038 (0.893) | 0.111 (0.616) | 0.016 (0.964) | 0.034 (0.894) | 0.122 (0.421) | 0.046 (0.812) | 0.000 (1.000) |
| Time-CNN | 0.051 (0.831) | 0.050 (0.880) | 0.054 (0.754) | 0.021 (0.952) | 0.140 (0.325) | 0.050 (0.781) | 0.100 (0.875) |
| InceptionTime | 0.022 (0.957) | 0.074 (0.789) | 0.036 (0.869) | 0.017 (0.968) | 0.136 (0.342) | 0.013 (0.975) | 0.100 (0.875) |
| BLSTM-LSTM | 0.036 (0.904) | 0.024 (0.966) | 0.022 (0.937) | 0.079 (0.600) | 0.121 (0.424) | 0.078 (0.582) | 0.300 (0.378) |
| Validation | | | | | | | |
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Fully-connected neural network | 0.367 (0.184) | 0.200 (0.622) | 0.183 (0.540) | 0.032 (0.949) | 0.400 (0.210) | 0.090 (0.814) | 0.000 (1.000) |
| Convolutional neural network | 0.475 (0.074) | 0.250 (0.513) | 0.050 (0.936) | 0.119 (0.721) | 0.400 (0.210) | 0.098 (0.788) | 0.000 (1.000) |
| ResNet | 0.217 (0.524) | 0.350 (0.275) | 0.133 (0.700) | 0.000 (1.000) | 0.400 (0.210) | 0.090 (0.814) | 0.000 (1.000) |
| Encoder | 0.600 (0.017) | 0.300 (0.378) | 0.150 (0.666) | 0.000 (1.000) | 0.400 (0.210) | 0.038 (0.941) | 0.000 (1.000) |
| Time-CNN | 0.175 (0.637) | 0.450 (0.124) | 0.167 (0.603) | 0.008 (0.968) | 0.300 (0.420) | 0.079 (0.839) | 0.250 (0.800) |
| InceptionTime | 0.400 (0.154) | 0.200 (0.622) | 0.200 (0.498) | 0.079 (0.843) | 0.300 (0.420) | 0.090 (0.814) | 0.000 (1.000) |
| BLSTM-LSTM | 0.175 (0.637) | 0.150 (0.737) | 0.183 (0.540) | 0.024 (0.954) | 0.400 (0.210) | 0.120 (0.712) | 0.500 (0.600) |

Table 12: The one-sided Kolmogorov-Smirnov test statistic (p-value) for the dyslexic group brain age gap distribution compared to the non-dyslexic group brain age gap distribution for the deep learning approaches.

| Test | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | | | | | | | | |
| | 11 | | 17 | | 23 | | 29 | | 35 | | 41 | | 47 | |
| Model | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D |
| Random forest | 10.32 | 6.20 | 7.57 | 5.44 | 3.32 | 2.59 | -0.91 | -1.16 | -6.25 | -4.24 | -9.95 | -12.22 | X | -15.34 |
| Linear support vector regressor | 11.92 | 6.00 | 7.86 | 5.29 | 3.45 | 2.62 | -0.29 | -1.79 | -6.40 | -5.12 | -10.25 | -12.81 | X | -15.82 |
| Support vector regressor | 9.61 | 4.24 | 6.78 | 4.10 | 3.11 | 2.15 | 0.45 | -1.28 | -5.62 | -3.56 | -8.28 | -11.22 | X | 12.97 |
| Relevance vector regressor | 11.17 | 5.99 | 7.81 | 5.04 | 3.59 | 2.95 | 0.06 | -1.03 | -5.95 | -3.86 | -9.15 | -11.48 | X | -14.80 |
| Fully-connected neural network | 7.37 | 3.08 | 5.40 | 3.21 | 2.04 | 1.44 | -1.32 | -2.44 | -6.95 | -5.08 | -10.15 | -12.92 | X | -14.66 |
| Train | | | | | | | | | | | | | | |
| | Age in months | | | | | | | | | | | | | |
| | 11 | | 17 | | 23 | | 29 | | 35 | | 41 | | 47 | |
| Model | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D |
| Random forest | 4.75 | 4.55 | 3.78 | 2.88 | 2.11 | 2.19 | 0.31 | -0.38 | -2.16 | -2.20 | -5.07 | -5.31 | -9.39 | -7.90 |
| Linear support vector regressor | 9.28 | 8.47 | 6.51 | 4.56 | 3.45 | 3.79 | 0.11 | -1.18 | -4.57 | -4.69 | -9.69 | -10.57 | -16.38 | -17.92 |
| Support vector regressor | 6.53 | 6.07 | 4.74 | 3.74 | 2.65 | 2.99 | 0.50 | -0.83 | -3.14 | -3.29 | -7.43 | -8.33 | -13.93 | -14.18 |
| Relevance vector regressor | 9.19 | 8.64 | 6.49 | 5.16 | 3.43 | 3.79 | 0.41 | -0.99 | -4.05 | -4.20 | -8.89 | -9.83 | -15.38 | -16.16 |
| Fully-connected neural network | 4.53 | 4.63 | 3.98 | 1.78 | 1.87 | 2.12 | -0.59 | -1.42 | -3.68 | -3.82 | -7.54 | -7.65 | -11.93 | -11.37 |

Table 13: The mean brain age gaps for the non-dyslexic (ND) and dyslexic (D) subjects per age group for the traditional machine learning approaches.

| Test | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | | | | | | | | |
| | 11 | | 17 | | 23 | | 29 | | 35 | | 41 | | 47 | |
| Model | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D |
| Fully-connected neural network | 6.27 | 5.35 | 3.31 | 3.98 | 1.36 | -0.20 | -0.64 | -2.31 | -5.77 | -6.84 | -9.24 | -13.53 | X | -13.23 |
| Convolutional neural network | 5.90 | 4.33 | 3.94 | 3.46 | 1.51 | 0.50 | 0.04 | -1.35 | -5.45 | -5.37 | -7.85 | -10.62 | X | -14.28 |
| ResNet | 5.79 | 5.82 | 3.90 | 4.12 | 2.26 | 0.92 | 0.35 | -0.78 | -5.23 | -4.08 | -7.92 | -9.47 | X | -13.51 |
| Encoder | 6.26 | 6.10 | 5.88 | 5.23 | 3.22 | 3.11 | 1.57 | 0.06 | -3.05 | -3.73 | -5.70 | -7.77 | X | -11.49 |
| Time-CNN | 7.73 | 7.14 | 6.00 | 5.25 | 2.44 | 1.39 | 0.42 | -0.13 | -5.50 | -5.84 | -7.98 | -10.87 | X | -15.34 |
| InceptionTime | 5.68 | 4.41 | 4.42 | 2.41 | 2.06 | 1.07 | 0.07 | -1.34 | -4.76 | -4.80 | -7.38 | -9.49 | X | -18.12 |
| BLSTM-LSTM | 13.13 | 13.56 | 8.68 | 10.01 | 4.02 | 3.59 | -1.30 | -0.61 | -6.96 | -6.68 | -12.23 | -13.36 | X | -13.49 |
| Train | | | | | | | | | | | | | | |
| | Age in months | | | | | | | | | | | | | |
| | 11 | | 17 | | 23 | | 29 | | 35 | | 41 | | 47 | |
| Model | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D |
| Fully-connected neural network | 3.86 | 3.42 | 2.22 | 1.13 | 0.71 | 0.31 | -1.58 | -2.01 | -4.58 | -4.39 | -7.80 | -8.93 | -11.19 | -13.23 |
| Convolutional neural network | 4.13 | 3.00 | 1.57 | 0.75 | 1.73 | 1.32 | -0.42 | -1.27 | -3.47 | -3.23 | -7.27 | -7.90 | -10.45 | -14.28 |
| ResNet | 4.07 | 2.68 | 1.89 | 1.59 | 1.94 | 1.53 | -0.30 | -1.00 | -3.26 | -2.77 | -6.45 | -7.18 | -9.41 | -13.51 |
| Encoder | 3.72 | 3.20 | 2.97 | 1.89 | 2.98 | 2.47 | 1.06 | 0.20 | -1.85 | -1.80 | -5.61 | -6.34 | -8.97 | -11.49 |
| Time-CNN | 6.37 | 5.23 | 3.97 | 2.79 | 2.72 | 2.28 | -0.13 | -1.15 | -3.55 | -2.99 | -7.65 | -8.32 | -10.94 | -15.34 |
| InceptionTime | 3.78 | 3.04 | 1.81 | 0.30 | 1.53 | 1.11 | -0.29 | -1.69 | -3.08 | -2.88 | -6.66 | -7.41 | -10.06 | -13.49 |
| BLSTM-LSTM | 13.39 | 12.82 | 9.11 | 8.56 | 4.30 | 3.85 | -1.08 | -1.30 | -6.28 | -5.94 | -11.92 | -12.08 | -17.06 | -18.12 |
| Validation | | | | | | | | | | | | | | |
| | Age in months | | | | | | | | | | | | | |
| | 11 | | 17 | | 23 | | 29 | | 35 | | 41 | | 47 | |
| Model | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D | ND | D |
| Fully-connected neural network | 4.65 | 4.91 | 1.86 | 0.95 | 0.80 | -0.00 | -0.11 | -1.73 | -2.93 | -1.88 | -9.68 | -12.05 | -11.15 | -15.54 |
| Convolutional neural network | 4.15 | 5.78 | 1.58 | 1.39 | 1.09 | -0.34 | 1.97 | -0.01 | -1.32 | -0.30 | -8.02 | -10.12 | -9.79 | -15.01 |
| ResNet | 4.21 | 4.34 | 1.07 | 1.92 | 1.67 | 0.11 | 2.47 | -0.18 | -1.05 | 0.43 | -6.85 | -9.24 | -6.71 | -14.82 |
| Encoder | 3.71 | 5.88 | 2.03 | 4.21 | 2.34 | 2.04 | 2.67 | 1.39 | -0.39 | 0.07 | -6.22 | -8.48 | -7.20 | -13.12 |
| Time-CNN | 6.36 | 5.70 | 1.35 | 3.15 | 1.99 | 1.01 | 2.67 | 0.44 | -1.09 | 0.03 | -8.35 | -9.97 | -10.81 | -14.14 |
| InceptionTime | 3.51 | 4.19 | 1.40 | 1.78 | 0.52 | 0.29 | 2.26 | 0.79 | -0.50 | -0.17 | -6.41 | -9.07 | -7.74 | -14.37 |
| BLSTM-LSTM | 13.17 | 12.93 | 8.57 | 8.09 | 3.58 | 3.50 | -0.12 | -1.01 | -6.21 | -4.88 | -12.76 | -13.03 | -17.87 | -16.71 |

Table 14: The mean brain age gaps for the non-dyslexic and dyslexic subjects per age group for the deep learning approaches.

## 5.5 Cross-validated best model

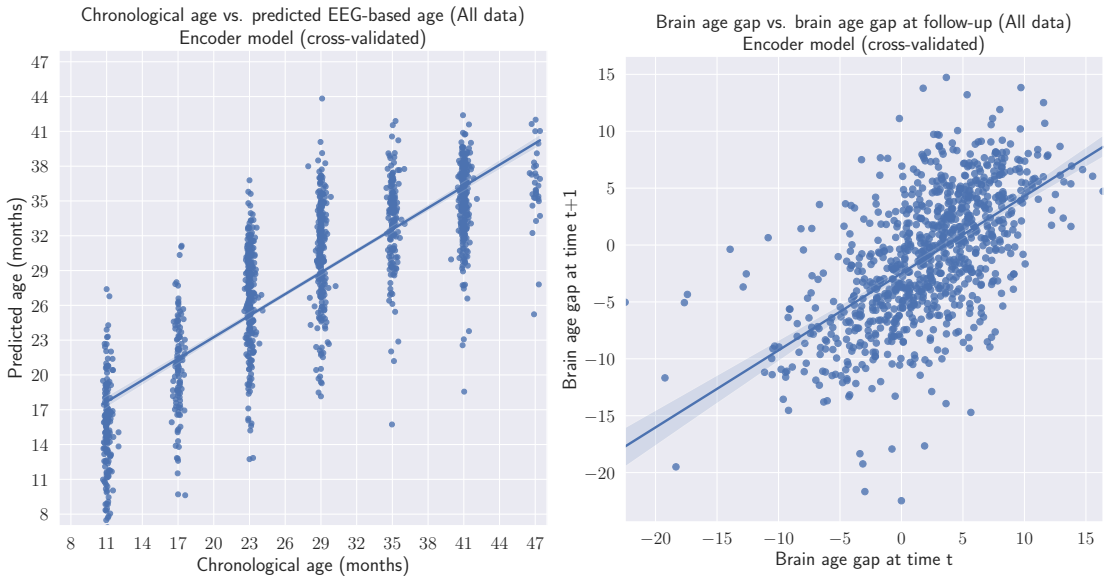
In this subsection, we will present the results of the best model that we have retrained using 7-fold cross-validation. The graph showing the brain age prediction compared to the chronological age can be found in figure 15a. The graph plotting the brain age gaps compared to the brain age gap at follow-up can be found in figure 15b.

As can be seen in the previous subsections in this results section, the Encoder model is the best performing model. Therefore, we have retrained this model, so we can present the same results on the data set as a whole.

The Encoder model (cross-validated) has an MAE of 4.82 months, an RMSE of 5.90 months, and an R-squared of 0.674 on the full data set. The Pearson's correlation coefficient between the brain age gap at time $t$ and the brain age gap at time $t + 1$ (error stability) is 0.597.

The Pearson's correlations (and p-values) between the receptive vocabulary at the ages of 17, 23, 29, and 35 months and the brain age gap are 0.198 (0.067), 0.148 (0.052), 0.138 (0.086), and 0.219 (0.021), respectively. The correlations between expressive vocabulary at the ages of 17, 23, 29, and 35 months and the brain age gaps are 0.056 (0.610), 0.198 (0.009), 0.124 (0.123), and 0.213 (0.026), respectively. The graphs comparing the receptive and expressive vocabulary sizes with the brain age gaps can be found in figures 16 and 17.

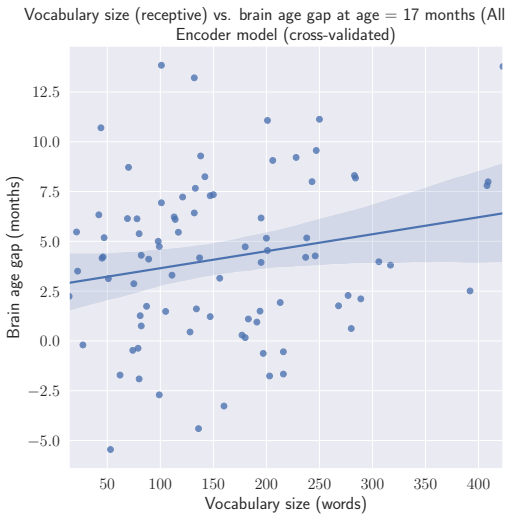
The one-sided Kolmogorov-Smirnov test statistic (and p-value) for the dyslexic brain age gap distribution compared to the non-dyslexic group brain age gap for the Encoder model (cross-validated) is 0.109 (0.404), 0.046 (0.870), 0.038 (0.820), 0.008 (0.990), 0.144 (0.203), 0.004 (0.993), and 0.059 (0.921) for the ages of 11, 17, 23, 29, 35, 41, and 47 months, respectively. The mean brain age gaps for the non-dyslexic group are 4.45, 3.94, 3.79, 2.13, $-1.57$, $-5.61$, and $-9.95$, respectively. The mean brain age gaps for the dyslexic group are 4.59, 3.08, 3.34, 0.86, $-1.15$, $-6.50$, and $-11.66$, respectively



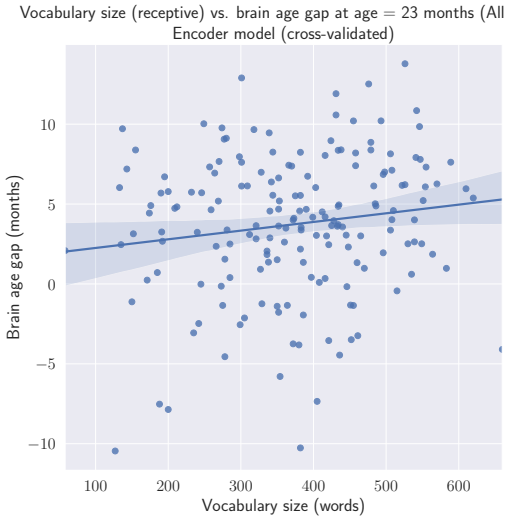

(a) Predicted age vs. chronological age

(b) Error stability

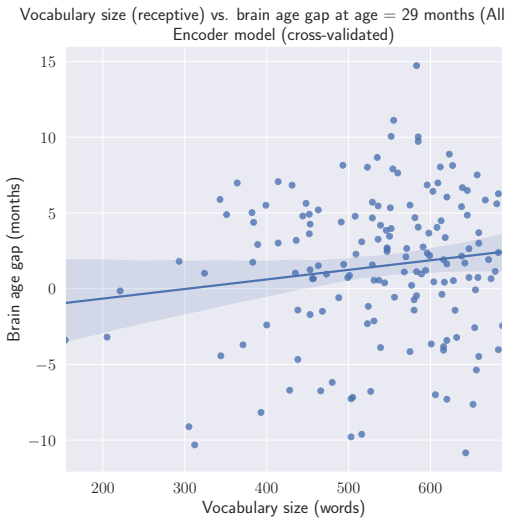
Figure 15: The predictions compared to the true chronological ages of the subject (left), and the brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ (right).



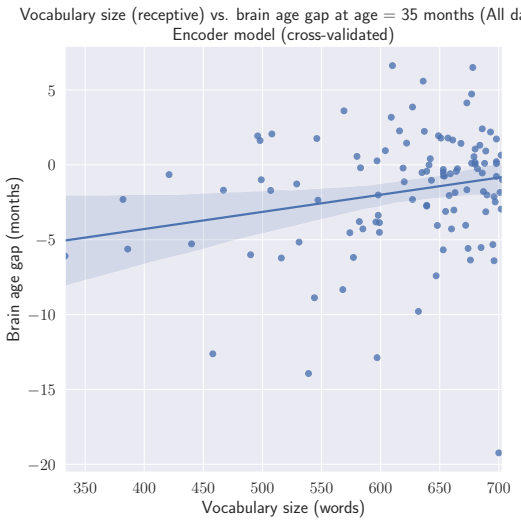

(a) Age = 17 months




(b) Age = 23 months

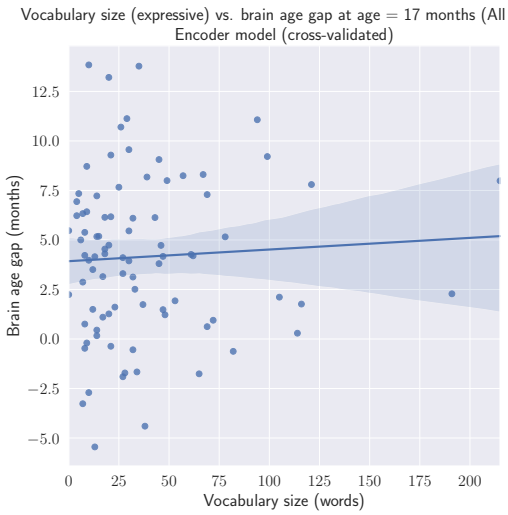



(c) Age = 29 months

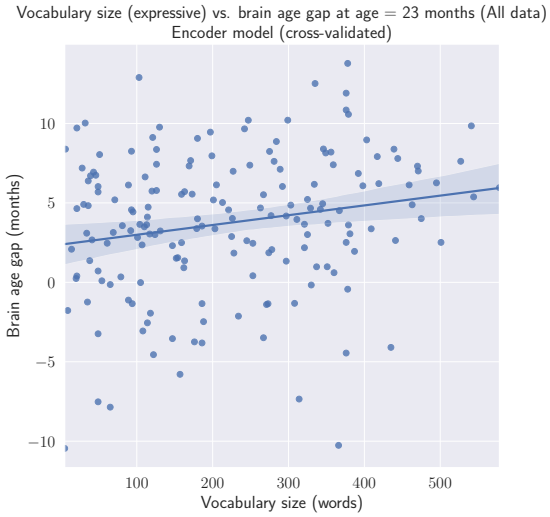



(d) Age = 35 months

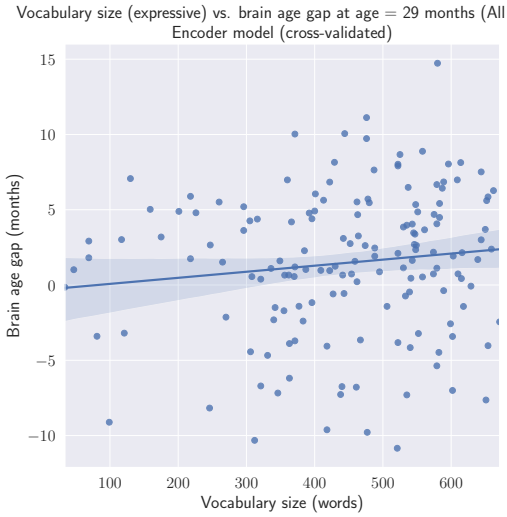
Figure 16: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the cross-validated Encoder model's brain age predictions on the full data set.



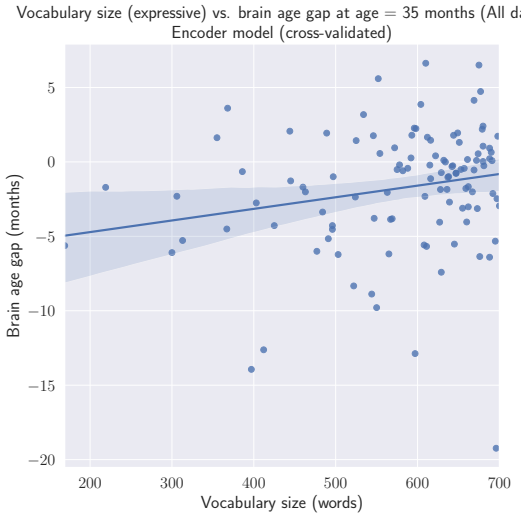

(a) Age = 17 months




(b) Age = 23 months




(c) Age = 29 months




(d) Age = 35 months

Figure 17: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the cross-validated Encoder model's brain age predictions on the full data set.

## 5.6 Explainability

In this subsection, we will present the data we have gathered regarding the explainability of the models. The traditional machine learning models' prediction uncertainty can be found in tables 15 and 16. For the best deep learning model, the Encoder model, the standard deviations of the predictions we have found by using the Monte Carlo Dropout approach can be found in table 17. Please note that the results for the traditional machine learning models and the deep learning model are not comparable, as different approaches have been used.

Furthermore, the visualization of the absolute mean weights of the trained deep learning fully-connected neural network between the input features and the first dense layer can be found in figure 18. The visualizations of the mean weights for the channels over all the time steps can be found in figure 19. In the supplementary materials, we have included the other figures for the smaller time step ranges. Lastly, the visualization of the mean weights for the time steps over all channels can be found in figure 20.

The first visualization shows the absolute weight values for each 2-millisecond time step (501 in total) per channel. In this visualization, the weights are colored and the higher weights are lighter and therefore this feature appears to be more important.

A second visualization translates the first figure to a 2D figure, showing the mean absolute weights over all time steps for each channel. This shows which channels are most important across the whole recording (0-1000ms) for getting a prediction. We have also divided the recording into 5 parts of 200ms, to see which channels are most important in these stages of the recording.

Lastly, we translated the first figure to a 2D figure again by making a visualization in which we took the mean absolute weight over all channels for each time step. This shows which time ranges in the recording are most important for making a prediction. We have added a rolling average of 20 milliseconds to be able to see the pattern better.

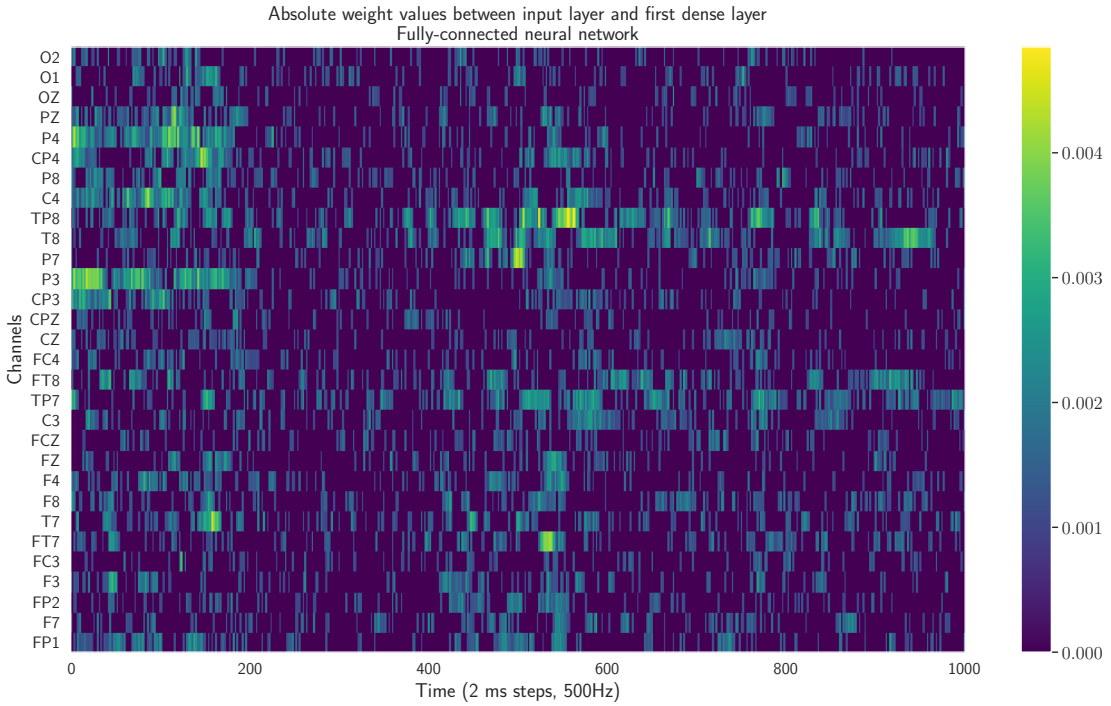



Figure 18: The mean absolute weight for each connection between the input features and the first dense layer of the trained fully-connected neural network. Clipping has been applied, weights below 0.001 are set to 0 for a clearer image.

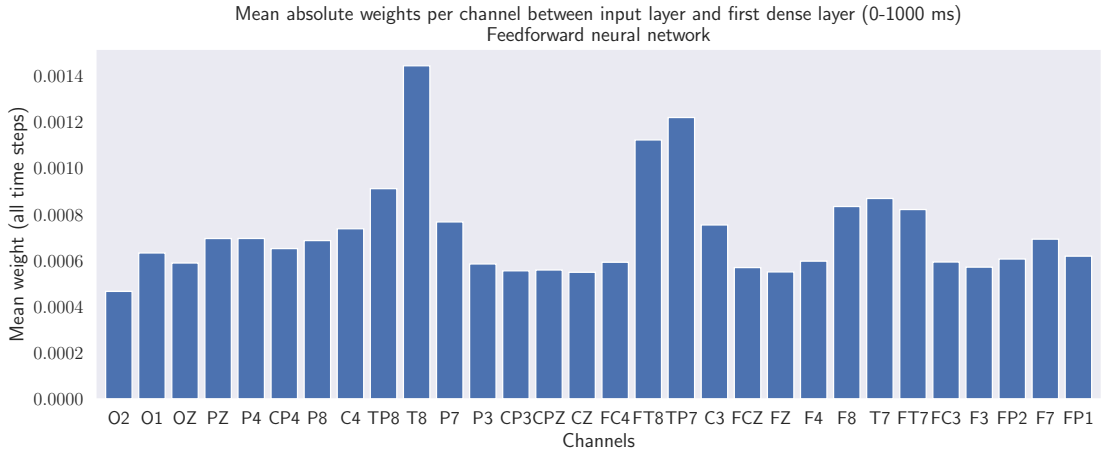



Figure 19: The mean absolute weight per channel (average across all time steps) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

| Test | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 0.135 | 0.151 | 0.151 | 0.129 | 0.121 | 0.131 | 0.129 |
| Linear support vector machine | 0.167 | 0.175 | 0.160 | 0.141 | 0.131 | 0.146 | 0.147 |
| Support vector machine | 0.179 | 0.180 | 0.188 | 0.184 | 0.165 | 0.188 | 0.205 |
| Relevance vector machine | 0.161 | 0.157 | 0.166 | 0.153 | 0.141 | 0.161 | 0.176 |
| Fully-connected neural network | 0.182 | 0.194 | 0.183 | 0.160 | 0.158 | 0.147 | 0.119 |
| Train | | | | | | | |
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 0.092 | 0.094 | 0.085 | 0.075 | 0.071 | 0.090 | 0.120 |
| Linear support vector machine | 0.167 | 0.180 | 0.157 | 0.151 | 0.129 | 0.143 | 0.144 |
| Support vector machine | 0.174 | 0.192 | 0.182 | 0.175 | 0.165 | 0.179 | 0.187 |
| Relevance vector machine | 0.150 | 0.159 | 0.168 | 0.156 | 0.151 | 0.157 | 0.158 |
| Fully-connected neural network | 0.143 | 0.189 | 0.162 | 0.153 | 0.137 | 0.140 | 0.204 |

Table 15: The standard deviation in months for the traditional machine learning models using the bootstrapping approach on the test and train sets.

| Test | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 3.334 | 3.760 | 3.779 | 3.397 | 3.361 | 3.243 | 3.414 |
| Linear support vector machine | 5.529 | 4.789 | 4.705 | 4.152 | 4.372 | 4.428 | 3.788 |
| Support vector machine | 4.935 | 4.852 | 4.859 | 4.661 | 4.659 | 4.678 | 4.667 |
| Relevance vector machine | 4.511 | 4.339 | 4.369 | 4.191 | 4.127 | 4.193 | 4.189 |
| Fully-connected neural network | 4.470 | 4.821 | 4.648 | 4.307 | 4.372 | 3.955 | 3.678 |
| Train | | | | | | | |
| | Age in months | | | | | | |
| Model | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Random forest | 2.544 | 2.376 | 2.103 | 1.989 | 1.993 | 2.276 | 2.867 |
| Linear support vector machine | 5.581 | 5.353 | 4.454 | 4.263 | 4.023 | 3.952 | 3.813 |
| Support vector machine | 4.776 | 4.597 | 4.472 | 4.369 | 4.377 | 4.472 | 4.549 |
| Relevance vector machine | 4.282 | 4.142 | 4.093 | 3.975 | 3.963 | 3.939 | 3.751 |
| Fully-connected neural network | 4.050 | 4.404 | 3.934 | 3.750 | 3.698 | 3.707 | 4.811 |

Table 16: The standard deviation of separate predictions in months for the traditional machine learning models on the test and train sets.

| | Age in months | | | | | | |
|---|---|---|---|---|---|---|---|
| Data set | 11 | 17 | 23 | 29 | 35 | 41 | 47 |
| Test | 0.412 | 0.398 | 0.383 | 0.387 | 0.352 | 0.357 | 0.425 |
| Validation | 0.352 | 0.413 | 0.420 | 0.370 | 0.349 | 0.374 | 0.367 |
| Train | 0.385 | 0.436 | 0.410 | 0.393 | 0.369 | 0.371 | 0.418 |

Table 17: The Monte Carlo Dropout standard deviation for the Encoder model on the train, validation, and test sets for all the age groups.

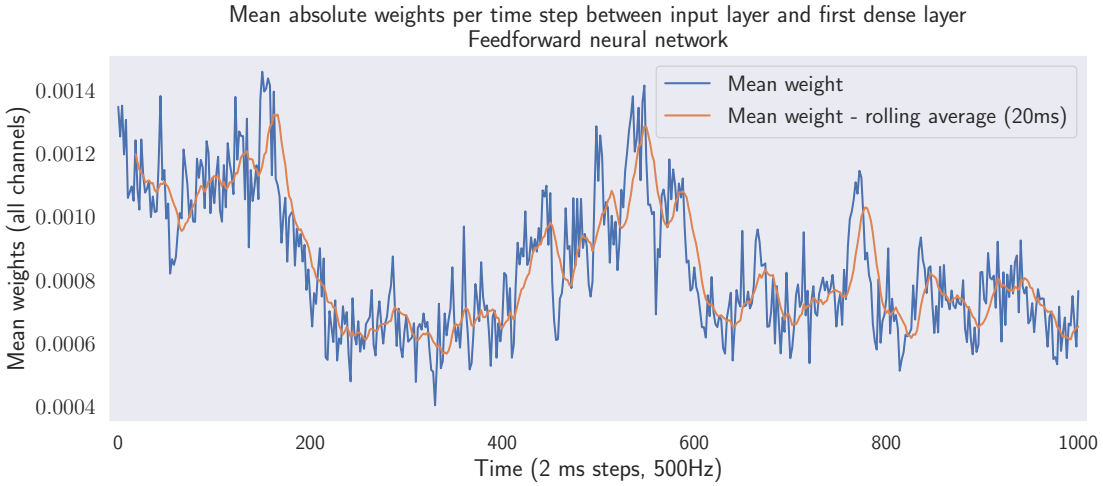



Figure 20: The mean absolute weight per time step (average across all channels) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

# 6 Discussion

In this section, we will discuss the results using the same subsection structure of the results section. The focus will be on the test set results, as the results on the train and validation sets are biased. Also, we will reflect on the aims we have formulated in the introduction. Furthermore, we will attempt to provide recommendations for future research.

## 6.1 Performance measures

In the introduction of this thesis, we observed that there is no literature on predicting (developmental) age in infants and children with deep learning regression using EEG data. One of the aims we have formulated is that we would like to determine how we can best use deep learning techniques for this prediction task and how this compares to traditional machine learning techniques.

Finding the absolute *best* deep learning model is an impossible task, as deep learning models can always be improved by tweaking the parameters a little more, but also because the search space of potential models is too large. Adding or removing layers, using different activation functions, adding dropout, there are plenty of options for improving the model. Therefore, we have decided to explore the best-performing models from Fawaz and colleagues' deep learning review studies [64][63], as they searched for the best-performing models on time series data.

Additionally, we added the BLSTM-LSTM model from Kaushik and colleagues [11], as they have specifically applied their model age group classification using EEG data.

To be able to determine whether the deep learning model's performance is good or not, we have implemented several traditional machine learning models as baseline models. Table 3 shows the performance measures of these traditional machine learning models. When looking at the MAE and RMSE on the test set we observe that all models perform better than simple dummy regressors, meaning that the models were able to pick up useful information from the features extracted out of the data set.

It can be seen that performance differs a lot between models, but the support vector regressor (SVR) does best with an MAE of 5.93 months, an RMSE of 7.31 months, and an R-squared of 0.45 on the test set. It should be noted that making the predictions on the training and test sets by the SVR took considerably longer than the other models, multiple hours compared to a few minutes for all other models on the machine used for model validation.

It can also be observed that the fully-connected neural network, which is the same as the one used for the deep learning models but trained on preprocessed data instead of raw EEG data (see subsection 4.4), is performing almost on par with the SVR.

Table 4 shows the performance of measures of the deep learning models. The first thing that can be seen is that the performance of the deep learning models, except for the BLSTM-LSTM, is approximately at least on par with the best traditional machine learning model, the SVR. The fully-connected neural network and the Time-CNN have a little higher MAE and RMSE and R-squared, but the differences are too small to conclude that they are performing worse than the SVR. However, when looking at the deep learning models overall and the traditional machine learning models overall, the deep learning models seem the have better performance.

The Encoder model, with an MAE of 5.06 months, RMSE of 6.15 months, and an R-squared of 0.613 on the test set, is the best performing model overall. Its performance seems to be considerably better than the other models, such as the second-best model, the InceptionTime model with an MAE of 5.63 months, RMSE of 7.04 months, and an R-squared of 0.492.

An observation we were able to make by looking at the graphs for the SVR and Encoder model (figure 9) comparing the chronological age of the subjects to the predicted age of the subjects, is that there is a bias in the predictions due to a negative correlation between the chronological age and the brain age gap. This results in a systematic over-prediction of the younger subjects' developmental age and an under-prediction of the older subjects' developmental age. This bias is observed in many brain age estimation studies, suggestions on why this is the case can be found in several studies [62][82]. One of the reasons might be an imbalance in the age of the samples used to construct the models, which could be resolved (partially) by oversampling younger and

older subjects during the training phase. There are several methods to adjust the models for the bias, of which a few are discussed by Treder and colleagues [83], which can be an interesting improvement for follow-up research.

One of the aims of this thesis is to determine which preprocessing approach is best. For the traditional models, preprocessing was performed, features were extracted and selected from the EEG data. For the deep learning models, simple preprocessing was performed (e.g. remove broken channels and interpolate), but the used features were just raw EEG data. An interesting comparison is the performances of the fully-connected neural networks (table 3 and 4), as these models are trained on both data preprocessing approaches. The performances do not differ so much to be able to conclude that one is better than the other. We can see however that the performances of the feature selection approach and raw data approach are similar for these two models, and the raw data approach has the advantage that it is simpler. The raw data approach also requires less domain knowledge, as it is not necessary to research which features can be extracted from EEG data. Also, the raw data approach enabled us the easily use data augmentation by adding Gaussian noise to the raw data. As a follow-up, it would be interesting to apply more deep learning models to the extracted features to make a more comprehensive comparison.

It is important to realize that the models are trained using the true age of the subjects as a proxy for the developmental age or brain age. Therefore, when looking at the performance measures, it is not possible to say when we have reached the point of perfect developmental age prediction. If the MAE would be zero, we are not predicting the developmental age anymore, but the true age. This means that the MAE should always be higher than zero for a model that is trying to predict the developmental age. The model's underestimation or overestimation compared to the true age should say something about the development of the subject. We verify this in the following sections.

## 6.2 Error stability

As mentioned before in section 4.7 and by the authors of [5], if our models were to predict the developmental age of the subject, we expect that the brain age gap (prediction error) of a subject is relatively stable over consecutive measurements. The assumption is that a subject with, for example, a delay in the development of a few months at 17 months, is not expected to have an advanced development with a few months at the next measurement. It should be noted that this does not mean a single subject cannot go from a delayed to an advanced development between two consecutive measurements, but for the population as a whole and multiple measurements, this appears to be a reasonable assumption.

It is also important to realize that we expect it to be true that the brain age gap is relatively stable when the models actually predict developmental age. However, the opposite is not necessarily true, a stable brain age gap does not imply the model actually predicts developmental age.

The brain age gap stability is expressed by the Pearson's correlation coefficient and can be found in tables 5 and 6. Figure 10 shows the graphs for the SVR and Encoder model on the test set. In these tables is shown that for all models the correlation is positive, ranging from a correlation of 0.534 to 0.835 on the test set. All correlations were significant (alpha of 0.05). These moderately to highly positive correlations confirm the expectation that the brain age gap should be stable over time and the overestimation and underestimation of the developmental age are systematic.

## 6.3 Vocabulary size

Another means to validate the model's ability to actually predict differences in subjects' development is by looking at another proxy for development. For many subjects, we have information on their receptive and expressive vocabulary sizes at the ages of 17, 23, 29, and 35 months and we computed the correlation between this and the brain age gap of the subject at that age. The results can be found in tables 7, 8, 9, and 10. Figures 11, 12, 13, and 14 show the graphs for SVR and Encoder models on the test set.

The results for the traditional machine learning models are all not significant (alpha of 0.05), except for the correlation for the expressive vocabulary at 35 months using the random forest model. However, this might be due to the multiple testing problem and the evidence is too little to conclude from this. Even though the results are not significant and therefore we cannot conclude that there is a correlation between the vocabulary size and the predictions of the traditional machine learning models, it can be observed that all the correlations on the test set are weakly to moderately positive. The fact that all correlations have this property, might indicate that the non-significance of the results is due to the sample size. Therefore, further research with a larger sample size might show different results.

The deep learning model results, however, show a significant correlation (alpha of 0.05) for six out of the seven models between the receptive vocabulary at 17 months and the brain age gap prediction, only the BLSTM-LSTM is not significant. The other correlations range from 0.548 to 0.705, moderately positive to strongly positive. For expressive vocabulary and other ages, the correlations are not significant. For these results, it is also interesting to see that most correlations are positive as well, and this might also be not significant due to the sample size. Further research would be necessary.

The significant positive correlations between the brain age prediction and the receptive vocabulary at 17 months, (might) indicate that our developmental age predictions actually can explain a little about the subject's development at a young age. A lower developmental age prediction at the age of 17 months by one of our deep learning models (except the BLSTM-LSTM), could indicate that the subject also has a lower receptive vocabulary at that age.

A caveat here is that the receptive vocabulary of a subject is defined as the number of words the subject *understands*, whereas the expressive vocabulary is the number of words a subject *understands and says*. These counts are determined via questionnaires filled out by the subject's parents. The receptive vocabulary is more subjective, as it is harder to determine whether a subject understands something without being able to say this word as well. Therefore, it is plausible that other factors, for example, parents with a higher education level reporting higher vocabulary sizes because they believe their child understands more than it actually does, are also influencing the statistics.

## 6.4 Dyslexia

The last method we have used to assess our models is by looking at dyslexia predisposition in the subject at a later age. If dyslexia is considered a condition related to the subject's development, there could be a relation between the brain age gap and dyslexia predisposition. We looked at the distributions of brain age gaps for subjects with and without recorded dyslexia at different ages. If the distributions differ significantly, this is supporting evidence that the models can differentiate children with higher and lower development. Furthermore, if the difference is very large, detecting dyslexia would be possible by using our models. The results for the traditional machine learning approaches can be found in tables 11 and 13, and the results for deep learning approaches can be found in tables 12 and 14.

The statistical tests we have performed on the test set to determine whether the distributions of brain age gaps of dyslexic and non-dyslexic subjects differ are not significant (alpha of 0.05). This means there is not enough evidence to conclude that the distributions of the brain age gaps of children with recorded dyslexia are lower than children without recorded dyslexia. This means that our models are not able to differentiate children with and without dyslexia based on their developmental age prediction.

We have also calculated the mean brain age gap predictions for the dyslexic and non-dyslexic groups at different ages. Interestingly, for most models and most age groups, the brain age gap of the non-dyslexic group is higher than the dyslexic group. This suggests there might be some dyslexia information present in the EEG data recordings. However, the differences are small and the information tells something on a group level, but on a subject basis, it is not possible to draw

any conclusions from the developmental age predictions. Future research could attempt to find the relationship between EEG data and dyslexia predisposition.

## 6.5 Cross-validated best model

To obtain unbiased results on the data set as a whole, we have retrained the best-performing model (Encoder) using a cross-validation approach. The performance on the full data set is even a bit higher than it was on the test set. This is likely due to the composition of the folds used. Like the models that were only validated on the test set, the error is stable.

The correlations between the predicted brain age gaps and the (receptive and expressive) vocabulary sizes are weakly positive. For the oldest age group, 35 months, this relationship is significant (alpha of 0.05) for both expressive and receptive vocabulary. The relationship between the brain age gap and the expressive vocabulary at the age of 23 months is significant as well. The other correlations are not significant with an alpha level of 0.05. Interestingly, when looking at the difference between the 'regular' Encoder model (trained on the train set, validated on the test set) and this cross-validated Encoder model, most correlations are less positive, but the p-values are lower as well. As the cross-validated Encoder model predictions contain more subjects, there is more evidence to draw conclusions, and further research with more subjects would be interesting.

Lastly, as with the regular deep learning models, the difference between the distributions of brain age gaps of the dyslexic and non-dyslexic groups is not significant. However, we observe the same difference in the mean brain age gaps between the groups, where the dyslexic group often has a slightly lower predicted age than the non-dyslexic group.

## 6.6 Explainability

A first attempt to better understand the models was made by taking a look at the prediction certainty of the models. The results can be found in tables 15, 16, and 17. For the traditional machine learning models, this was calculated by looking at the standard deviation using a bootstrapping technique and by looking at the standard deviation of predictions for single EEG epochs. The bootstrapping technique shows that deviations in the EEG data used to calculate the final prediction do not have a large impact on the final predictions of the models. This means that the final predictions are relatively stable.

When looking at the predictions on separate EEG epochs and the standard deviations for these predictions, it can be seen that the standard deviation is a lot higher. This means that the models are not very certain about the developmental age prediction for single EEG epochs.

Given the high variability between epochs and the generally low signal-to-noise ratio, this was expected. This is not necessarily a problem as the final predictions are made by taking the median of many predictions. Also, it is common in EEG/ERP research to record many epochs of data. Models do not have to rely on single EEG epochs for making predictions. For both the bootstrapping and the separate epoch approach, the random forest model has the lowest standard deviation, which also happens to be a property (low variability) of random forests.

Unfortunately, due to computational limitations, it was only possible to calculate the Monte Carlo Dropout-based standard deviation for the best deep learning model, the Encoder model. This makes it less informative, but it can still be observed that randomly disabling nodes in the network does not have a high impact on the predictions made on the final predictions. By disabling these nodes, parts of the EEG epochs' information are not available to the model anymore. The low standard deviations mean that model does not seem to rely on specific parts of the EEG epoch/features. If it were, the predictions would be (more) incorrect and a higher standard deviation is expected.

Another means to better understand the deep learning models was by a visual inspection of the weights of the fully-connected neural network. We have chosen this model instead of the (best) Encoder model, as this model has the simplest deep architecture and is easiest the visualize. Also, its visualization maps one-to-one to the input shape of the EEG recordings. It's important to realize that this is only a visualization of the input layer to the first dense layer of the network and that a neural network is very complex and cannot be simply explained by a few visualizations. However, it might give a little insight into how the network works.

In the first visualization, figure 18, it's possible to see which channels and time steps have higher weights during the recording. Each small rectangle represents the weight of a 2ms sample (500 Hz) from a single channel, which is used as an input feature for the network. Here, we see that in the beginning (first 200ms) especially the channels `P4, C4`, and `P3` have high weights. This means that they have an above-average contribution to the prediction process in the first part of the network. Then, in the part between 200 and 400ms, most features seem to have relatively low weights. Between 400 and 600ms, the higher weights are seen in the channels `TP8, T8`, and `TP7`. `TP7` remains to have a higher weight in the next 200ms, whereas `T8` and `TP7` remain to have relatively high weights until the end. `FT8` also has relatively higher weights between 600 and 1000ms.

The second visualization (figure 19) eliminates the time component and shows the mean absolute weight over all time steps of the channels. Here we observe that indeed the aforementioned channels have higher weights overall, especially `T8, FT8`, and `TP7`. But, we are also able to see here all channels contribute to the first stage of the prediction process. The channel with the

highest mean weight (T8) is only approximately 3.1 times larger than the smallest (O2). This is not a very large difference, meaning that information from all channels is used by the network.

The last visualization (figure 20) eliminates the channels component and shows the mean absolute weight over all channels of each time step. Here, it is visible that the first stage of the EEG recording between 0-200ms and the part between 400-600ms are contributing more to the neural network's prediction in the first stage of the network. Interestingly, the auditory stimulus is presented to the subject at 200ms and the main focus is before and after the period immediately after the stimulus. This might be explained by the delay between the stimulus and the response of the subject. Then, when there is a response, this has a high weight in the network. Further research could explore this more.

These visualizations are only an outlook on what might be possible in the field of making a deep neural network more explainable. These visualizations were made based on the fully-connected neural network, which is not the best model we have developed. Therefore, further research could investigate how to map the weights of the Encoder model to the input features. Furthermore, it would be very interesting to find out whether cognitive scientists, developmental scientists, psychiatrists, or other experts in the field can explain the focus on certain channels or time ranges of the EEG recording by the neural network.

# 7 Conclusion

Looking at the results and the discussion of the results, we can draw a few conclusions from this thesis research project. One of the reasons behind this project was to explore whether deep learning models can predict (developmental) age more accurately using EEG data than traditional machine learning models. Based on the results, we can conclude that it is possible to use deep learning regression to predict the (developmental) age of infants and children with EEG data. In general, deep learning models are performing better on this task than traditional machine learning models.

Age prediction on EEG data has been done before with deep learning classification or traditional machine learning regression, but this is the first research of its kind, where deep learning regression models are applied to EEG data without feature extraction to predict the ages of children or even people in general. Using regression opened the possibility to analyze the brain age gap and therefore differences in the development of subjects. Also, regression's interpretability is higher than classification, as we can determine how close the prediction is compared to the chronological age.

A key strength of deep learning is the ability to learn different representations of the input data, and therefore they can extract features from raw data. The deep learning models have higher performance on predicting the developmental age on (processed) raw EEG data compared to traditional machine learning models using features extracted from the EEG data. Out of all tested architectures, the Encoder model is the deep learning model that does this best, whereas the support vector regressor is the best-performing traditional machine learning model. The fact that several of the tested deep learning models outperformed traditional ML models indicated that the feature extraction and selection process can be avoided. This is an advantage because it requires less domain knowledge, as the researcher does not have to investigate which features are usually extracted in EEG research and opens the door for more researchers from other fields to participate in EEG-based research. Another advantage is that the deep learning models might be able to extract features or characteristics from the EEG data that humans are not aware of that they are of importance.

Interestingly, the BLSTM-LSTM model does not perform well. The best original model [11] used extracted features, and it seems like it is not able to extract features from raw data very well. This could be due to the architecture type, it is the only recurrent neural network, or it might be the case that the model is too simple.

An important assumption we made, based on the hypothesis of Vandenbosch [5] and colleagues, is that the brain age gaps of a subject should be stable over time. This assumption is confirmed for all models. The brain age gaps have a moderate to strong positive correlation with

the brain age gap at follow-up for all models.

We have externally validated the models by comparing the predicted brain age gap against the receptive and expressive vocabulary of the subjects. For the regular deep learning models, there is a significant moderate to a strong positive correlation between the models' brain age gap predictions at the age of 17 months and the receptive vocabulary size of the subjects. For the cross-validated Encoder model, we see a weakly positive correlation between the brain age gap and the receptive vocabulary at the age of 35 months, and for the expressive vocabulary at the ages of 23 and 35 months.

This is not enough evidence to conclude that the predicted age reflects the maturational level at all ages. However, the results do give some cause for optimism, as this does mean that the models' predictions do contain information about the subjects' development, even if it is just for a few age groups. The other results were not significant, but further research would be very interesting, to determine whether this relationship genuinely does not exist or whether it is due to the sample size.

Unfortunately, we were not able to find a significant difference between the brain age gap distributions of the dyslexic and non-dyslexic groups, but the mean ages of the non-dyslexic subjects are higher for almost all models and age groups. This also gives reason to explore this relationship further.

We have attempted to make the predictions of the (deep learning) models more explainable, especially by the weights inspection of the input layer to the first dense layer of the fully-connected neural network. We have been able to show that this network is focusing more on specific channels in the EEG recordings (`T8, FT8, TP7`). Also, the channels it focuses on change over time. Furthermore, the first 200 milliseconds and the range between 400 and 600 milliseconds of the recording have the highest weights, meaning that these contribute most to the first stage of the prediction process.

These steps to make the networks more explainable are only small steps, and there is more possible to make them more explainable. However, this does give an outlook on what is possible in the context of making a deep neural network explainable. The next steps would include visualizing the weights of the best model (Encoder) as well and making an attempt to relate these findings to actual physiological events in the human brain.

Lastly, we have shown that deep learning models can use noisy data like EEG data to determine the (developmental) age of a subject relatively well. Parts of the data set we have used are from 2002, EEG recording techniques have improved since and have become less noisy. Less noisy data could improve the performance of the models. Furthermore, the data set used was not specifically recorded with developmental age prediction in mind, and therefore it might not

be ideal for the task at hand. There are also other improvements possible to our experiments, for example making use of the oddball paradigm. The data set we used did make use of this paradigm, but we have not used this as it decreased the size of our data set drastically, which is an issue for training deep learning models. Another improvement might be using more EEG channels. As we have shown that deep learning models can improve the performance on a task compared to traditional machine learning models, and because there is still room for improvement on both the data and the model side, we expect that it is possible to use deep learning models to find other biomarkers in EEG data as well.

# References


[1] Benjamin Zablotsky et al. “Prevalence and trends of developmental disabilities among children in the United States: 2009–2017”. In: *Pediatrics* 144.4 (2019).

[2] Orazio Attanasio et al. “Human capital growth and poverty: Evidence from Ethiopia and Peru”. In: *Review of economic dynamics* 25 (2017), pp. 234–259.

[3] Linda Richter et al. “Early childhood development: an imperative for action and measurement at scale”. In: *BMJ global health* 4.Suppl 4 (2019), e001302.

[4] Hedieh Sajedi and Nastaran Pardakhti. “Age prediction based on brain MRI image: a survey”. In: *Journal of medical systems* 43.8 (2019), p. 279.

[5] Marjolein MLJZ Vandenbosch et al. “EEG-based age-prediction models as stable and heritable indicators of brain maturational level in children and adolescents”. In: *Human brain mapping* 40.6 (2019), pp. 1919–1926.

[6] Obada Al Zoubi et al. “Predicting age from brain EEG signals—A machine learning approach”. In: *Frontiers in aging neuroscience* 10 (2018), p. 184.

[7] Katja Franke et al. “Brain maturation: predicting individual BrainAGE in children and adolescents using structural MRI”. In: *Neuroimage* 63.3 (2012), pp. 1305–1312.

[8] Hieu Pham et al. “Meta pseudo labels”. In: *arXiv preprint arXiv:2003.10580* (2020).

[9] Tom B Brown et al. “Language models are few-shot learners”. In: *arXiv preprint arXiv:2005.14165* (2020).

[10] Aryan van der Leij et al. “Precursors of developmental dyslexia: an overview of the longitudinal Dutch dyslexia programme study”. In: *Dyslexia* 19.4 (2013), pp. 191–213.

[11] Pallavi Kaushik et al. “EEG-based age and gender prediction using deep BLSTM-LSTM network model”. In: *IEEE Sensors Journal* 19.7 (2018), pp. 2634–2641.

[12] Andrea Biasiucci, Benedetta Franceschiello, and Micah M Murray. “Electroencephalography”. In: *Current Biology* 29.3 (2019), R80–R85.

[13] Jeffrey W Britton et al. *Electroencephalography (EEG): An introductory text and atlas of normal and abnormal findings in adults, children, and infants*. American Epilepsy Society, Chicago, 2016.

[14] Yannick Roy et al. “Deep learning-based electroencephalography analysis: a systematic review”. In: *Journal of neural engineering* 16.5 (2019), p. 051001.

[15] Mainak Jas et al. “Autoreject: Automated artifact rejection for MEG and EEG data”. In: *NeuroImage* 159 (2017), pp. 417–429.

[16] Erik K St Louis et al. "Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults". In: *Children, and Infants* (2016).

[17] Tom M Mitchell et al. "Machine learning. 1997". In: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[18] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[19] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[20] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. "A training algorithm for optimal margin classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. 1992, pp. 144–152.

[21] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[22] Harris Drucker et al. "Support vector regression machines". In: *Advances in neural information processing systems* 9 (1997), pp. 155–161.

[23] Michael E Tipping. "Sparse Bayesian learning and the relevance vector machine". In: *Journal of machine learning research* 1.Jun (2001), pp. 211–244.

[24] Leo Breiman et al. *Classification and regression trees*. CRC press, 1984.

[25] Gareth James et al. *An introduction to statistical learning*. Vol. 112. Springer, 2013.

[26] J Russell Stuart and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2010.

[27] Kjell M Fauske. *Example: Neural network*. `https://texample.net/tikz/examples/neural-network/`. [Online; accessed 05-February-2021]. 2006.

[28] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[29] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7 (2011).

[30] Matthew D Zeiler. "Adadelta: an adaptive learning rate method". In: *arXiv preprint arXiv:1212.5701* (2012).

[31] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[32] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent". In: *Cited on* 14.8 (2012).

[33] Timothy Dozat. "Incorporating nesterov momentum into adam". In: (2016).

[34] Geoffrey Hinton, Yann LeCun, and Yoshua Bengio. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[35] Zhou Lu et al. "The expressive power of neural networks: A view from the width". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 6232–6240.

[36] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[37] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. "Learning to forget: Continual prediction with LSTM". In: *Neural Computation* (1999).

[38] Kyunghyun Cho et al. "On the properties of neural machine translation: Encoder-decoder approaches". In: *arXiv preprint arXiv:1409.1259* (2014).

[39] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[40] Wikimedia Commons. *Long Short-Term Memory*. `https://commons.wikimedia.org/w/index.php?title=File:Long_Short-Term_Memory.svg&oldid=488381453`. [Online; accessed 30-January-2021]. 2020.

[41] Mathworks. *Convolutional neural network*. `https://nl.mathworks.com/discovery/convolutional-neural-network-matlab.html`. [Online; accessed 30-January-2021]. 2018.

[42] Hugo G Schnack et al. "Accelerated brain aging in schizophrenia: a longitudinal pattern recognition study". In: *American Journal of Psychiatry* 173.6 (2016), pp. 607–616.

[43] Donald B Lindsley. "A longitudinal study of the occipital alpha rhythm in normal children: frequency and amplitude standards". In: *The Pedagogical Seminary and Journal of Genetic Psychology* 55.1 (1939), pp. 197–213.

[44] Filippo Zappasodi et al. "Age-related changes in electroencephalographic signal complexity". In: *PloS one* 10.11 (2015), e0141995.

[45] Andrey P Anokhin et al. "Age increases brain complexity". In: *Electroencephalography and clinical Neurophysiology* 99.1 (1996), pp. 63–68.

[46] AKI Chiang et al. "Age trends and sex differences of alpha rhythms including split alpha peaks". In: *Clinical Neurophysiology* 122.8 (2011), pp. 1505–1517.

[47] Adam R Clarke et al. “Age and sex effects in the EEG: development of the normal child”. In: *Clinical neurophysiology* 112.5 (2001), pp. 806–814.

[48] Robert J Barry et al. “Age and gender effects in EEG coherence: I. Developmental trends in normal children”. In: *Clinical neurophysiology* 115.10 (2004), pp. 2252–2258.

[49] P Matthis et al. “Changes in the background activity of the electroencephalogram according to age”. In: *Electroencephalography and clinical neurophysiology* 49.5-6 (1980), pp. 626–635.

[50] Theo Gasser et al. “Development of the EEG of school-age children and adolescents. I. Analysis of band power”. In: *Electroencephalography and clinical neurophysiology* 69.2 (1988), pp. 91–99.

[51] Lucy Cragg et al. “Maturation of EEG power spectra in early adolescence: a longitudinal study”. In: *Developmental science* 14.5 (2011), pp. 935–943.

[52] C Benninger, P Matthis, and D Scheffner. “EEG development of healthy boys and girls. Results of a longitudinal study”. In: *Electroencephalography and clinical neurophysiology* 57.1 (1984), pp. 1–12.

[53] Peter J Marshall, Yair Bar-Haim, and Nathan A Fox. “Development of the EEG from 5 months to 4 years of age”. In: *Clinical Neurophysiology* 113.8 (2002), pp. 1199–1208.

[54] Mohammad-Parsa Hosseini, Amin Hosseini, and Kiarash Ahi. “A Review on Machine Learning for EEG Signal Processing in Bioengineering”. In: *IEEE reviews in biomedical engineering* (2020).

[55] Phuoc Nguyen et al. “EEG-based age and gender recognition using tensor decomposition and speech features”. In: *International conference on neural information processing*. Springer. 2013, pp. 632–639.

[56] Phuoc Nguyen et al. “Age and gender classification using EEG paralinguistic features”. In: *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*. IEEE. 2013, pp. 1295–1298.

[57] Barjinder Kaur, Dinesh Singh, and Partha Pratim Roy. “Age and gender classification using brain–computer interface”. In: *Neural Computing and Applications* 31.10 (2019), pp. 5887–5900.

[58] Lilian Ribeiro Mendes de Paiva et al. “Analysis of the relationship between EEG signal and aging through Linear Discriminant Analysis (LDA)”. In: *Revista Brasileira de Engenharia Biomedica* 28.2 (2012), pp. 155–168.

[59] Nik Khadijah Nik Aznan et al. “On the classification of SSVEP-based dry-EEG signals via convolutional neural networks”. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2018, pp. 3726–3731.

[60] Vernon J Lawhern et al. “EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces”. In: *Journal of neural engineering* 15.5 (2018), p. 056013.

[61] Alexandre Barachant et al. “Classification of covariance matrices using a Riemannian-based kernel for BCI applications”. In: *Neurocomputing* 112 (2013), pp. 172–178.

[62] James H Cole et al. “Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker”. In: *NeuroImage* 163 (2017), pp. 115–124.

[63] Hassan Ismail Fawaz et al. “Inceptiontime: Finding alexnet for time series classification”. In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[64] Hassan Ismail Fawaz et al. “Deep learning for time series classification: a review”. In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[65] Zhiguang Wang, Weizhong Yan, and Tim Oates. “Time series classification from scratch with deep neural networks: A strong baseline”. In: *2017 International joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

[66] Joan Serrà, Santiago Pascual, and Alexandros Karatzoglou. “Towards a Universal Neural Network Encoder for Time Series.” In: *CCIA*. 2018, pp. 120–129.

[67] Bendong Zhao et al. “Convolutional neural networks for time series classification”. In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169.

[68] Zhicheng Cui, Wenlin Chen, and Yixin Chen. “Multi-scale convolutional neural networks for time series classification”. In: *arXiv preprint arXiv:1603.06995* (2016).

[69] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. “Data augmentation for time series classification using convolutional neural networks”. In: *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2016.

[70] Yi Zheng et al. “Time series classification using multi-channels deep convolutional neural networks”. In: *International conference on web-age information management*. Springer. 2014, pp. 298–310.

[71] Yi Zheng et al. “Exploiting multi-channels deep convolutional neural networks for multivariate time series classification”. In: *Frontiers of Computer Science* 10.1 (2016), pp. 96–112.

[72] Pattreeya Tanisaro and Gunther Heidemann. “Time series classification using time warping invariant echo state networks”. In: *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2016, pp. 831–836.

[73] Jason Lines, Sarah Taylor, and Anthony Bagnall. “Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification”. In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 1041–1046.

[74] Alexandre Gramfort et al. “MEG and EEG data analysis with MNE-Python”. In: *Frontiers in neuroscience* 7 (2013), p. 267.

[75] J-B Schiratti et al. “An ensemble learning approach to detect epileptic seizures from long intracranial EEG recordings”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 856–860.

[76] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[77] Martın Abadi et al. “Tensorflow: Large-scale machine learning on heterogeneous distributed systems”. In: *arXiv preprint arXiv:1603.04467* (2016).

[78] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

[79] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[80] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).

[81] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *international conference on machine learning*. PMLR. 2016, pp. 1050–1059.

[82] Habtamu M Aycheh et al. “Biological brain age prediction using cortical thickness data: a large scale cohort study”. In: *Frontiers in aging neuroscience* 10 (2018), p. 252.

[83] Matthias S Treder et al. “Correlation constraints for regression models: controlling bias in brain age prediction”. In: *Frontiers in psychiatry* 12 (2021), p. 25.

# Supplementary materials

In this supplementary materials section, we present the graphs of the models that were not presented in the main content of the thesis. We will do this on a per-model basis, and for completeness, we will include the ones presented in the main content of the thesis as well. Lastly, we will present the graphs from the weights visualization section.

## Random forest

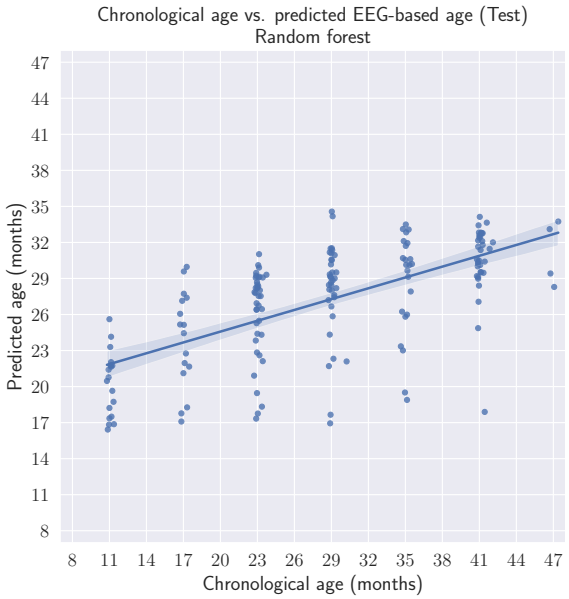



Figure 21: The predictions of the random forest compared to the true chronological ages of the subject on the test set.

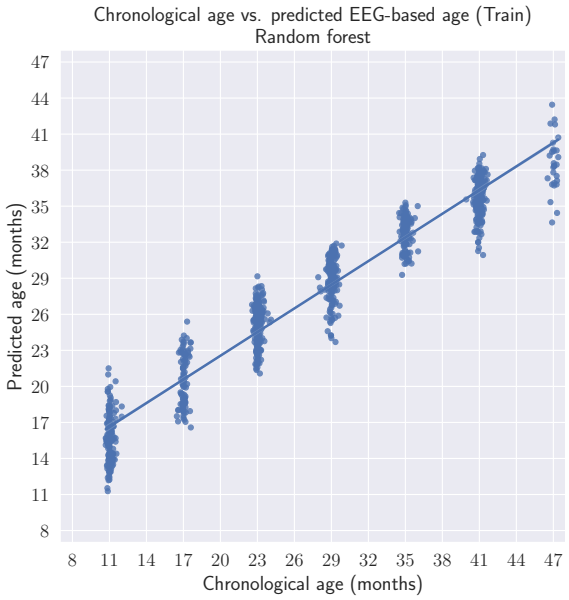



Figure 22: The predictions of the random forest compared to the true chronological ages of the subject on the train set.

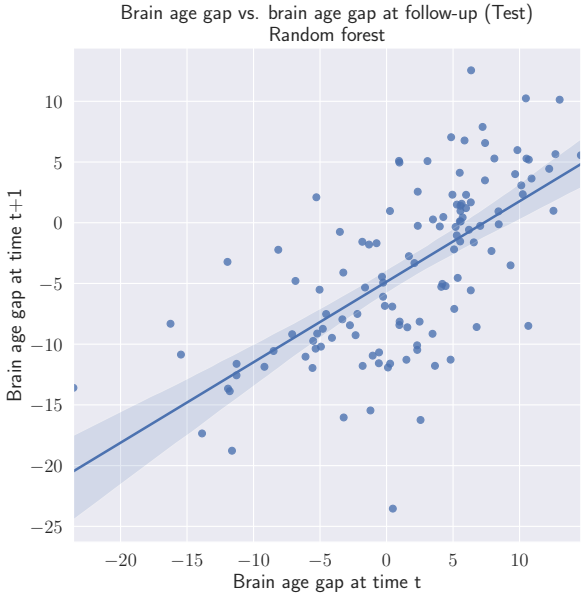
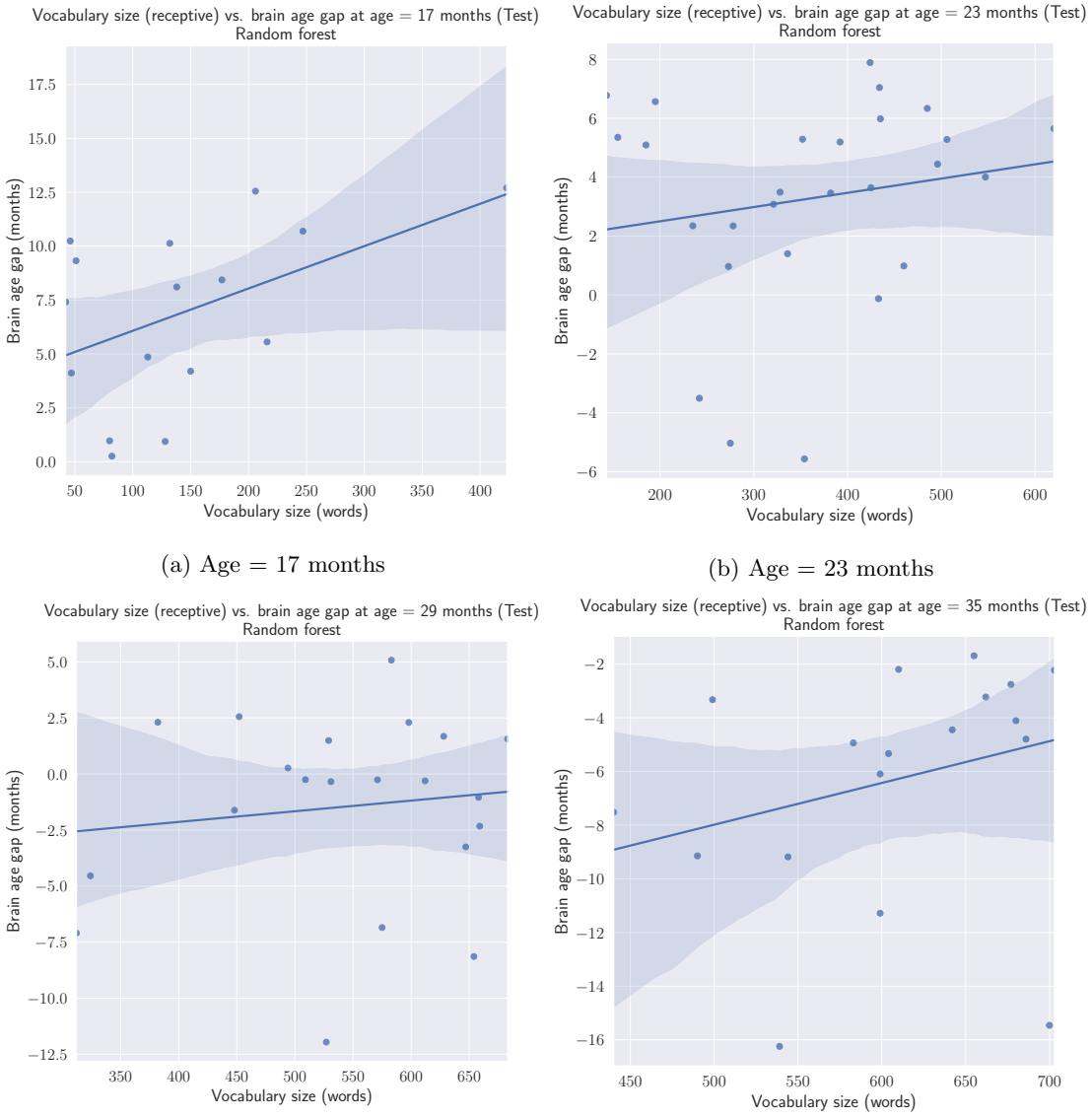



Figure 23: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the random forest on the test set.


Figure 24: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the random forest on the train set.



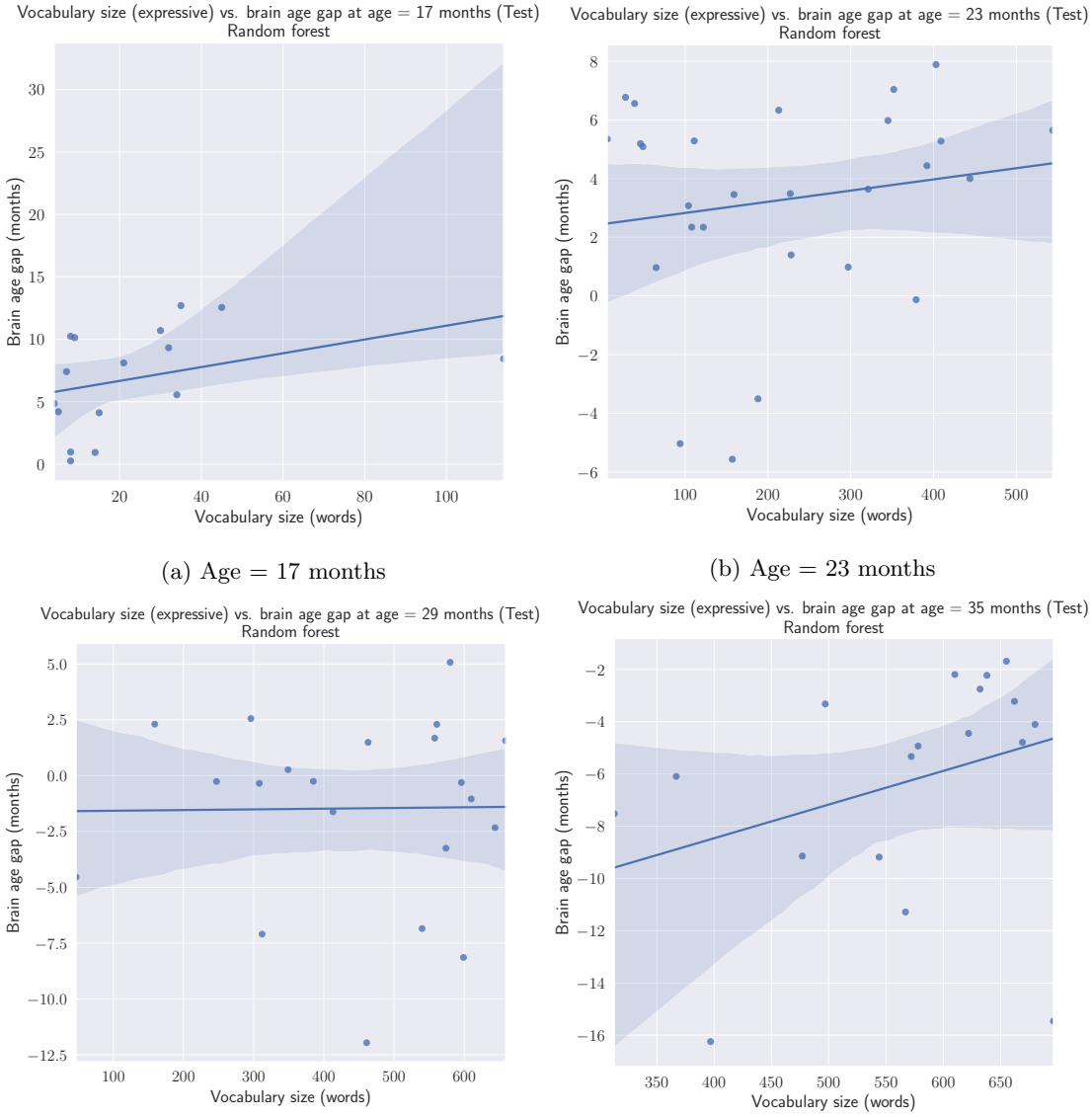

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 25: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the random forest's brain age predictions on the test set.



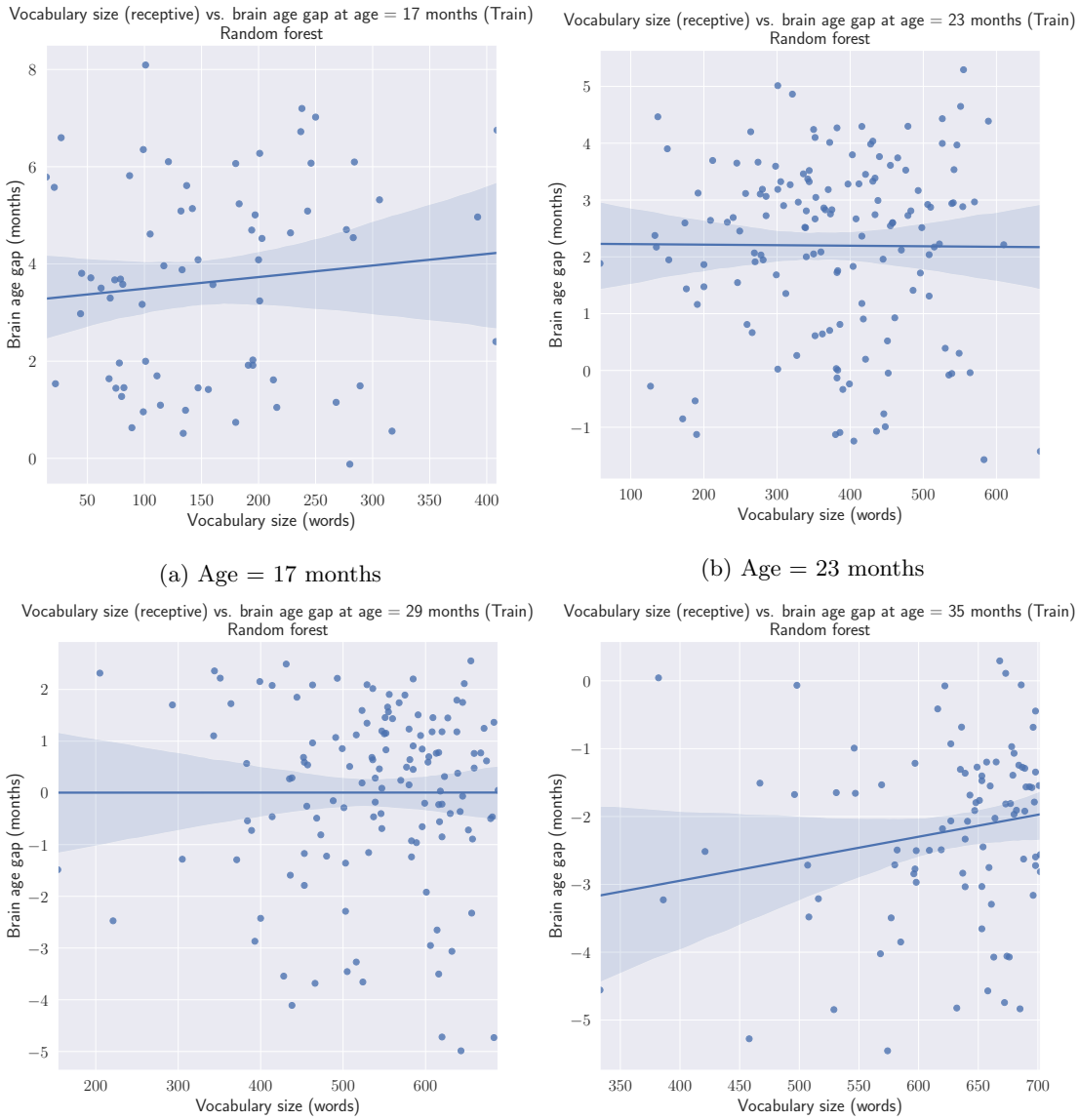

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 26: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the random forest's brain age predictions on the test set.



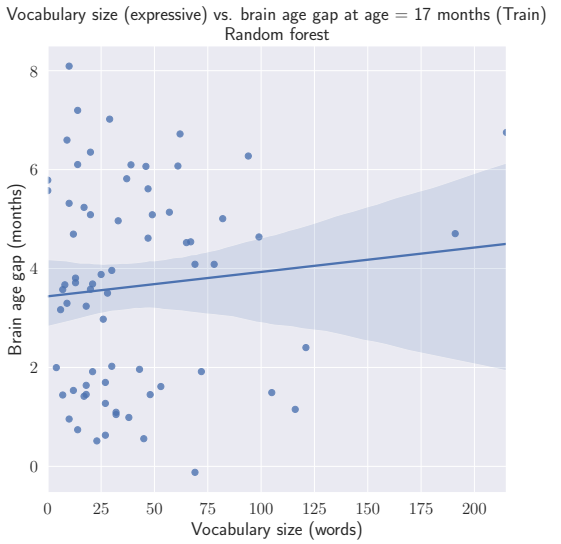

(a) Age = 17 months

(b) Age = 23 months

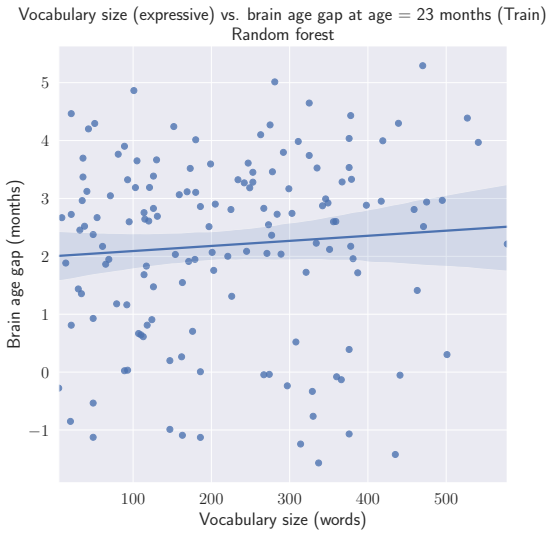
(c) Age = 35 months

(d) Age = 35 months

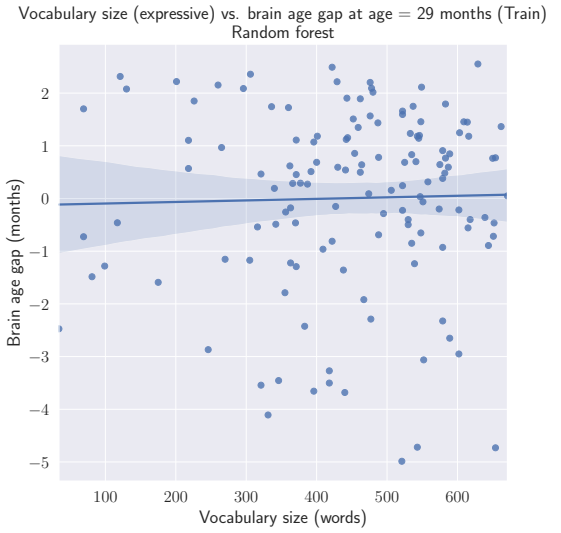
Figure 27: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the random forest's brain age predictions on the train set.



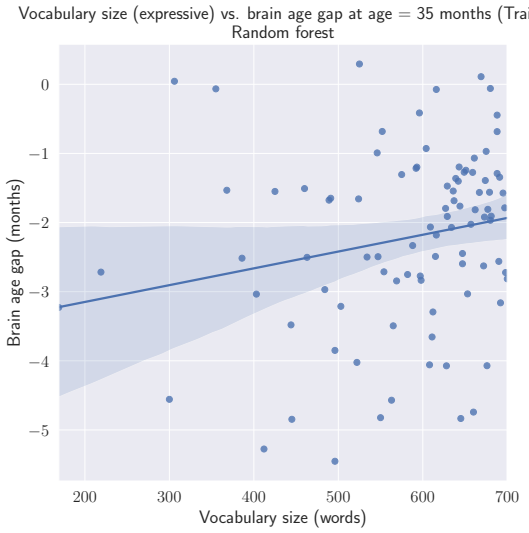

(a) Age = 17 months




(b) Age = 23 months




(c) Age = 29 months




(d) Age = 35 months

Figure 28: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the random forest's brain age predictions on the train set.

## Linear support vector regressor

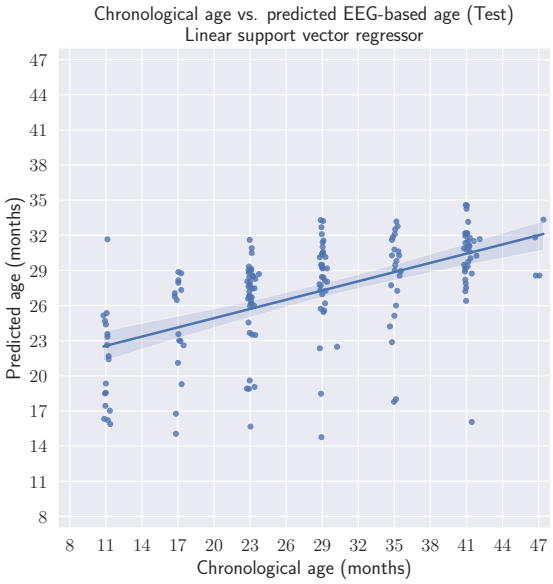



Figure 29: The predictions of the linear support vector regressor compared to the true chronological ages of the subject on the test set.

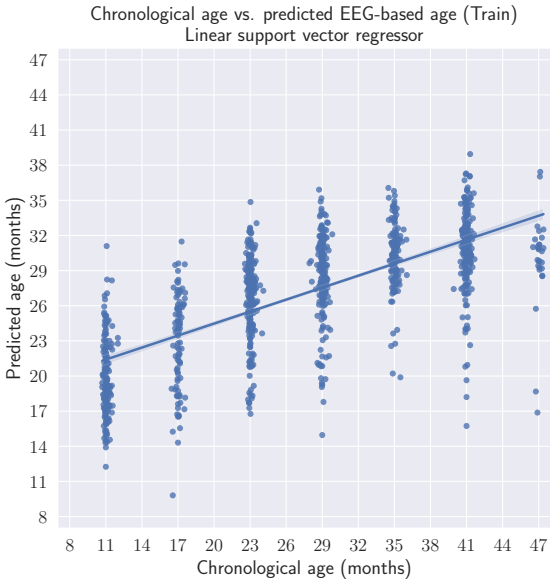



Figure 30: The predictions of the linear support vector regressor compared to the true chronological ages of the subject on the train set.

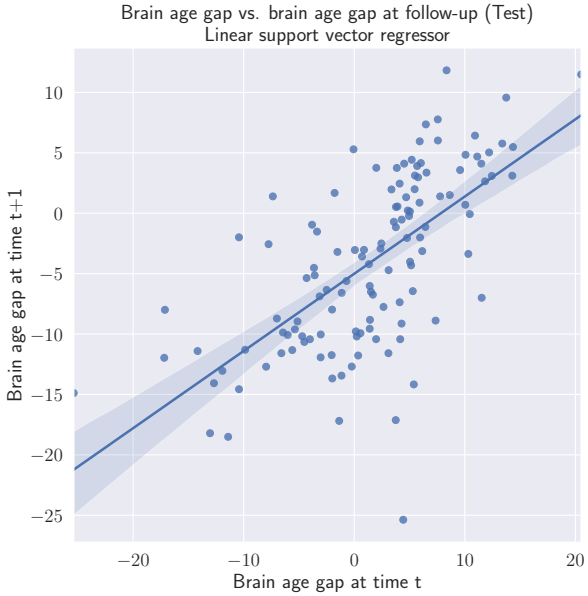



Figure 31: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the linear support vector regressor on the test set.

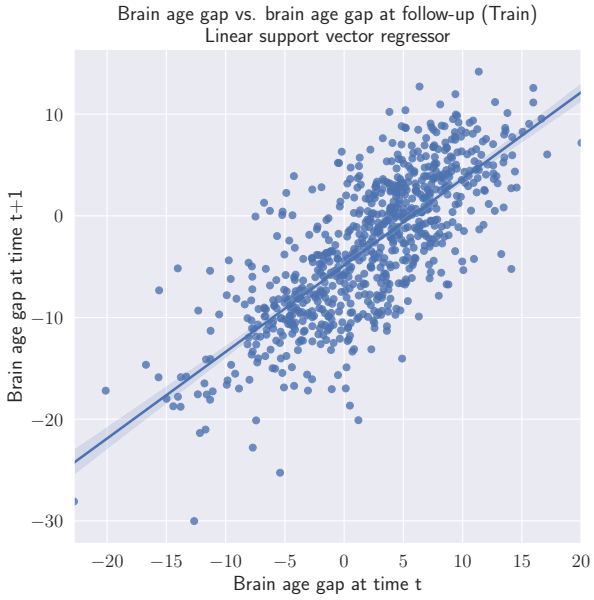
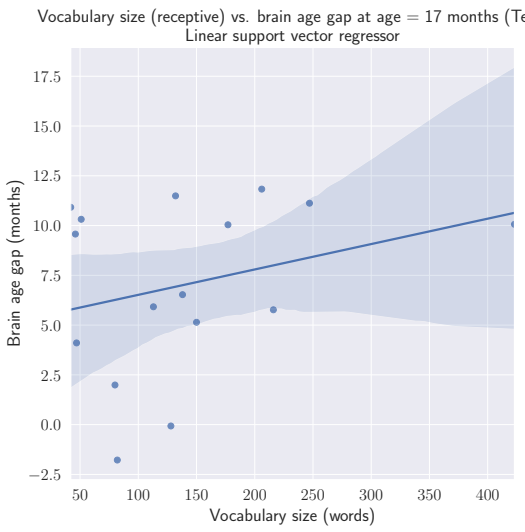
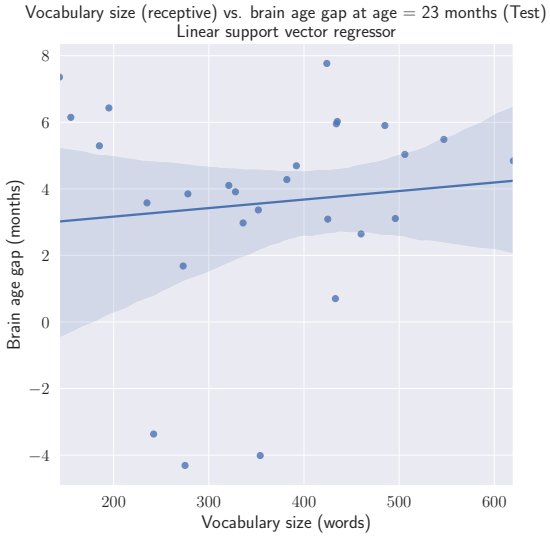


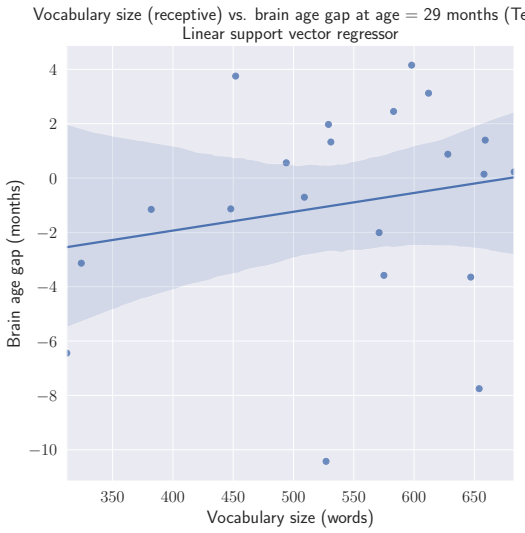

Figure 32: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the linear support vector regressor on the train set.



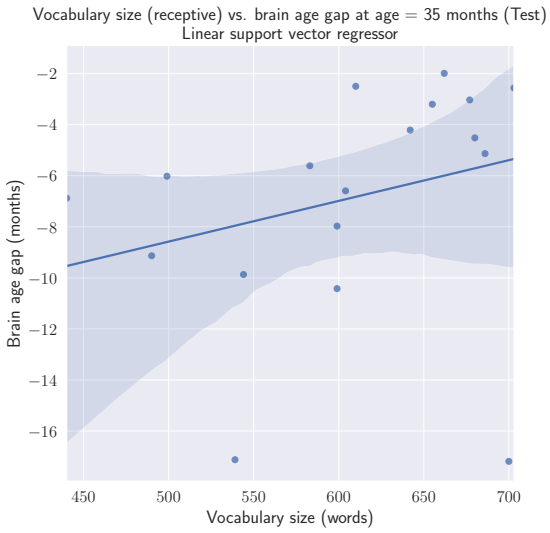

(a) Age = 17 months




(b) Age = 23 months

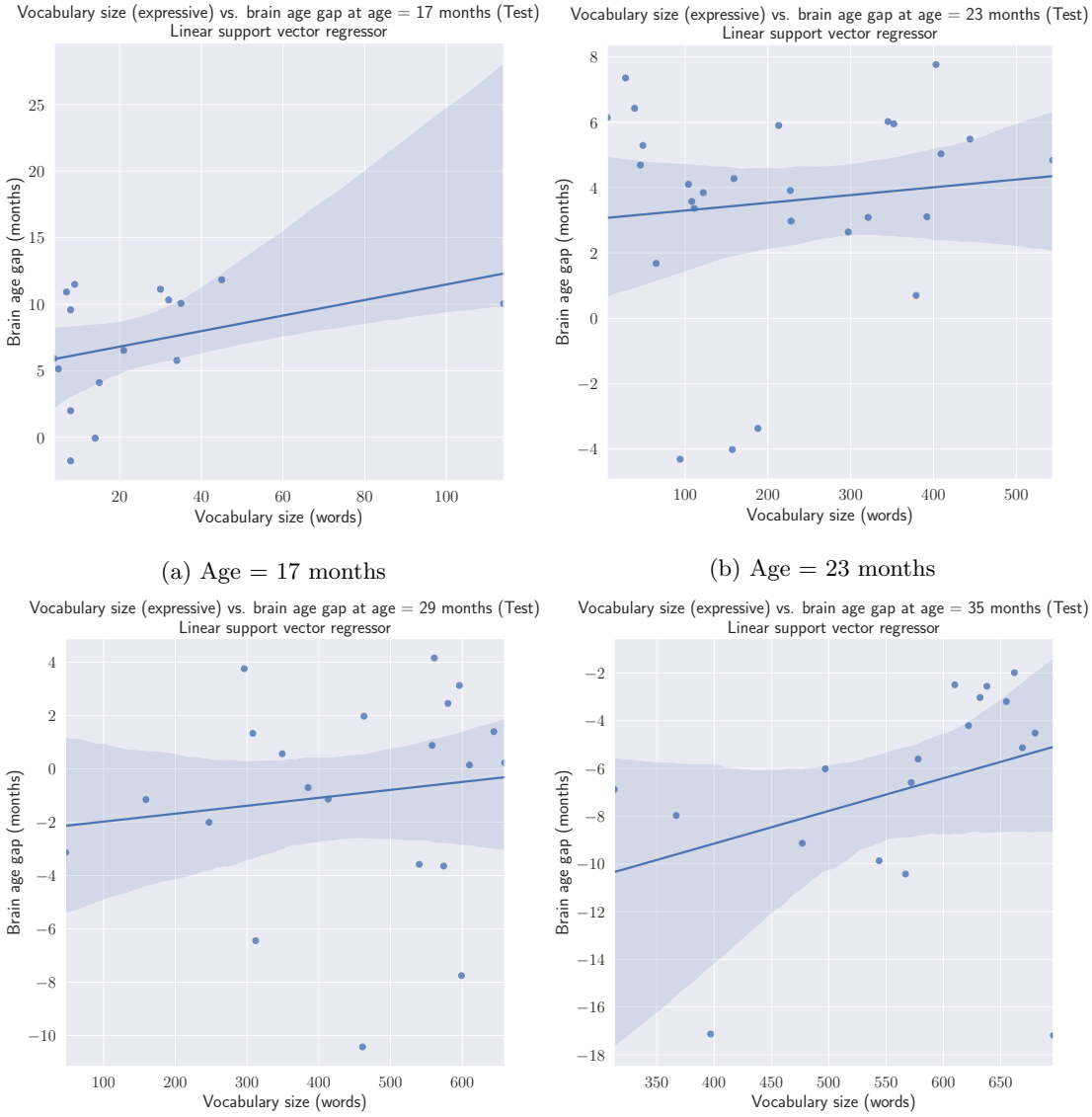



(c) Age = 35 months




(d) Age = 35 months

Figure 33: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the linear support vector regressor's brain age predictions on the test set.



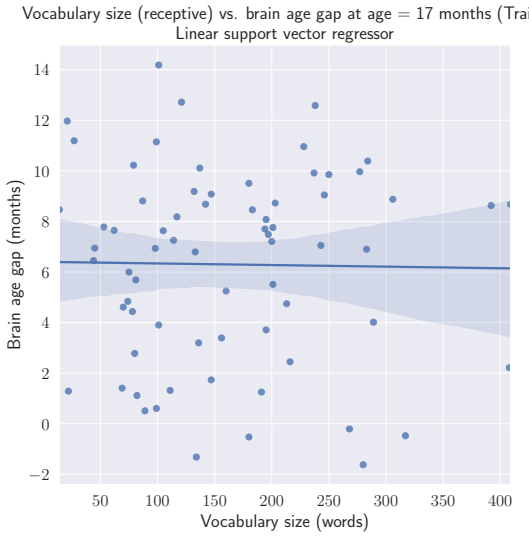

(a) Age = 17 months

(b) Age = 23 months

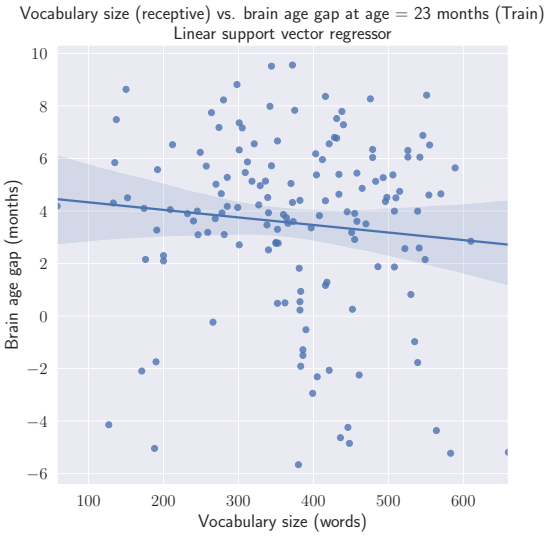
(c) Age = 29 months

(d) Age = 35 months

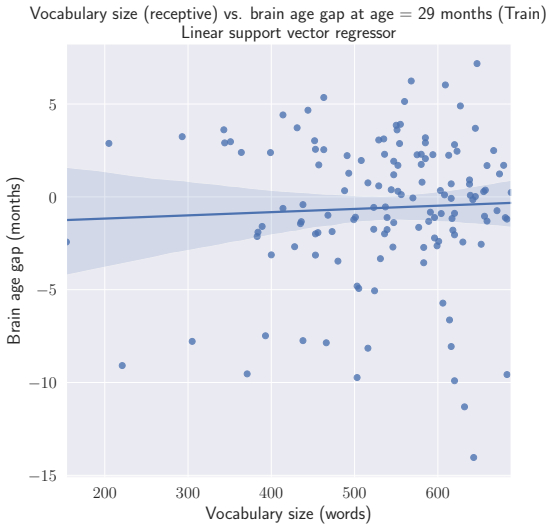
Figure 34: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the linear support vector regressor's brain age predictions on the test set.



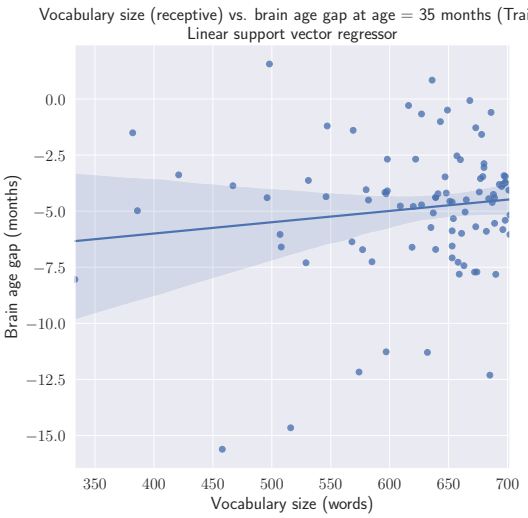

(a) Age = 17 months




(b) Age = 23 months

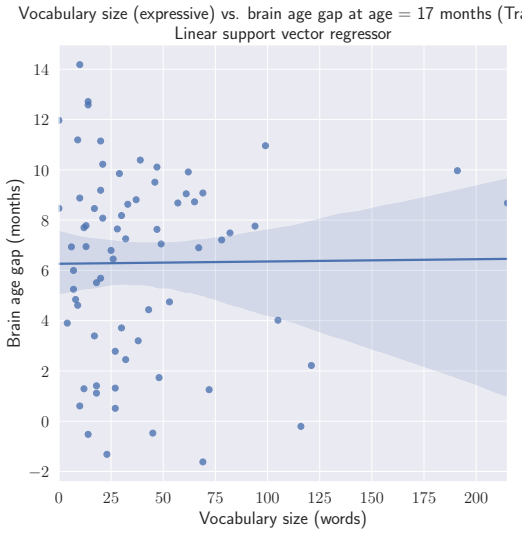



(c) Age = 35 months

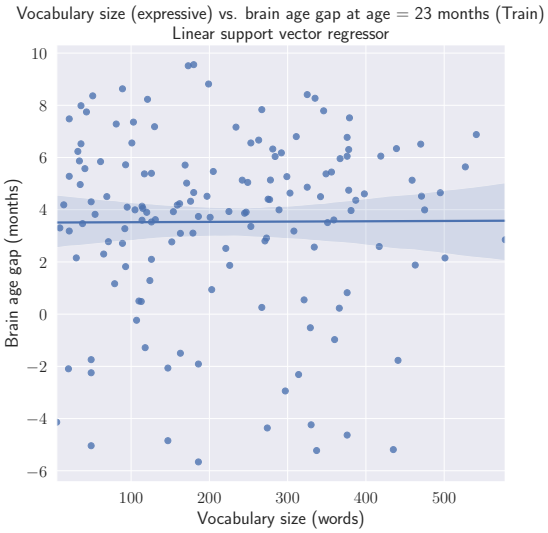



(d) Age = 35 months

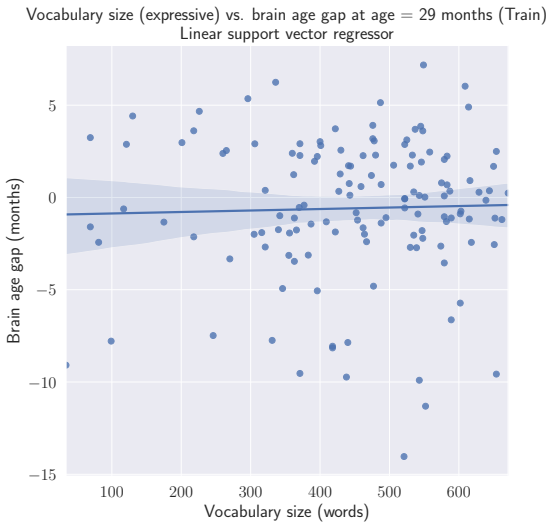
Figure 35: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the linear support vector regressor's brain age predictions on the train set.



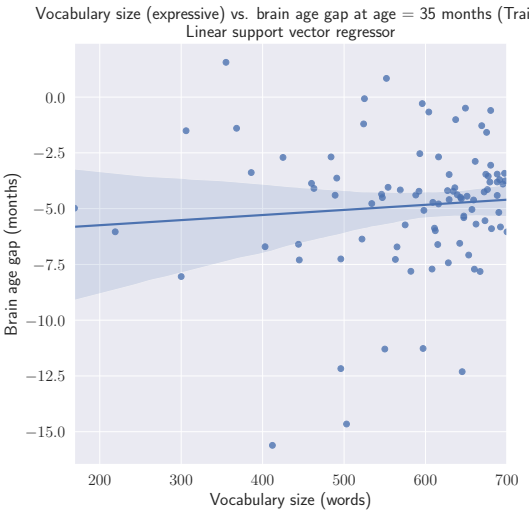

(a) Age = 17 months




(b) Age = 23 months




(c) Age = 29 months




(d) Age = 35 months

Figure 36: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the linear support vector regressor's brain age predictions on the train set.

### Support vector regressor

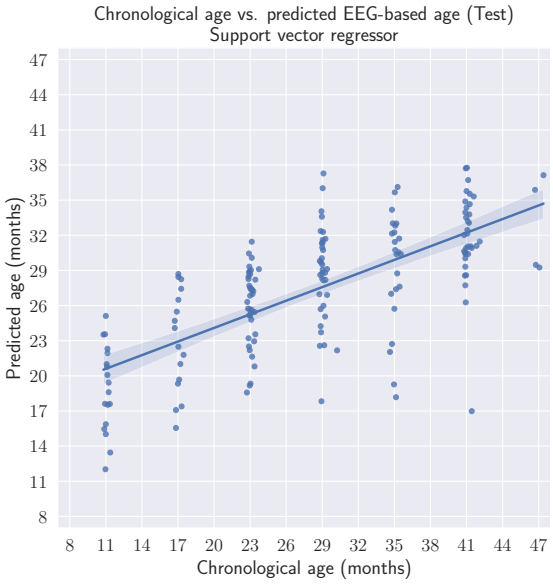



Figure 37: The predictions of the support vector regressor compared to the true chronological ages of the subject on the test set.

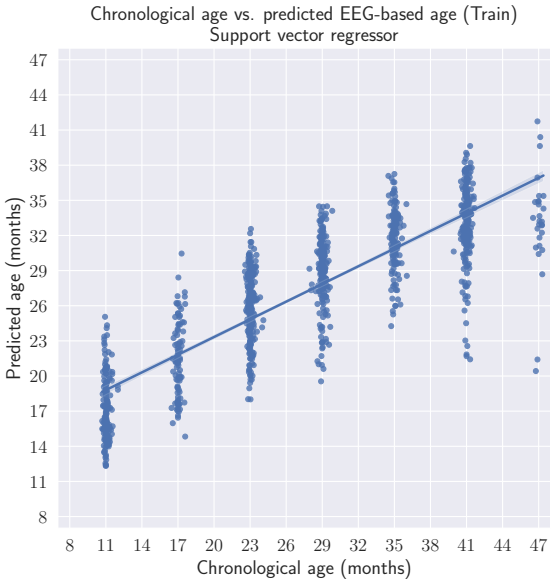



Figure 38: The predictions of the support vector regressor compared to the true chronological ages of the subject on the train set.

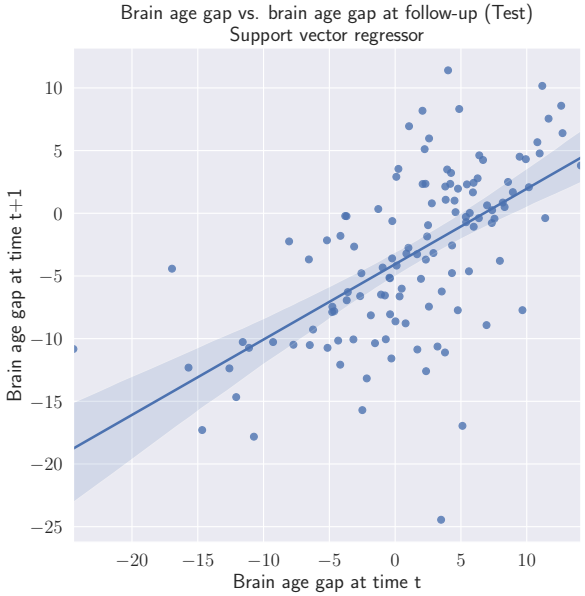



Figure 39: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the support vector regressor on the test set.

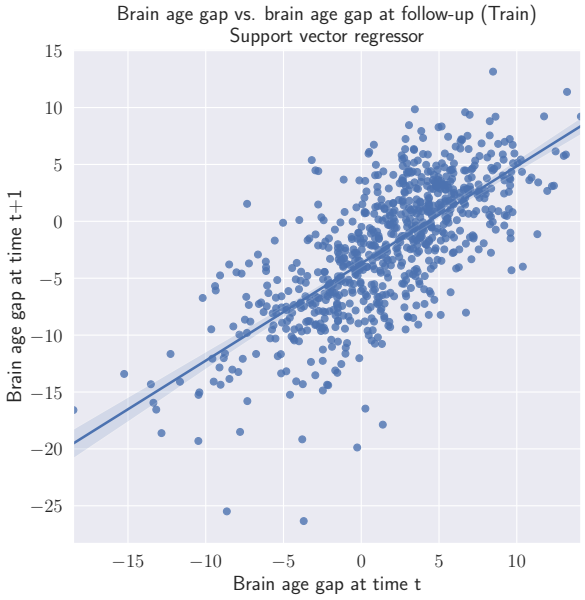
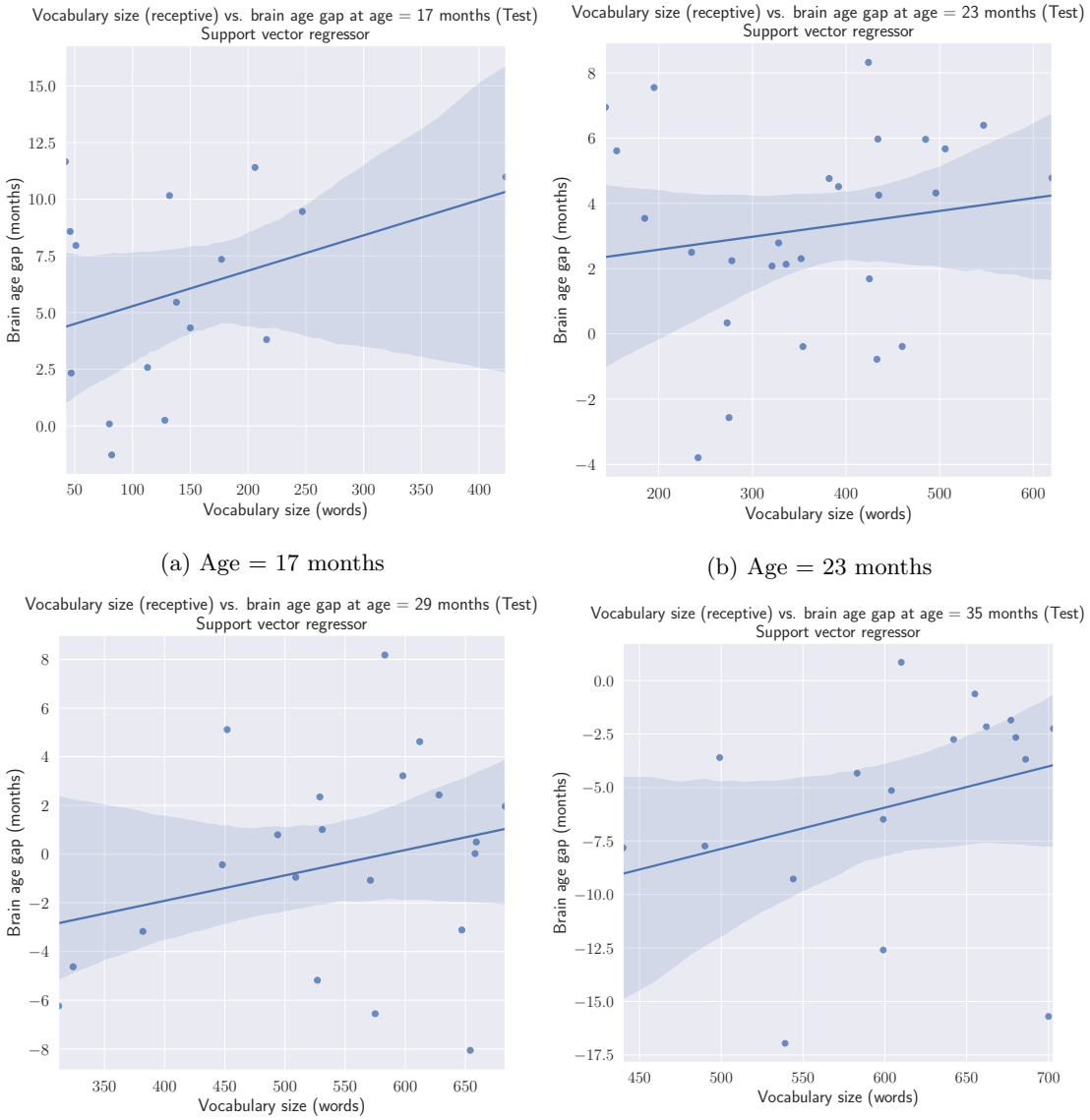



Figure 40: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the support vector regressor on the train set.



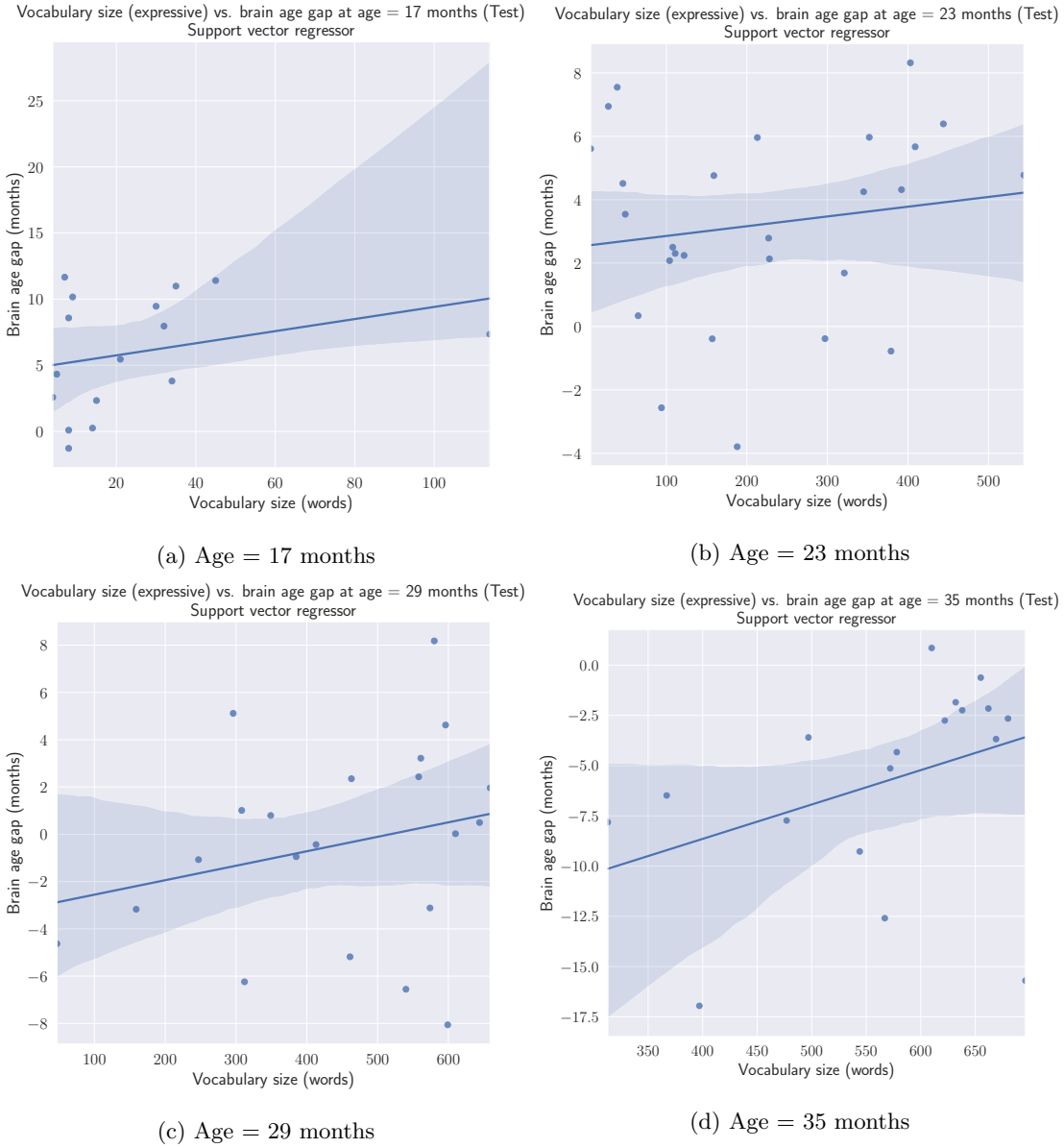

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 41: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the test set.



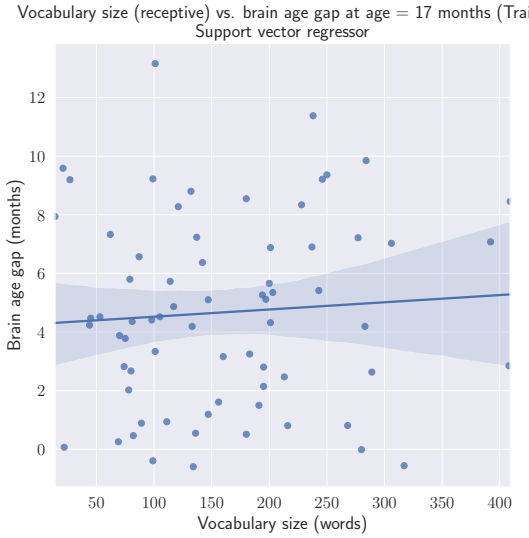

(a) Age = 17 months

(b) Age = 23 months

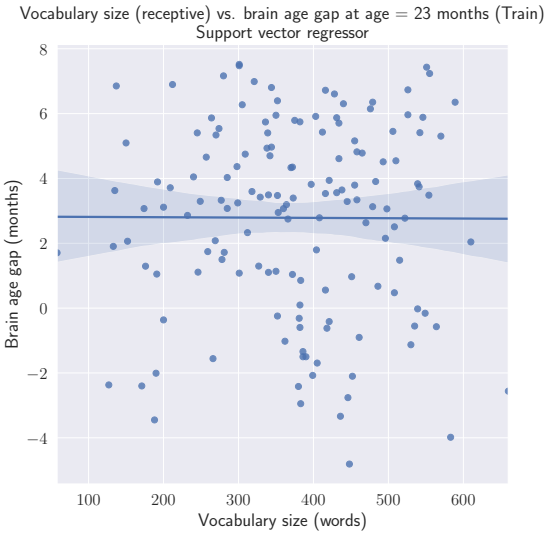
(c) Age = 29 months

(d) Age = 35 months

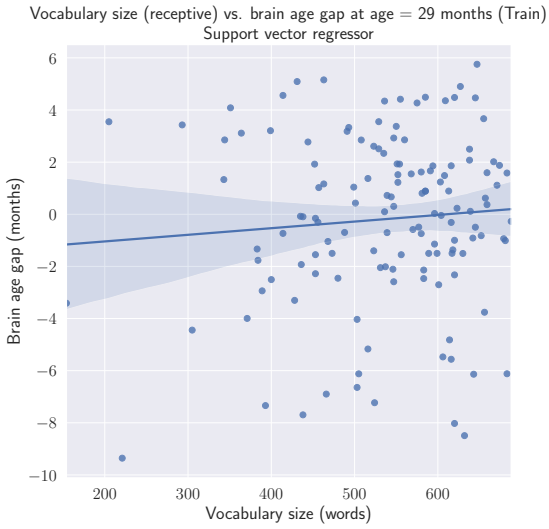
Figure 42: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the test set.



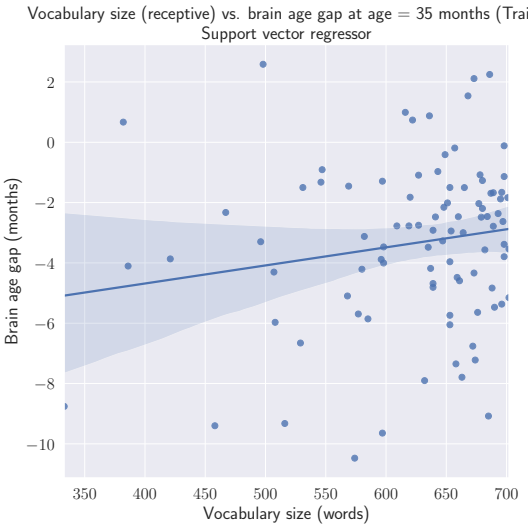

(a) Age = 17 months




(b) Age = 23 months

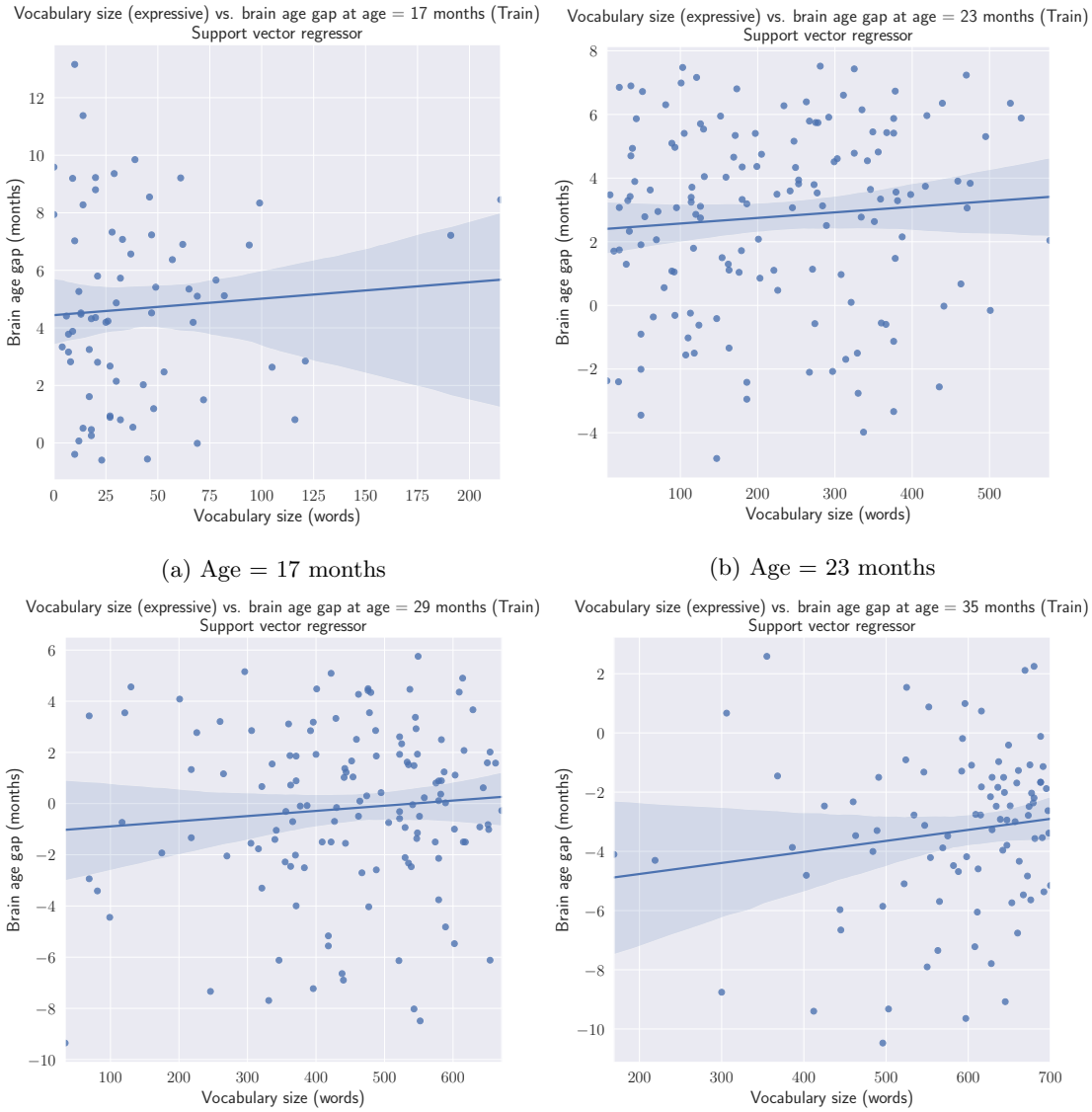



(c) Age = 35 months




(d) Age = 35 months

Figure 43: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the train set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 44: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the support vector regressor's brain age predictions on the train set.

## Relevance vector regressor

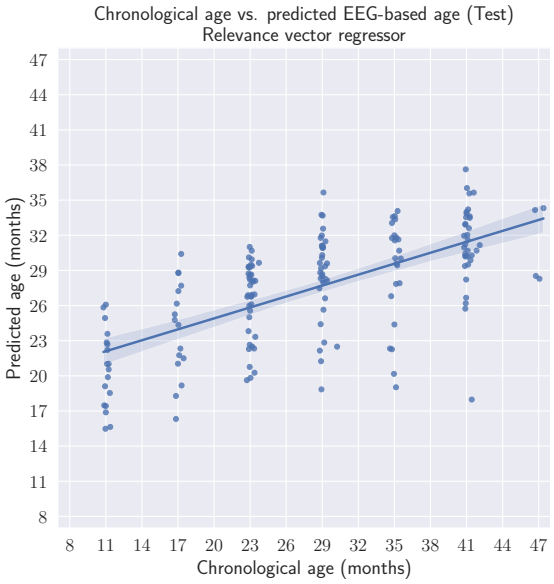



Figure 45: The predictions of the relevance vector regressor compared to the true chronological ages of the subject on the test set.

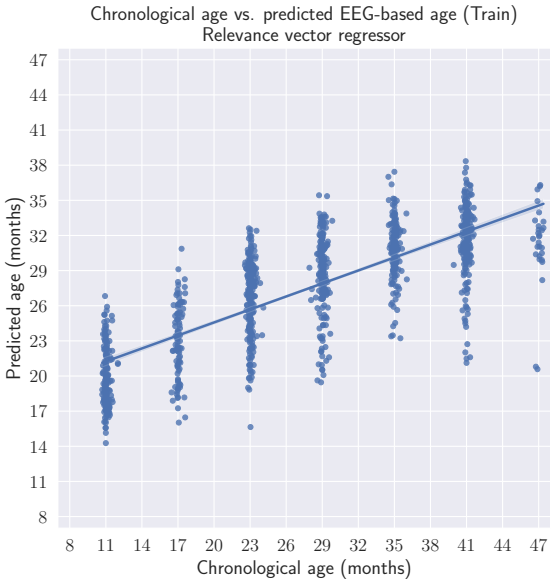
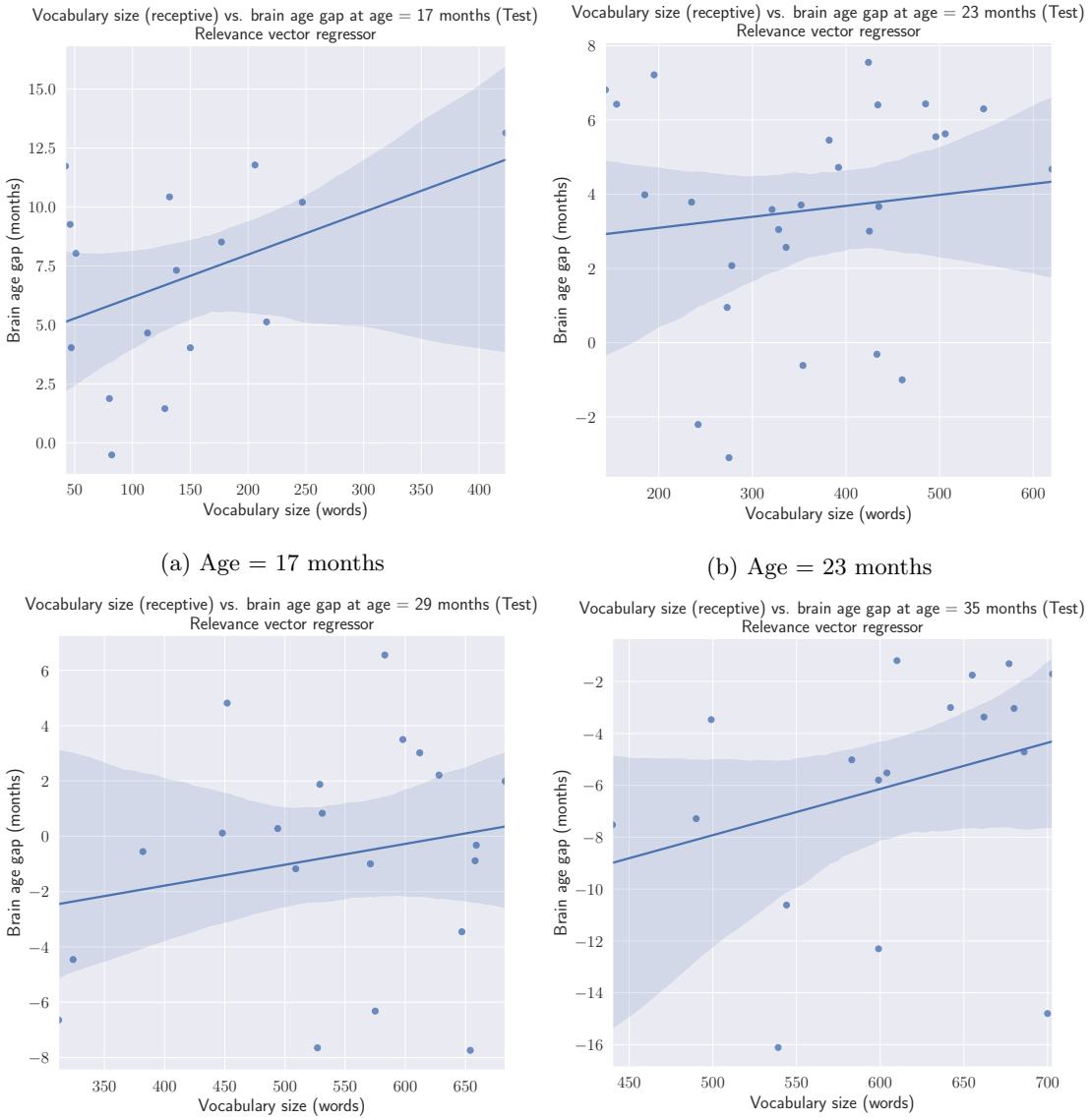



Figure 46: The predictions of the relevance vector regressor compared to the true chronological ages of the subject on the train set.


Figure 47: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the relevance vector regressor on the test set.


Figure 48: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the relevance vector regressor on the train set.



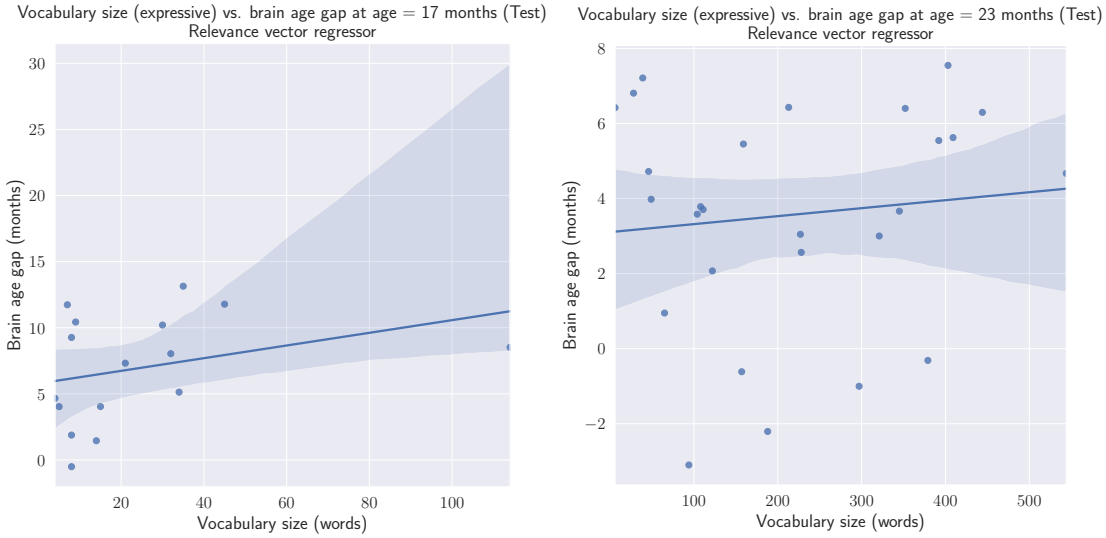

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

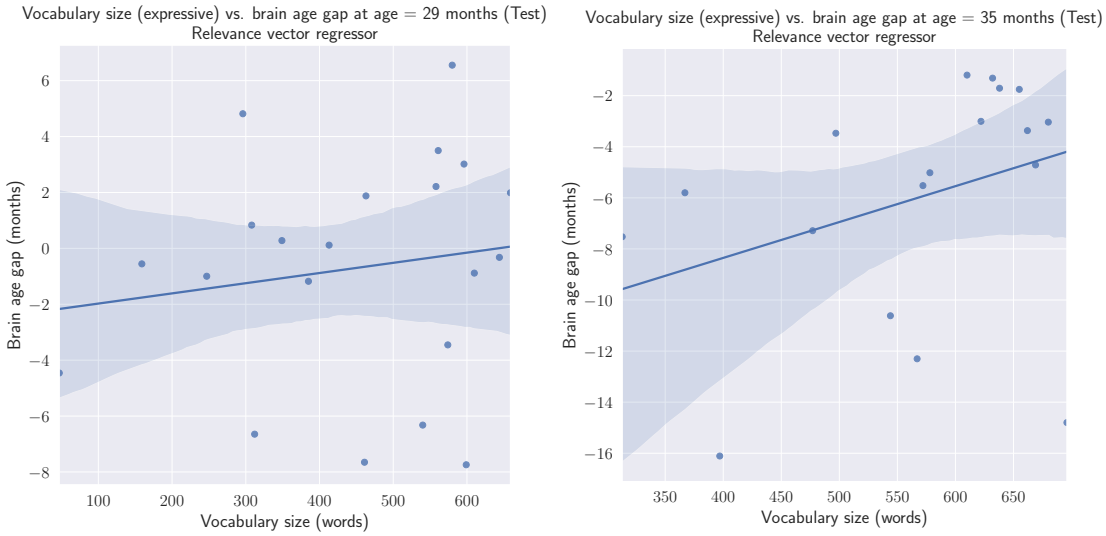
(d) Age = 35 months

Figure 49: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the relevance vector regressor's brain age predictions on the test set.




(a) Age = 17 months

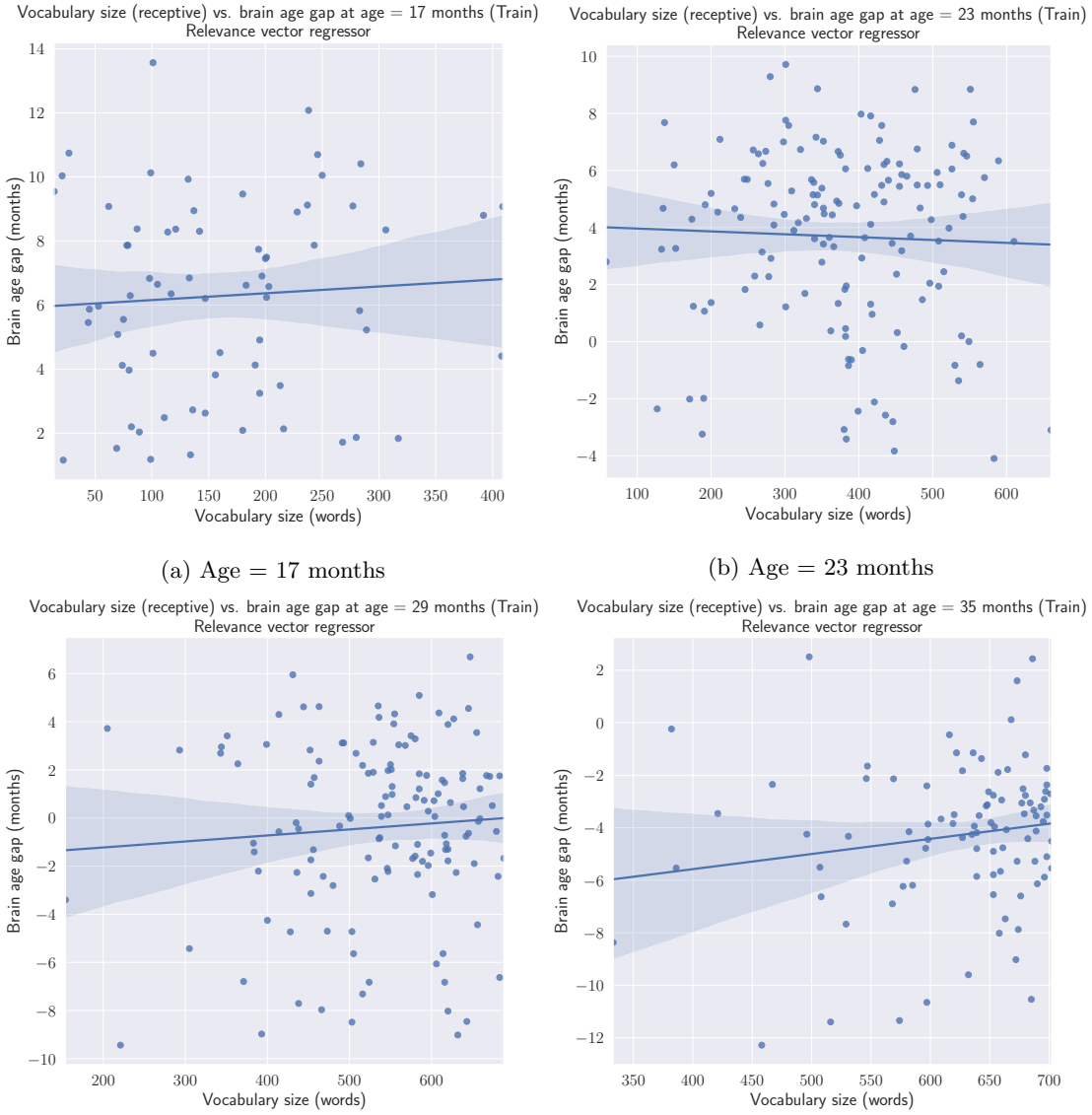
(b) Age = 23 months




(c) Age = 29 months

(d) Age = 35 months

Figure 50: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the relevance vector regressor's brain age predictions on the test set.



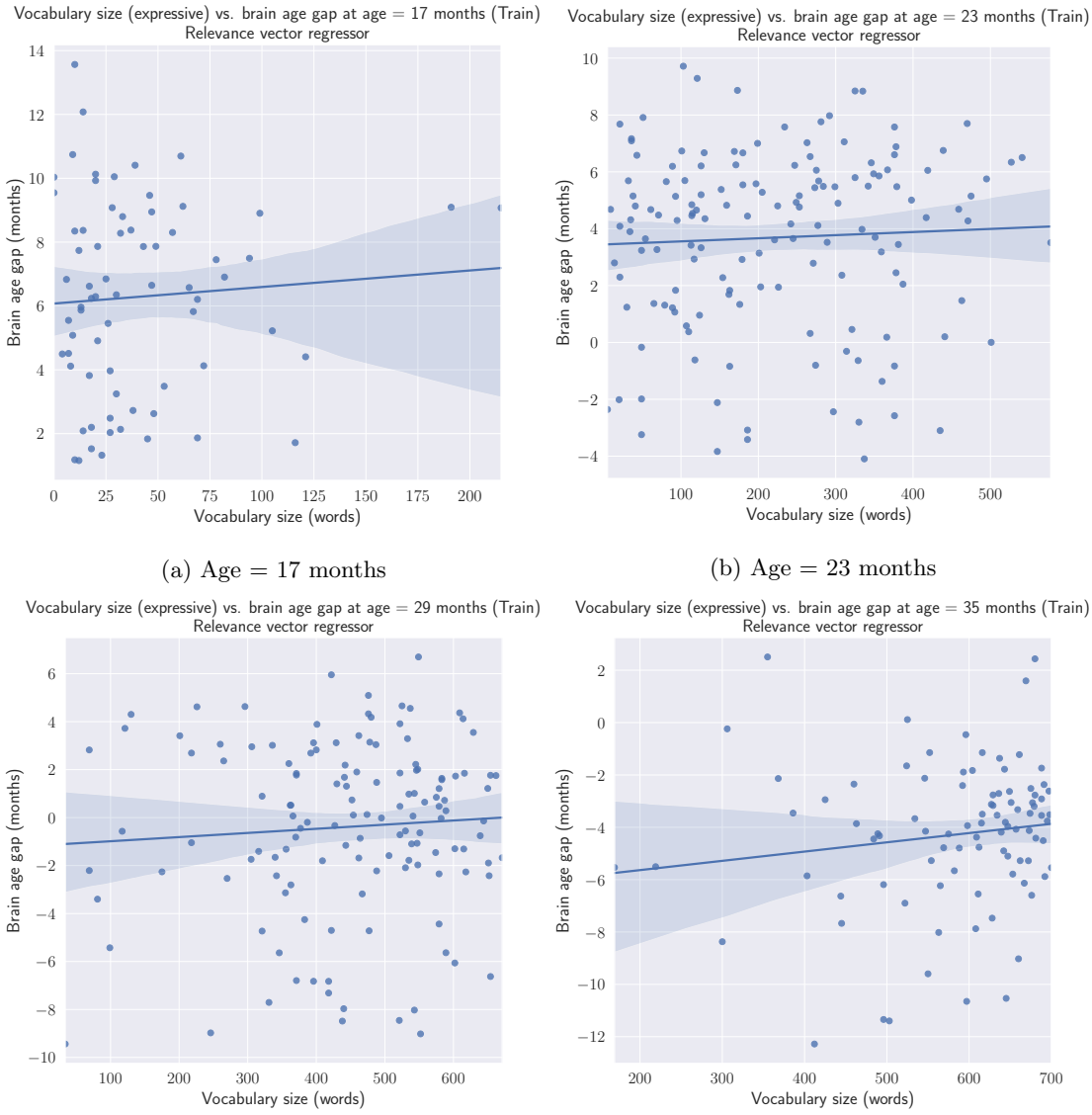

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 51: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the relevance vector regressor's brain age predictions on the train set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 52: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the relevance vector regressor's brain age predictions on the train set.

## Fully-connected neural network (traditional machine learning)

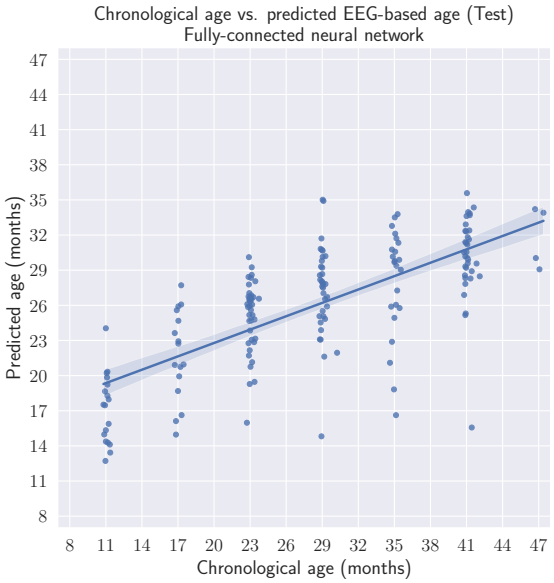



Figure 53: The predictions of the fully-connected neural network compared to the true chronological ages of the subject on the test set.

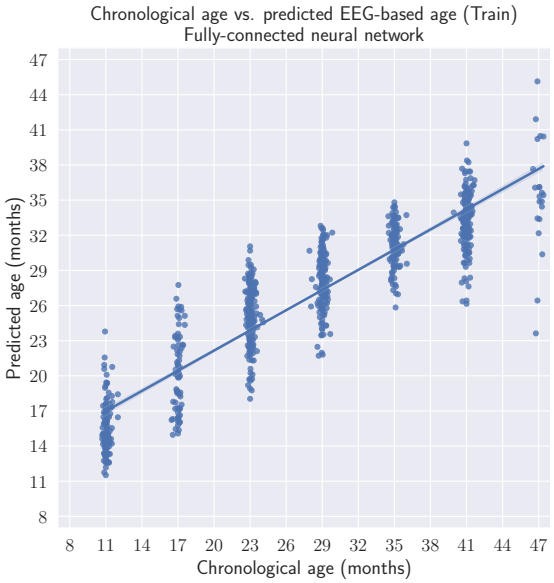



Figure 54: The predictions of the fully-connected neural network compared to the true chronological ages of the subject on the train set.

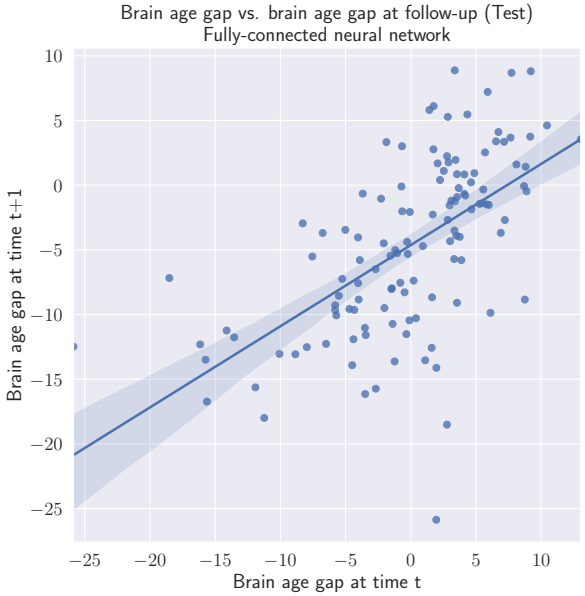



Figure 55: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the fully-connected neural network on the test set.

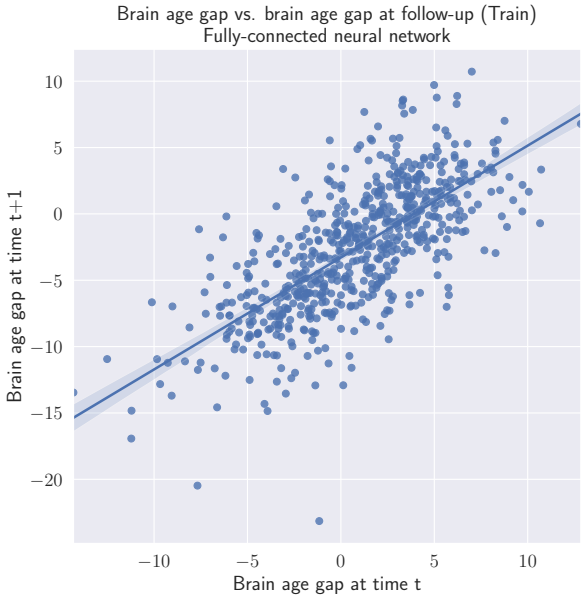
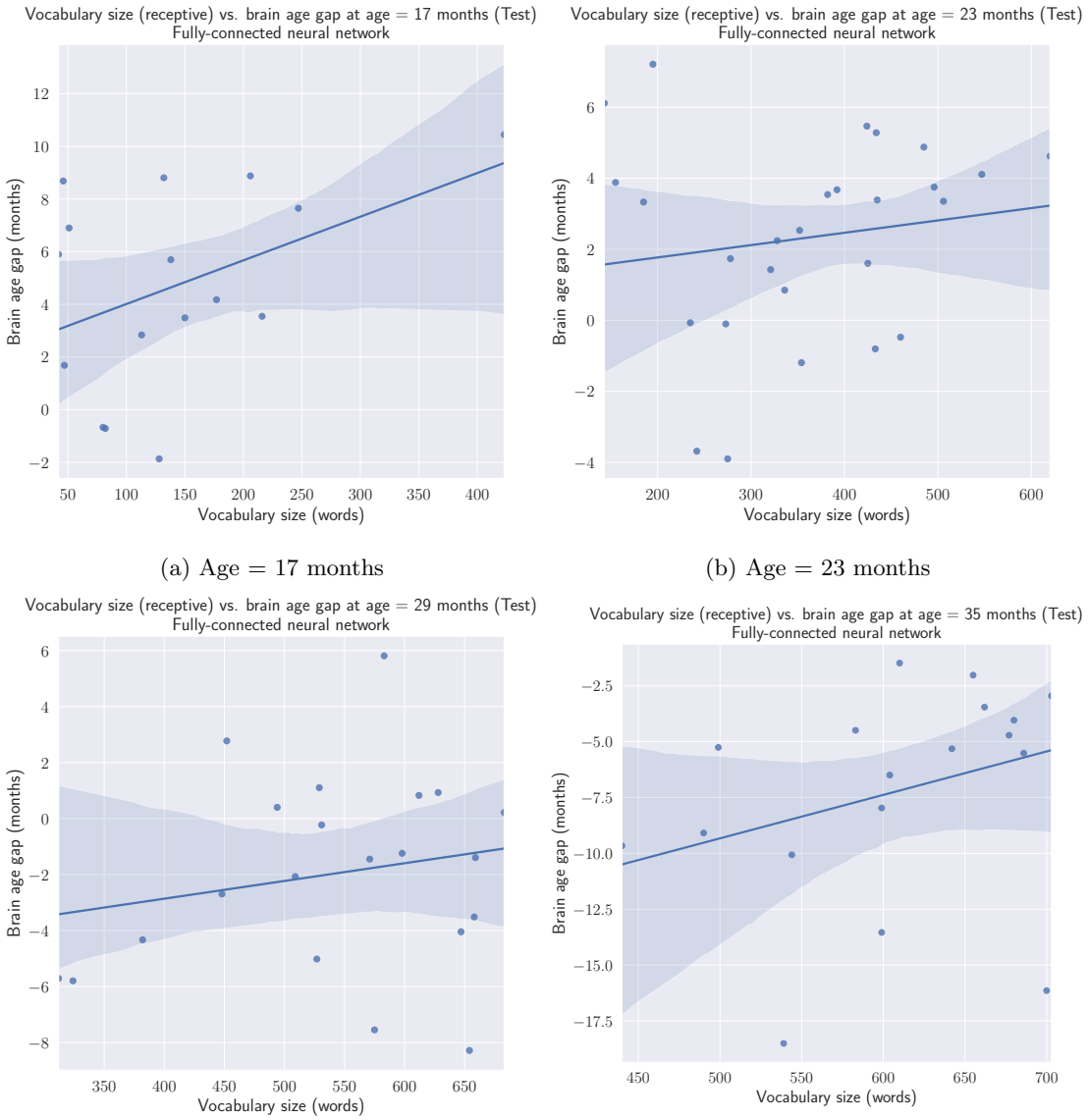



Figure 56: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the fully-connected neural network on the train set.



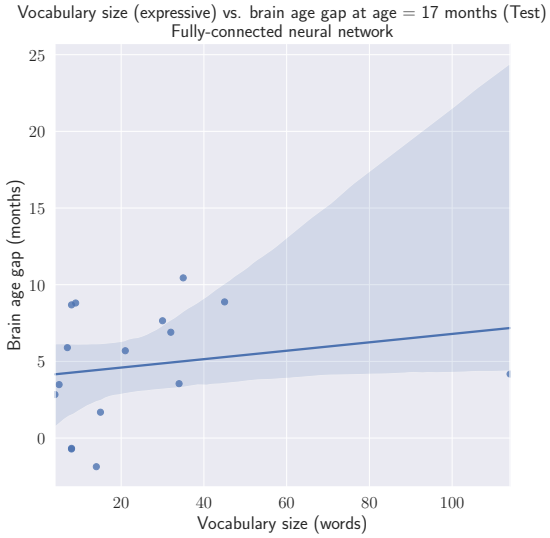

(a) Age = 17 months

(b) Age = 23 months

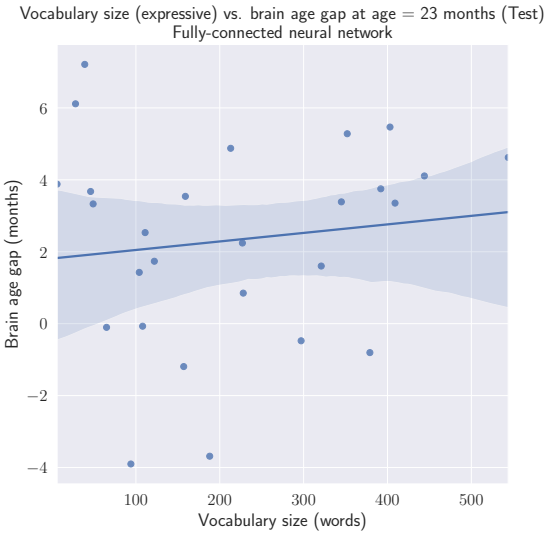
(c) Age = 35 months

(d) Age = 35 months

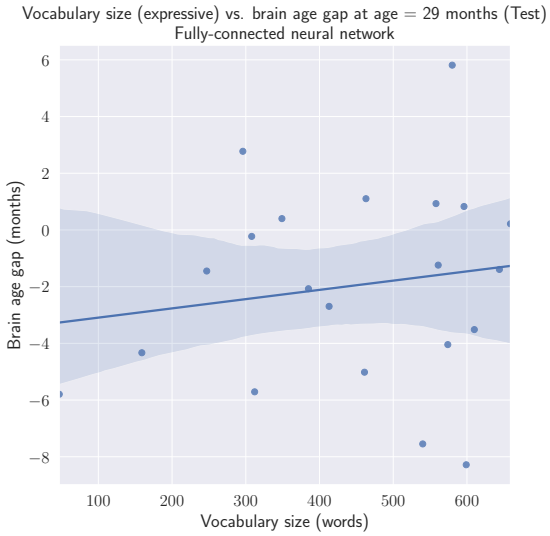
Figure 57: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the test set.



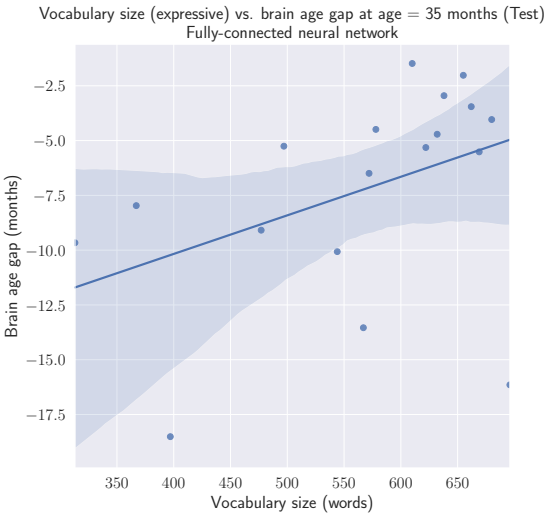

(a) Age = 17 months




(b) Age = 23 months

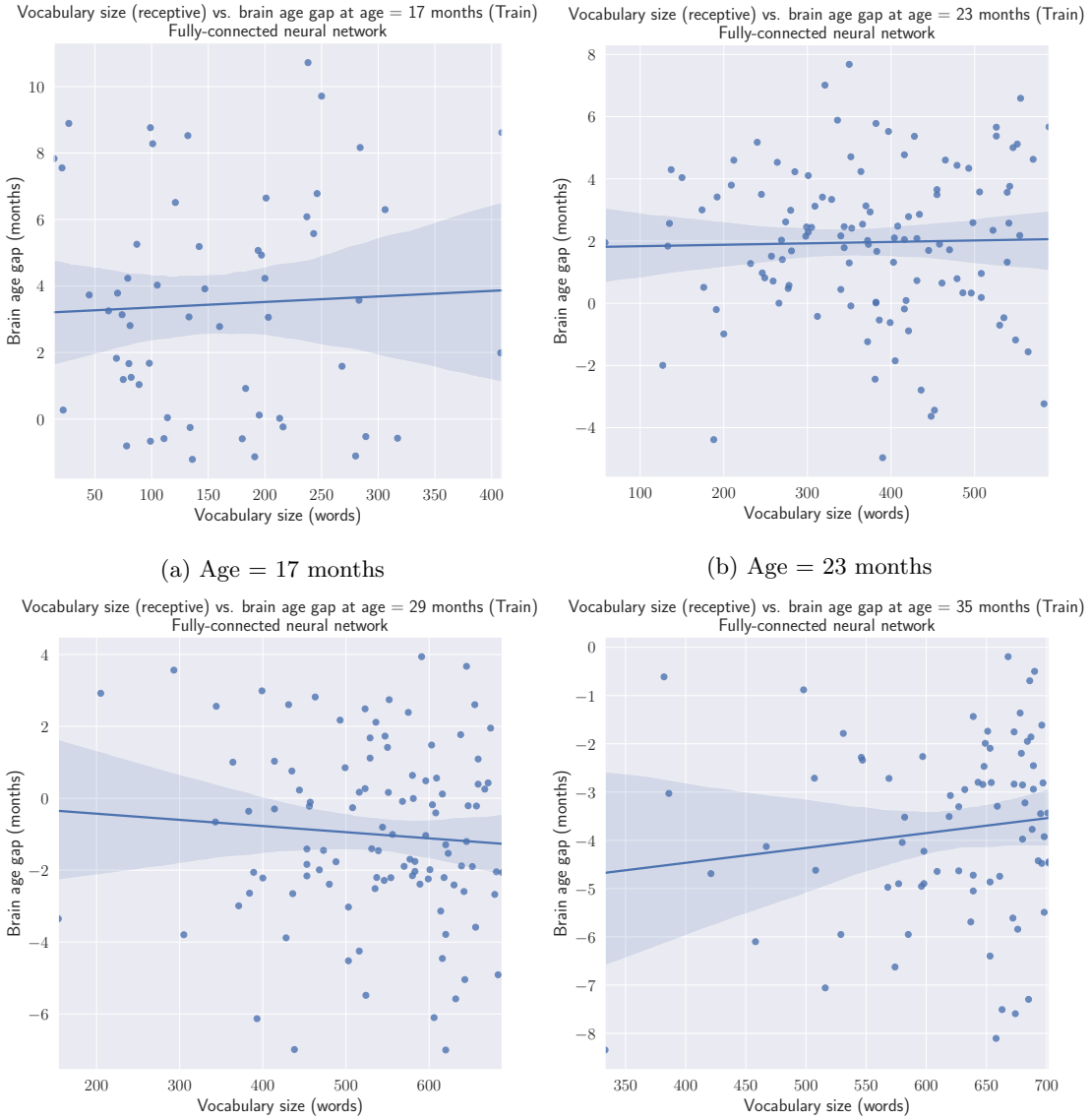



(c) Age = 29 months




(d) Age = 35 months

Figure 58: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the test set.



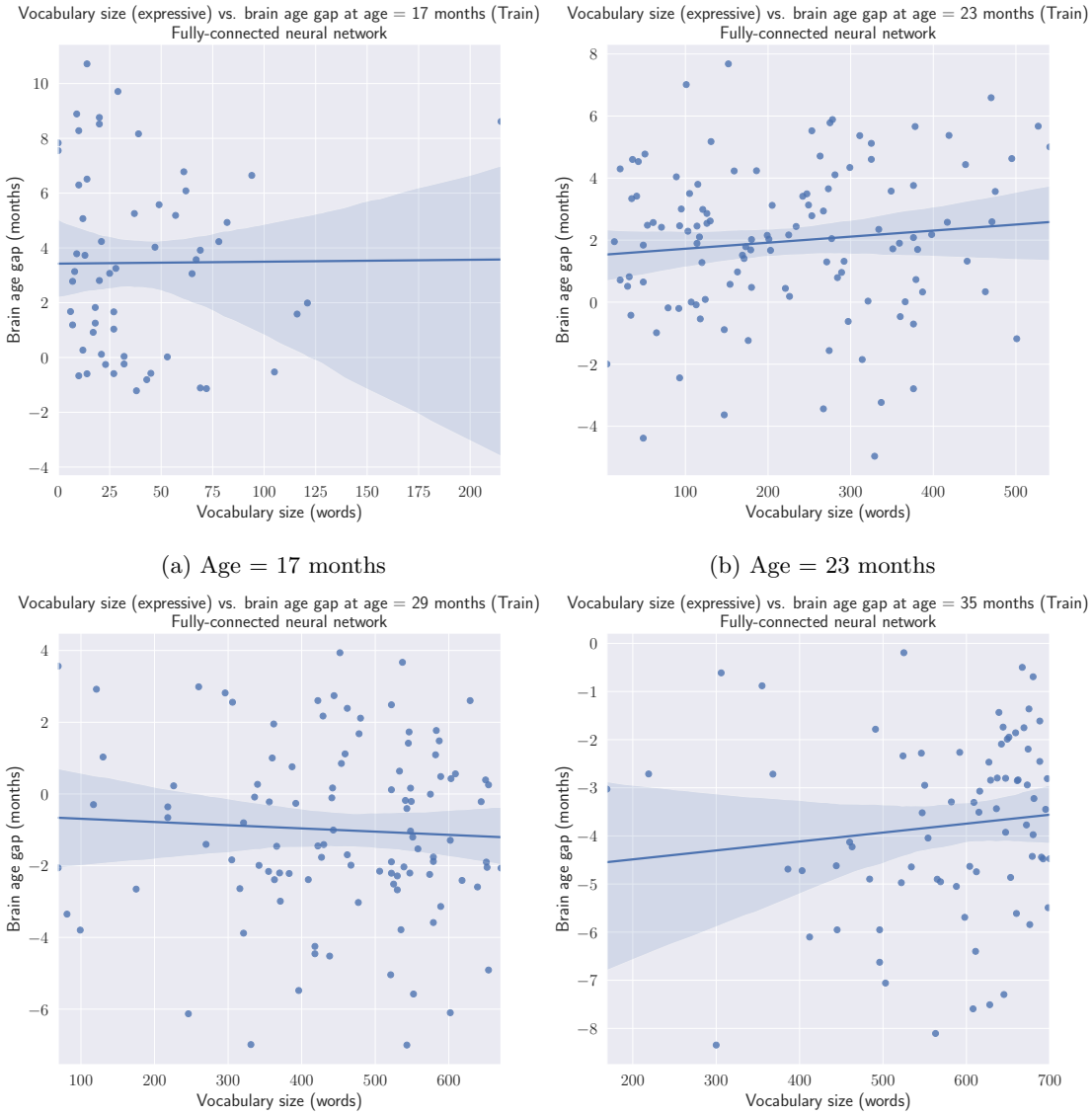

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 59: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the train set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 60: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the train set.

## Fully-connected neural network (deep learning)

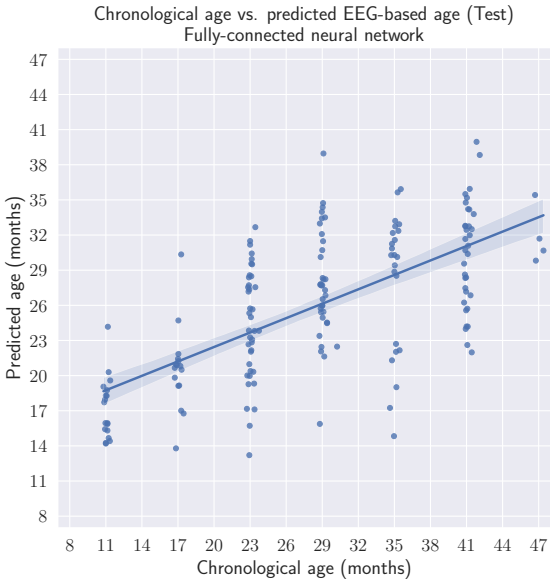



Figure 61: The predictions of the fully-connected neural network compared to the true chronological ages of the subject on the test set.

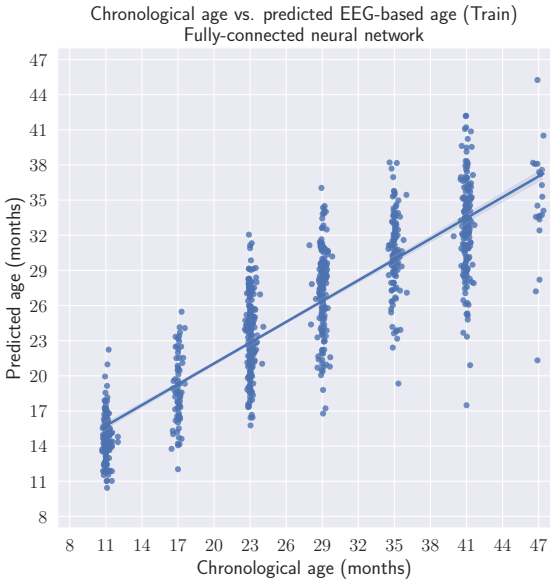



Figure 62: The predictions of the fully-connected neural network compared to the true chronological ages of the subject on the train set.

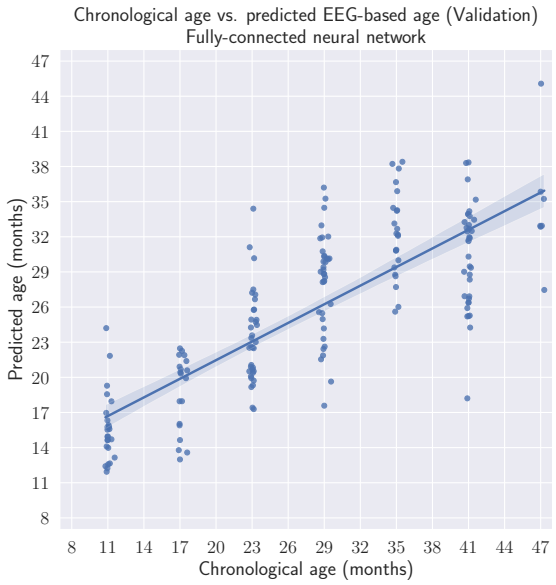



Figure 63: The predictions of the fully-connected neural network compared to the true chronological ages of the subject on the validation set.

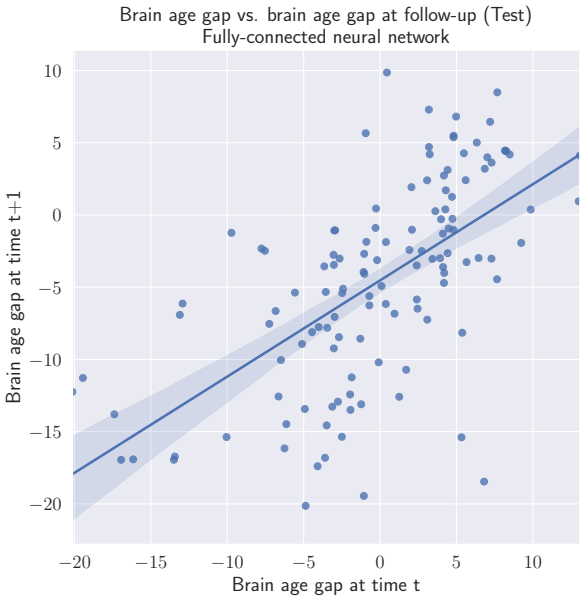



Figure 64: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the fully-connected neural network on the test set.

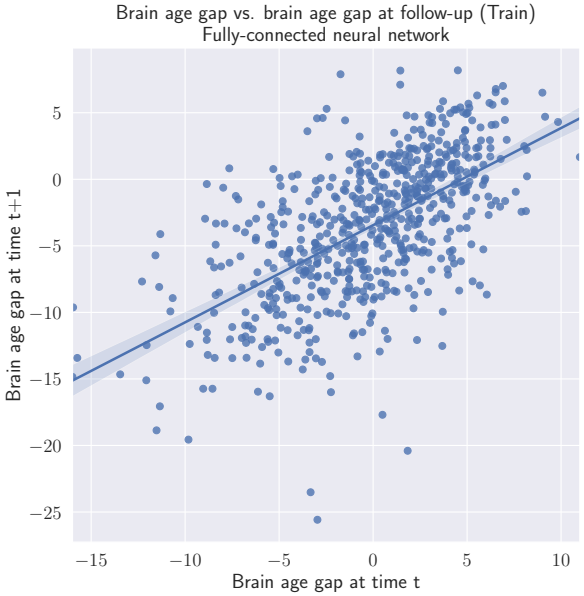
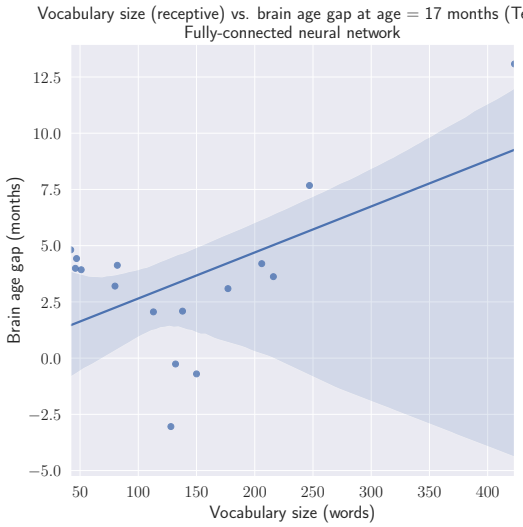
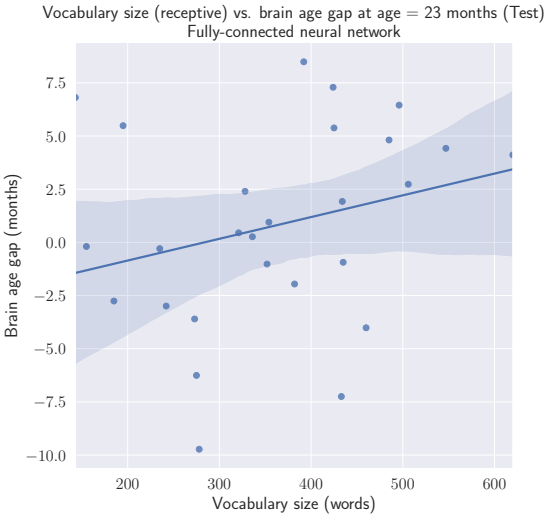



Figure 65: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the fully-connected neural network on the train set.


Figure 66: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the fully-connected neural network on the validation set.

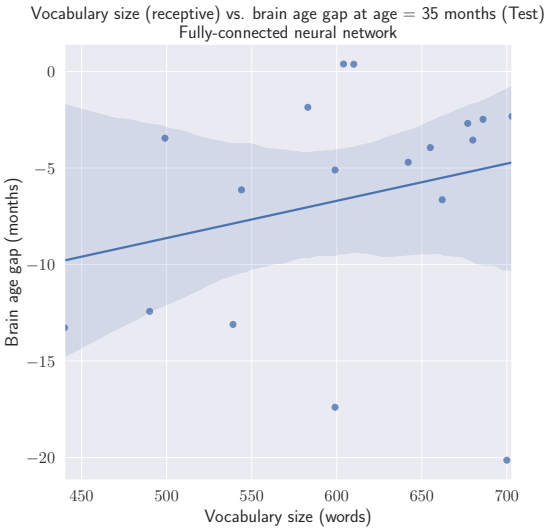



(a) Age = 17 months



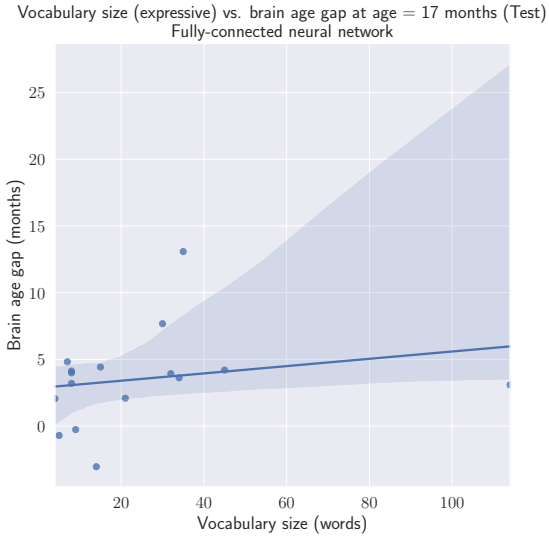

(b) Age = 23 months


(c) Age = 35 months

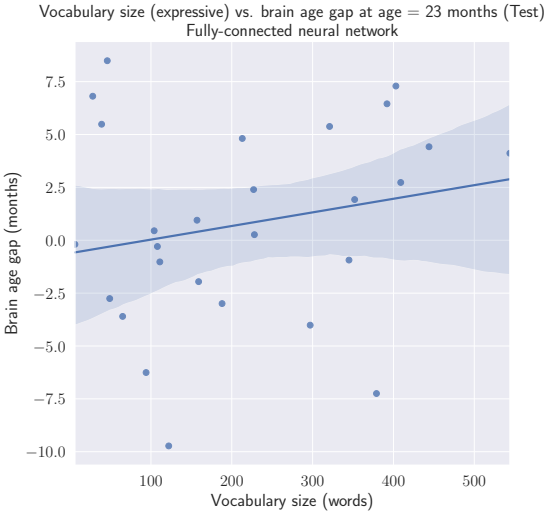



(d) Age = 35 months

Figure 67: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the test set.

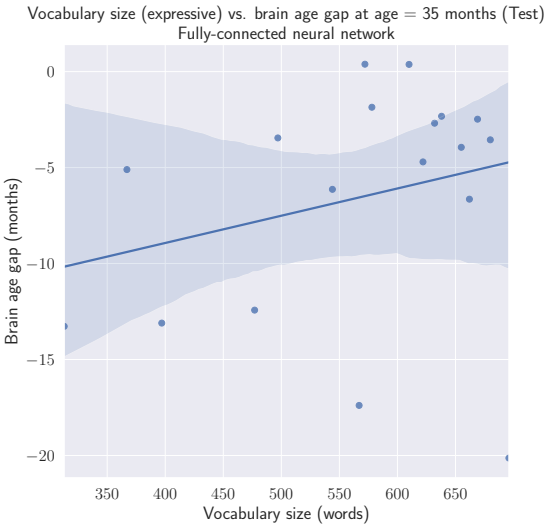



(a) Age = 17 months



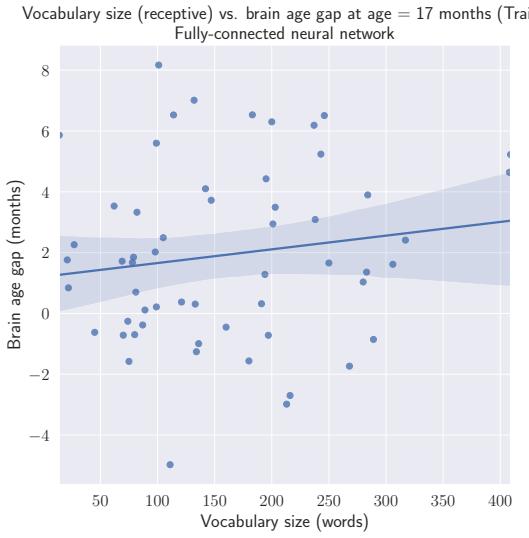

(b) Age = 23 months


(c) Age = 29 months

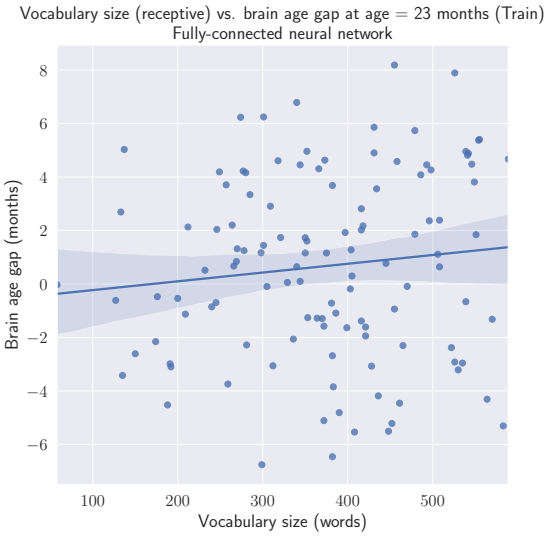



(d) Age = 35 months

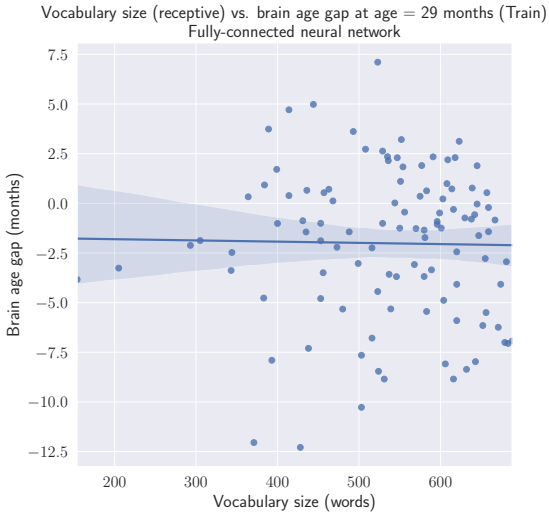
Figure 68: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the test set.



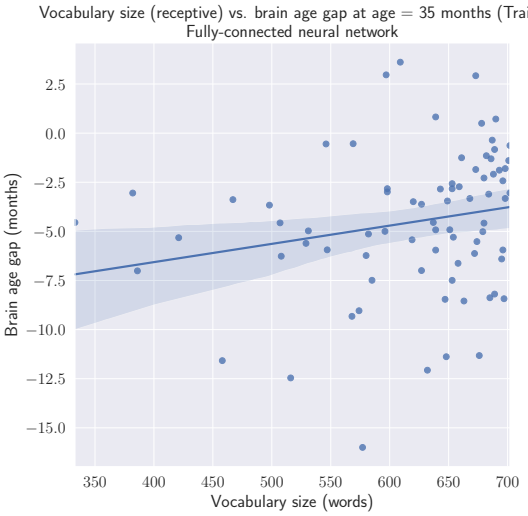

(a) Age = 17 months




(b) Age = 23 months

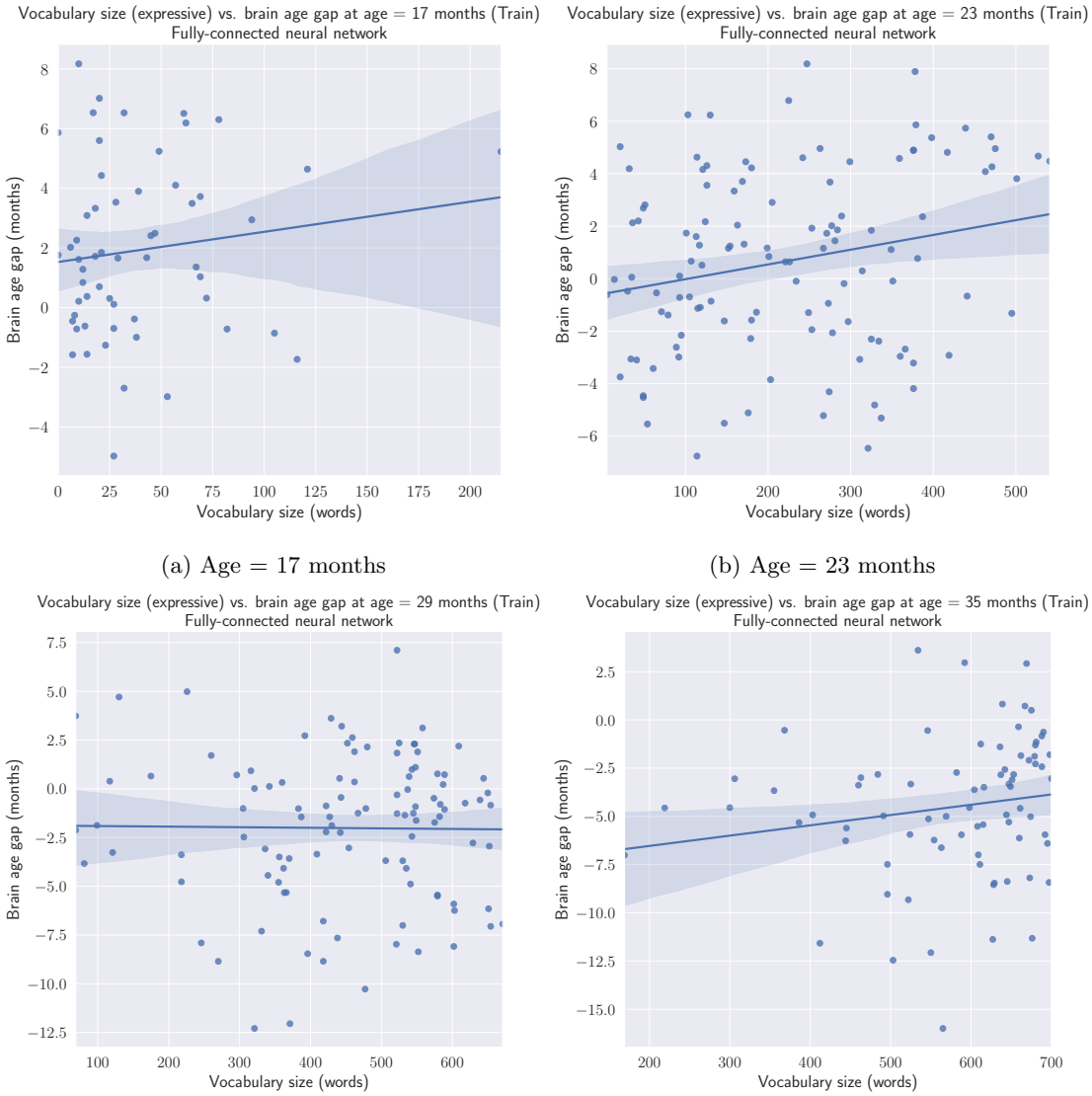



(c) Age = 35 months




(d) Age = 35 months

Figure 69: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the train set.



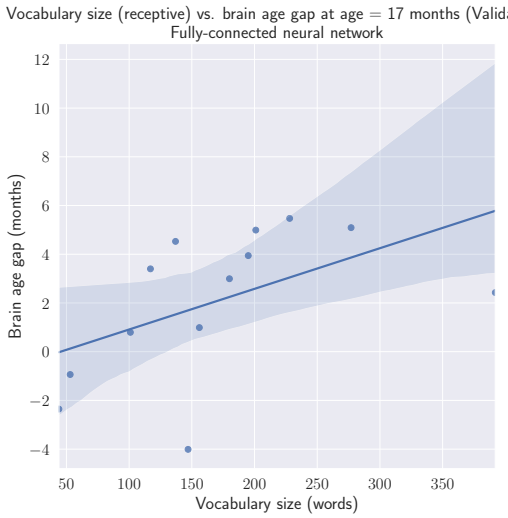

(a) Age = 17 months

(b) Age = 23 months

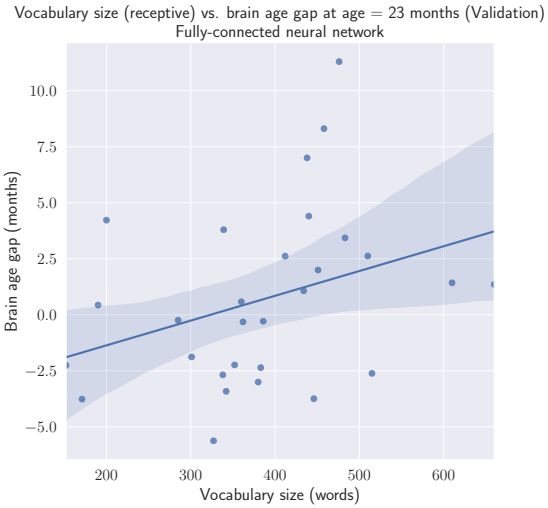
(c) Age = 29 months

(d) Age = 35 months

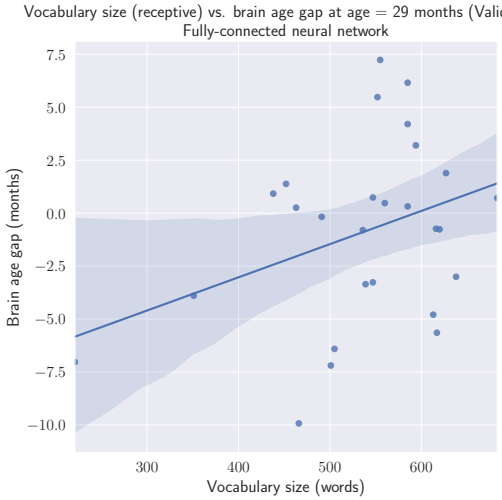
Figure 70: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the train set.



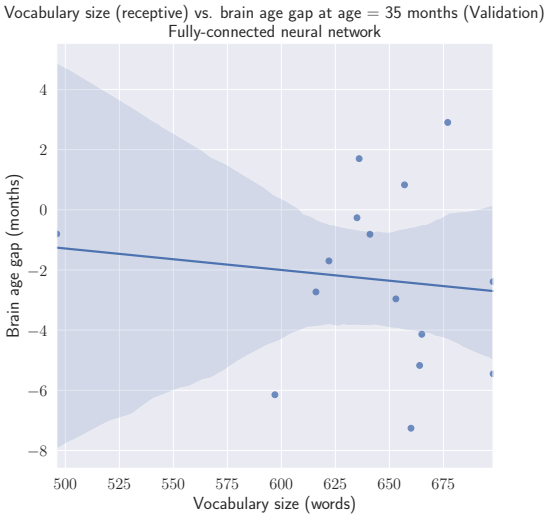

(a) Age = 17 months




(b) Age = 23 months

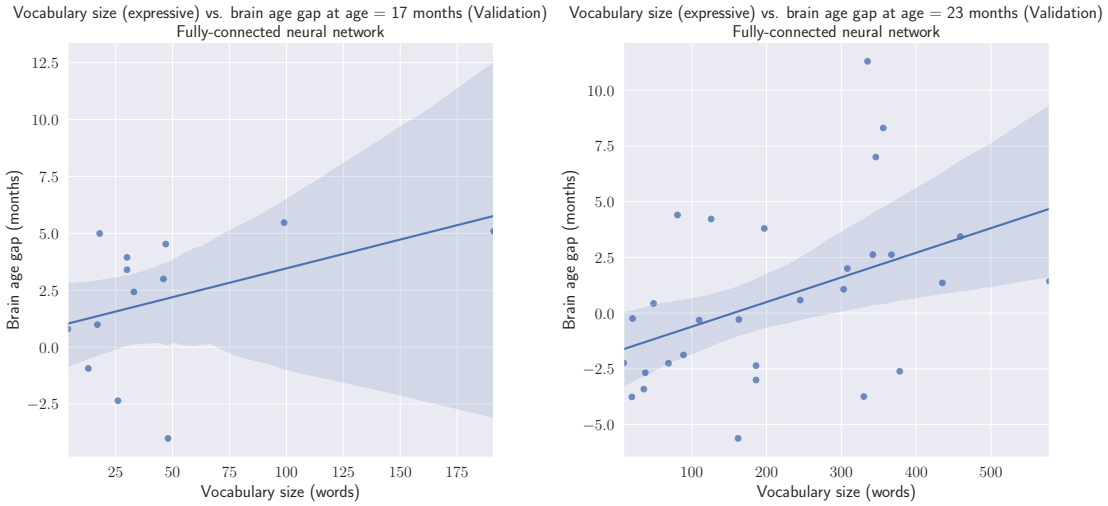



(c) Age = 35 months



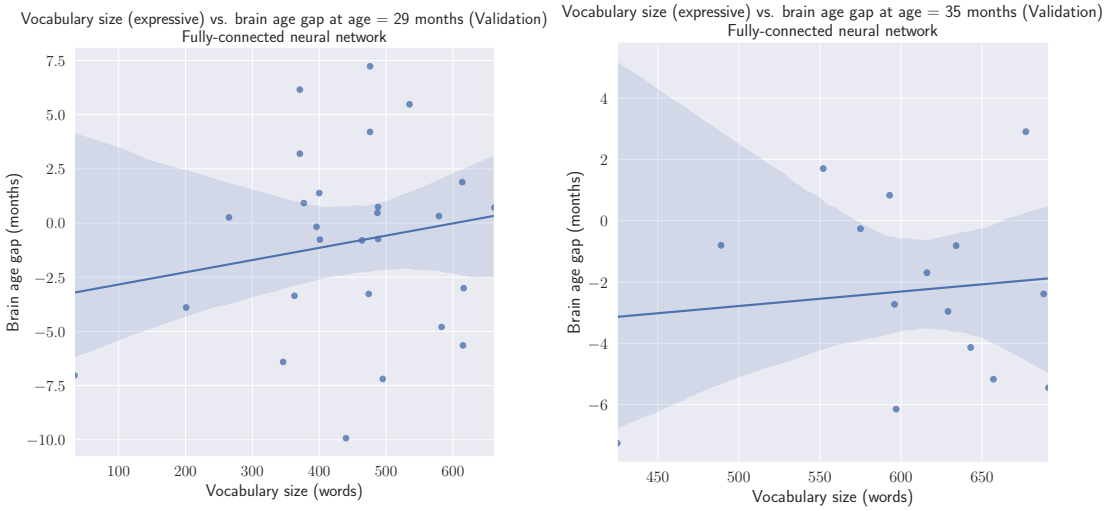

(d) Age = 35 months

Figure 71: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months




(c) Age = 29 months

(d) Age = 35 months

Figure 72: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the fully-connected neural network's brain age predictions on the validation set.

## Convolutional neural network

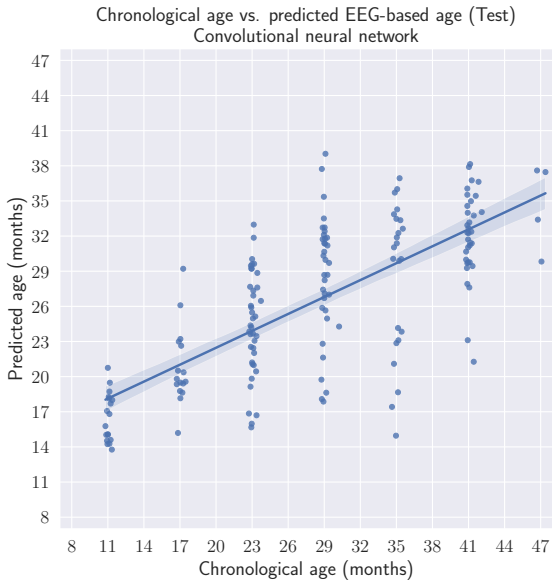



Figure 73: The predictions of the convolutional neural network compared to the true chronological ages of the subject on the test set.

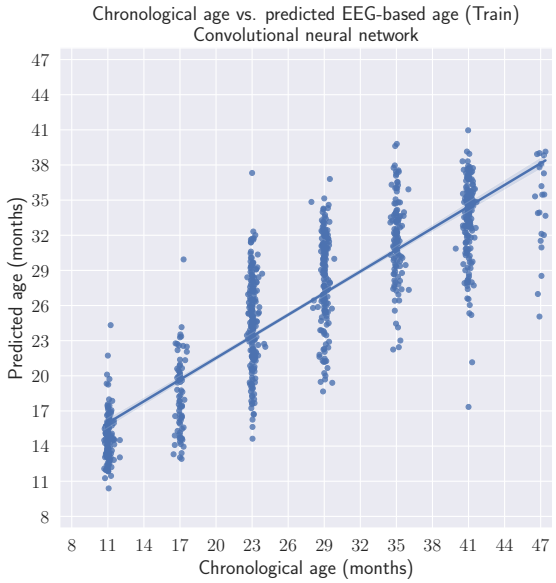



Figure 74: The predictions of the convolutional neural network compared to the true chronological ages of the subject on the train set.

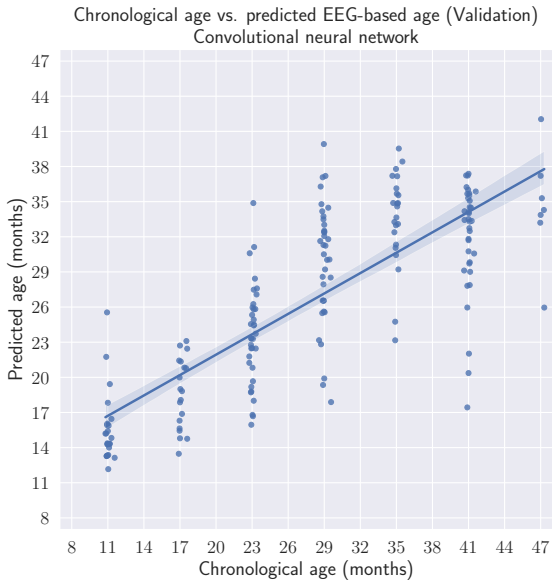



Figure 75: The predictions of the convolutional neural network compared to the true chronological ages of the subject on the validation set.

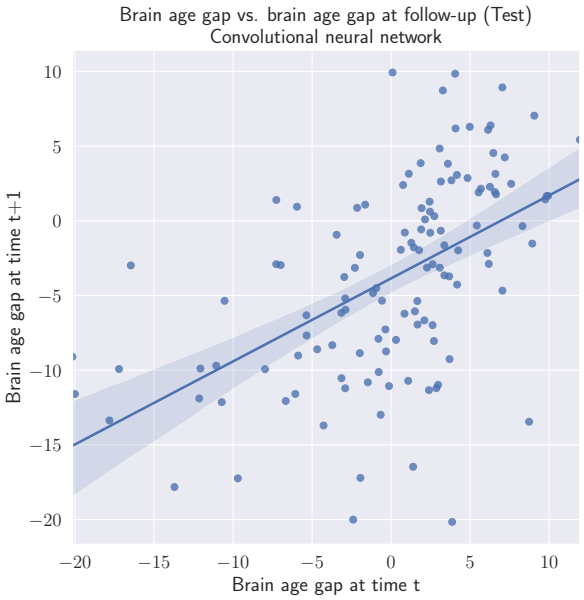



Figure 76: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the convolutional neural network on the test set.

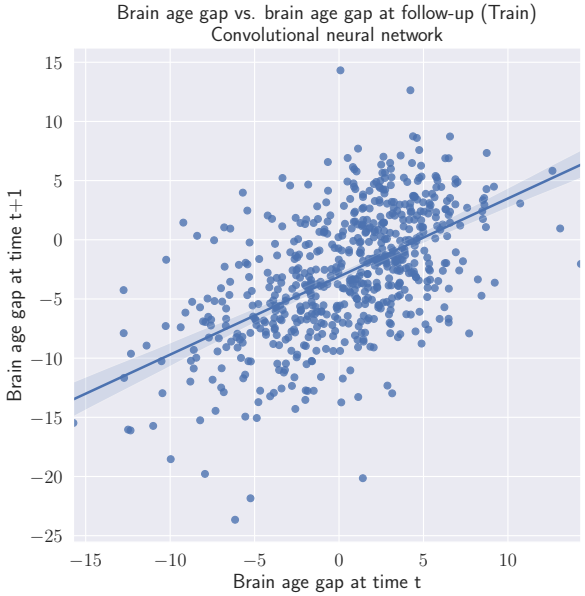



Figure 77: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the convolutional neural network on the train set.

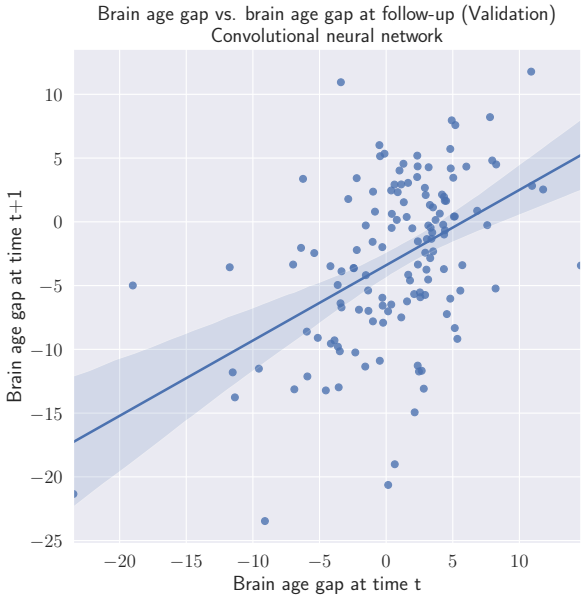
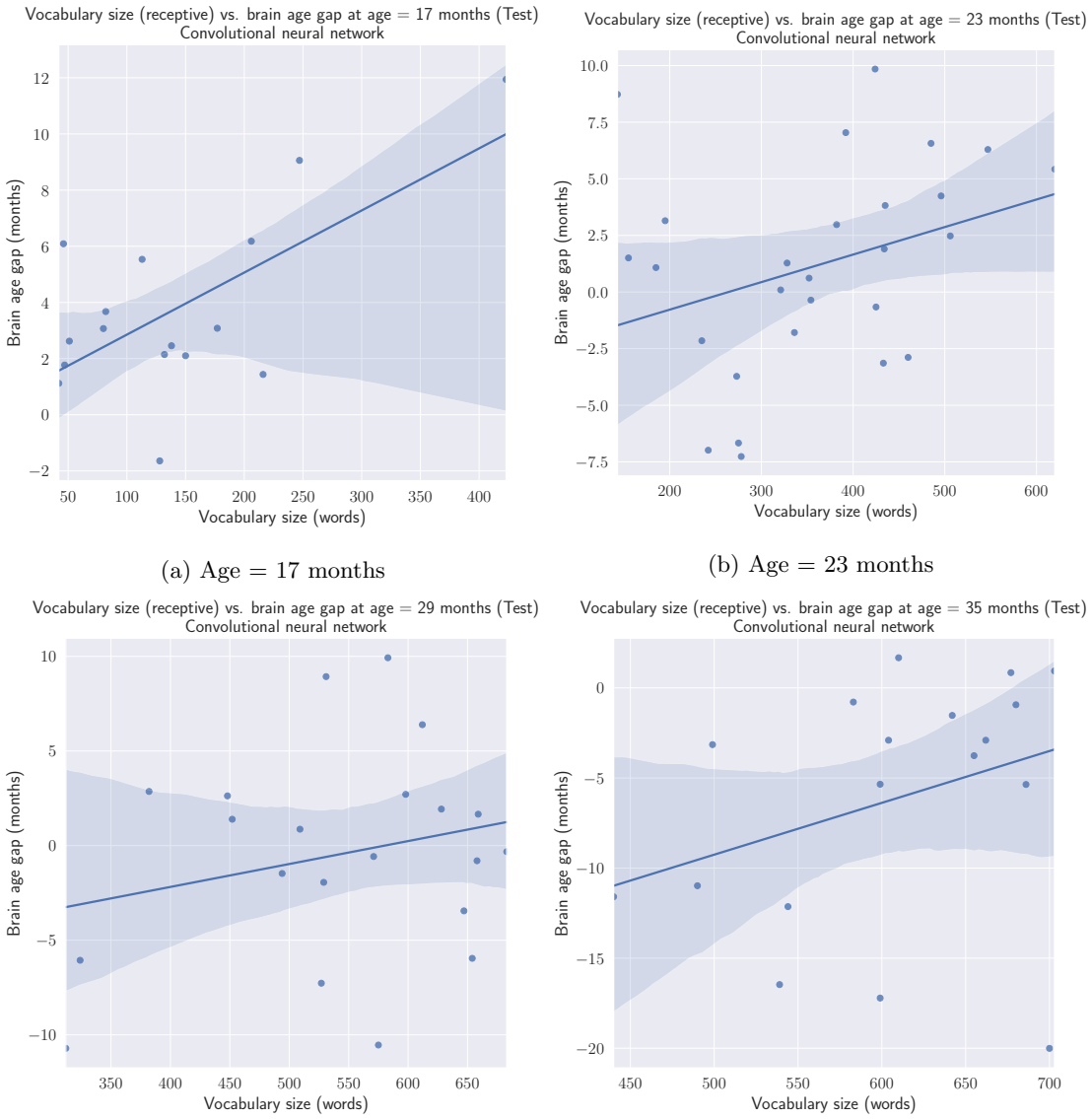



Figure 78: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the convolutional neural network on the validation set.



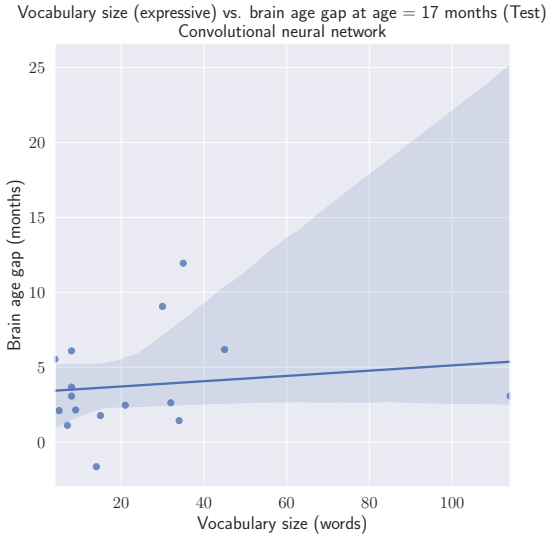

(a) Age = 17 months

(b) Age = 23 months

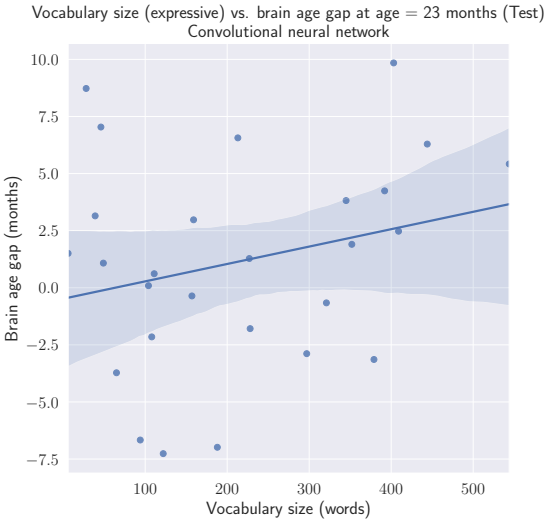
(c) Age = 35 months

(d) Age = 35 months

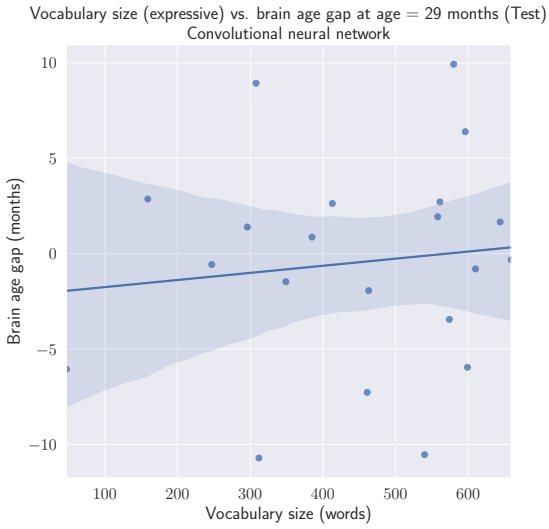
Figure 79: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the test set.



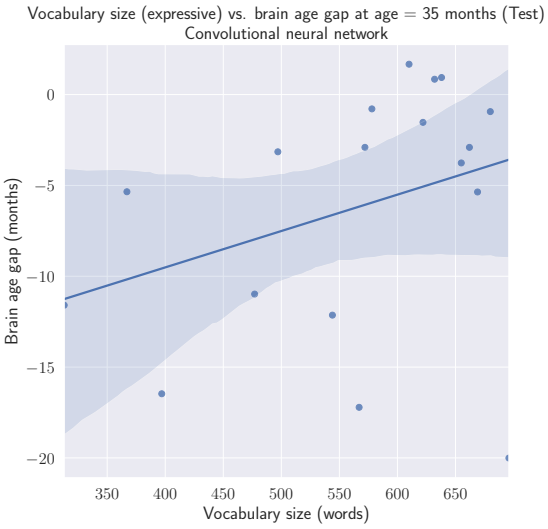

(a) Age = 17 months




(b) Age = 23 months

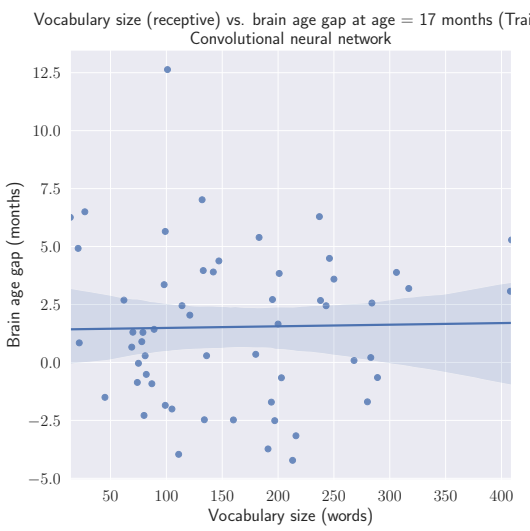



(c) Age = 29 months

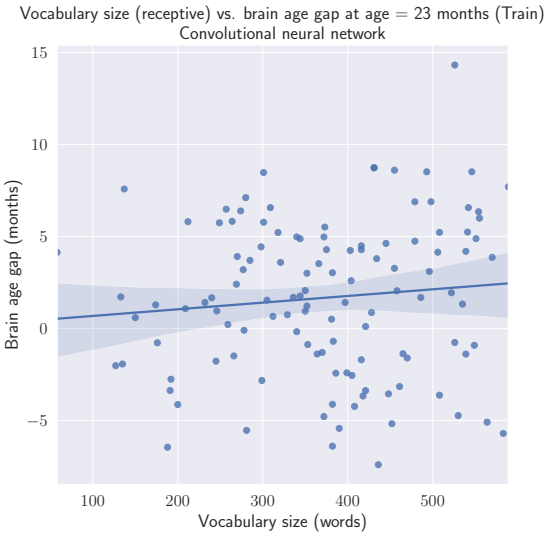



(d) Age = 35 months

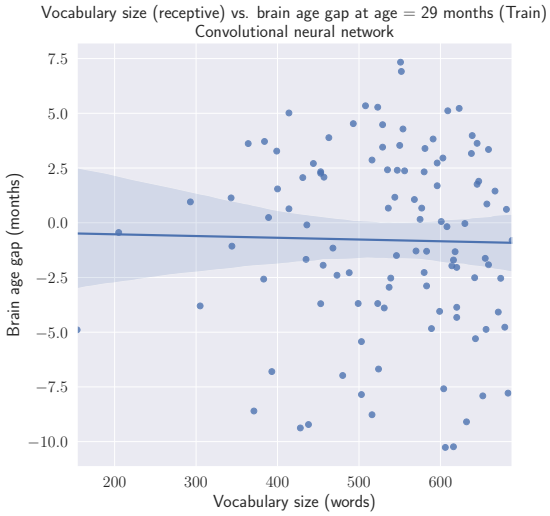
Figure 80: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the test set.



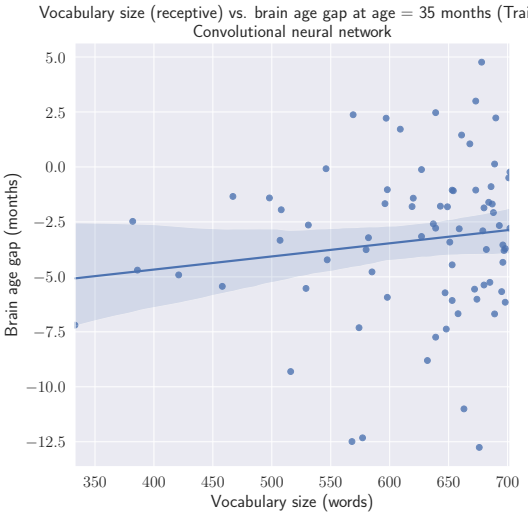

(a) Age = 17 months




(b) Age = 23 months

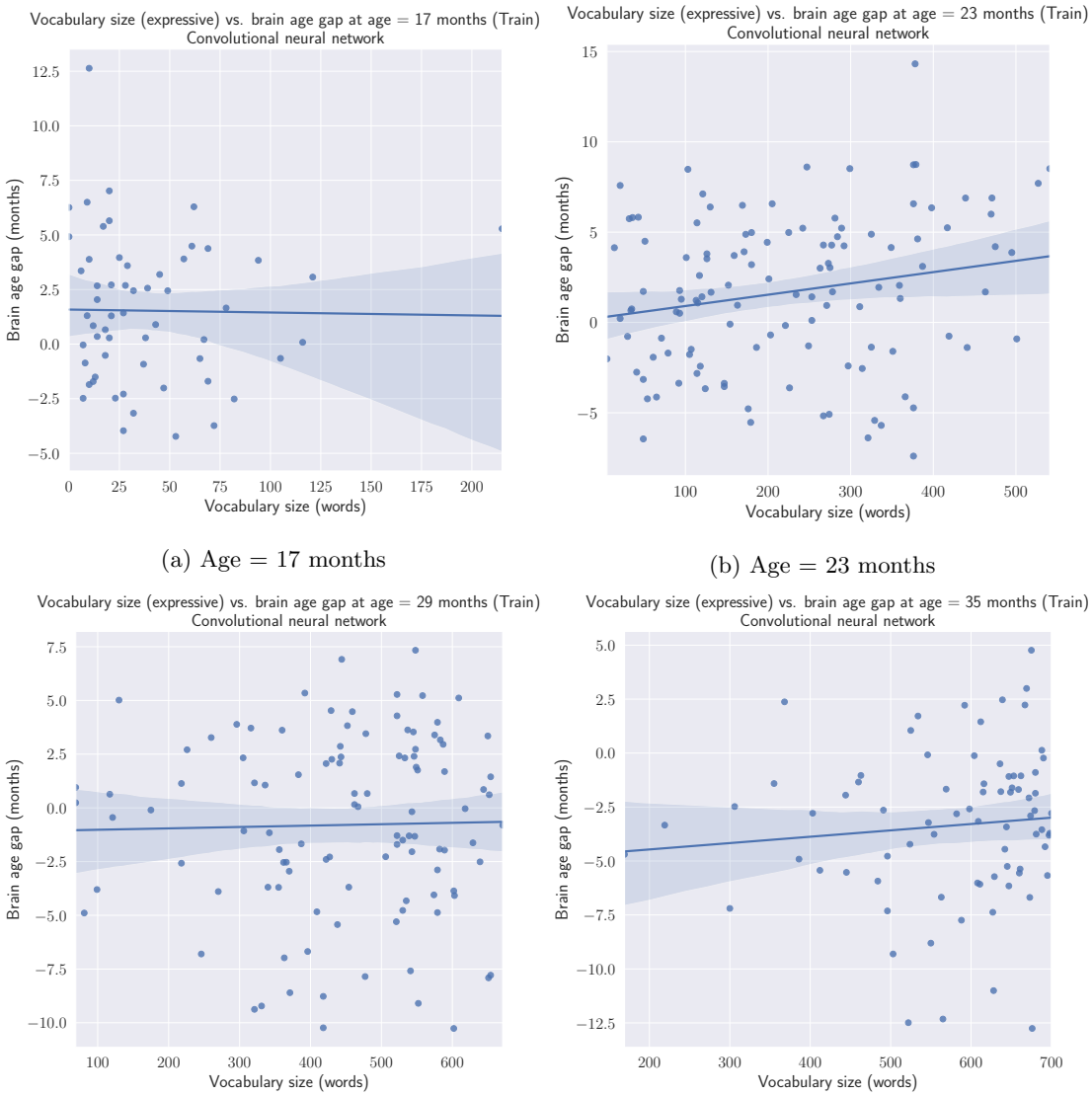



(c) Age = 35 months




(d) Age = 35 months

Figure 81: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the train set.



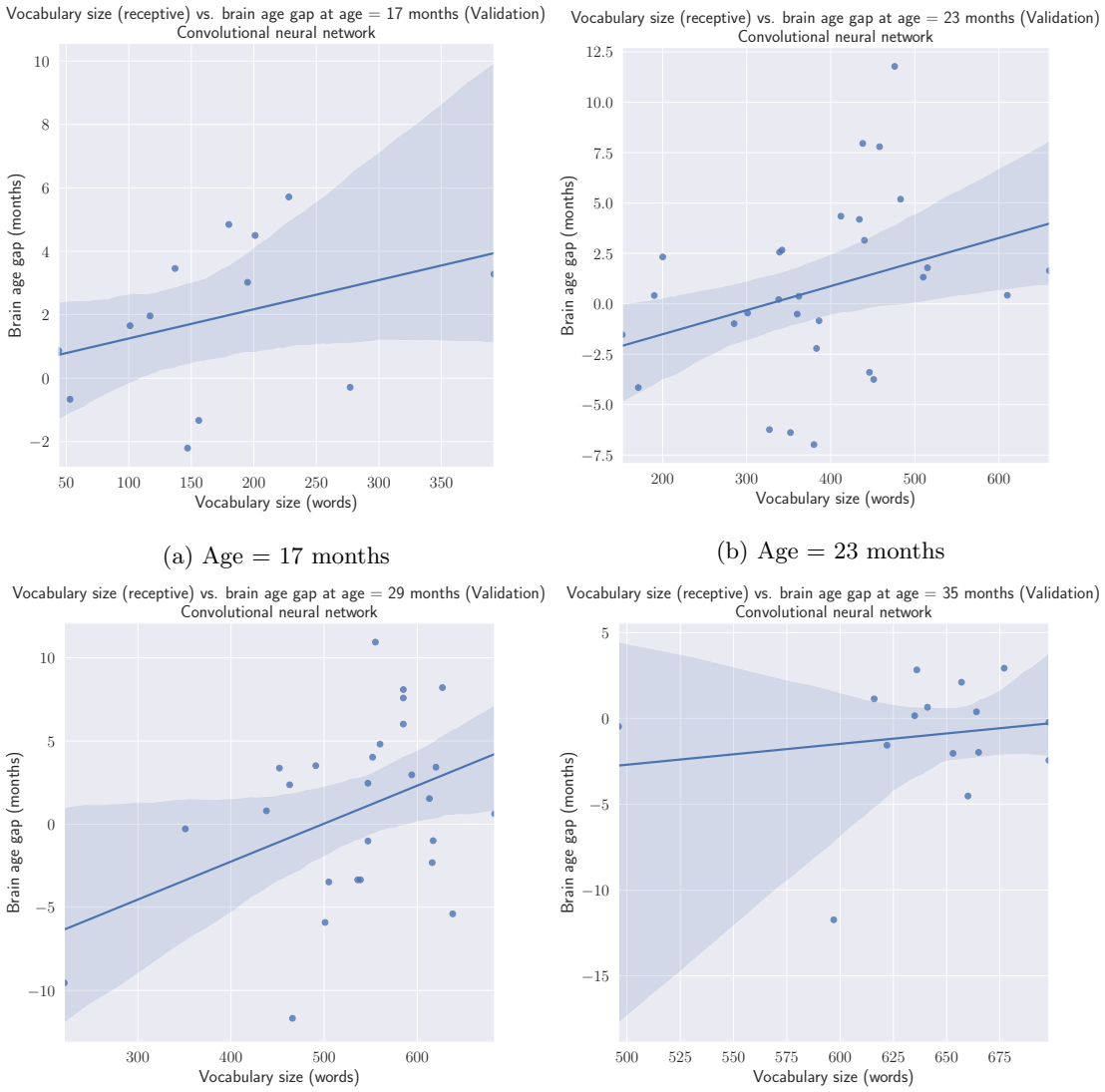

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 82: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the train set.



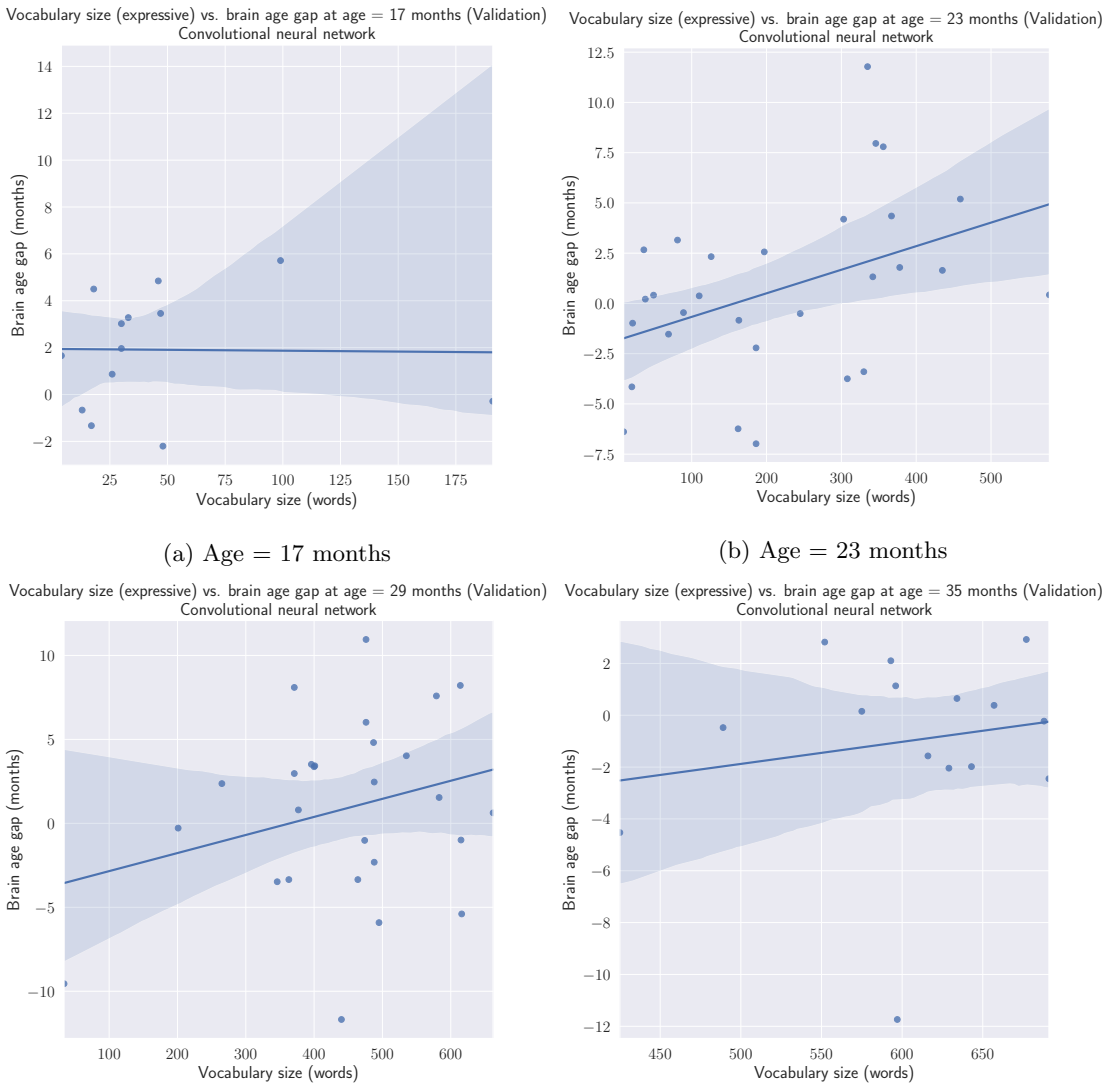

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 83: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 84: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the convolutional neural network's brain age predictions on the validation set.

## ResNet

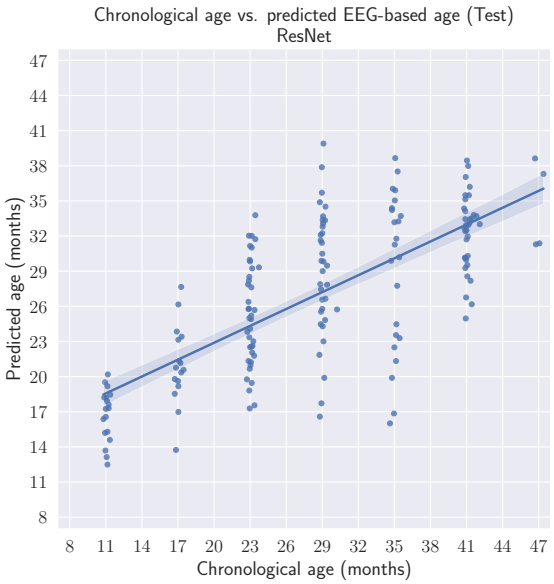



Figure 85: The predictions of the ResNet compared to the true chronological ages of the subject on the test set.

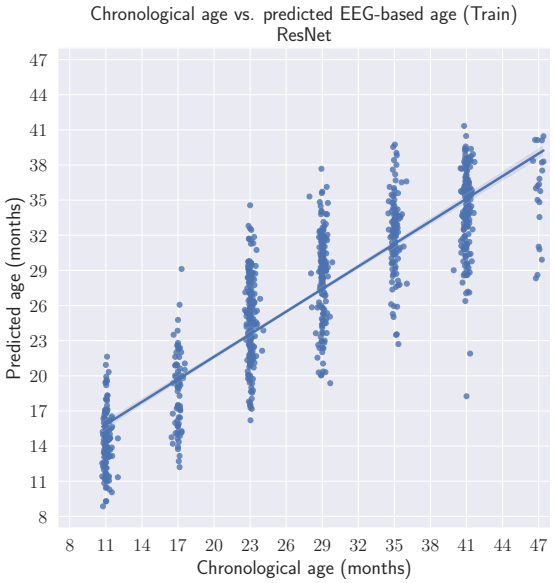



Figure 86: The predictions of the ResNet compared to the true chronological ages of the subject on the train set.

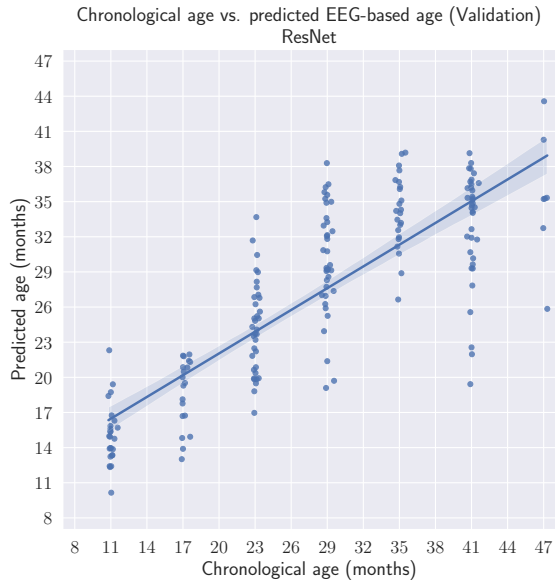



Figure 87: The predictions of the ResNet compared to the true chronological ages of the subject on the validation set.

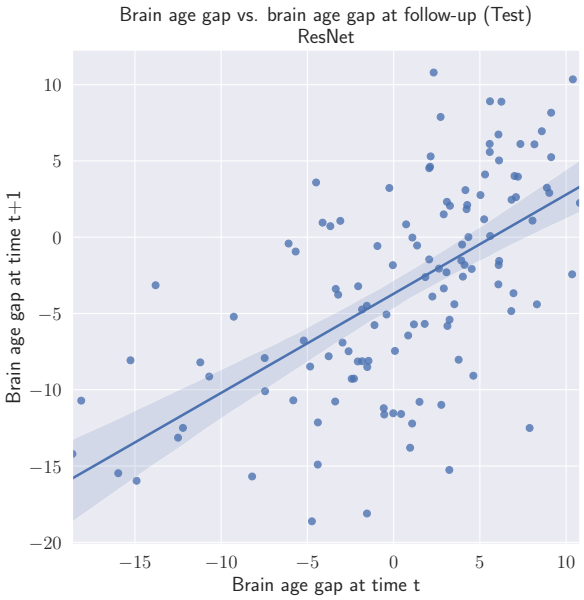



Figure 88: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the ResNet on the test set.

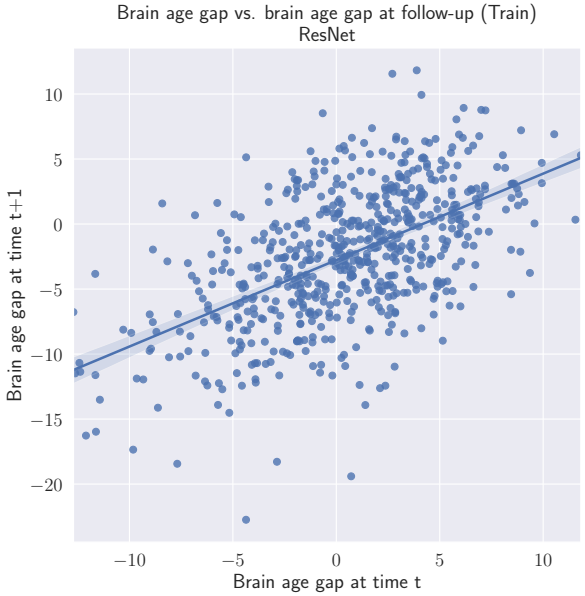



Figure 89: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the ResNet on the train set.

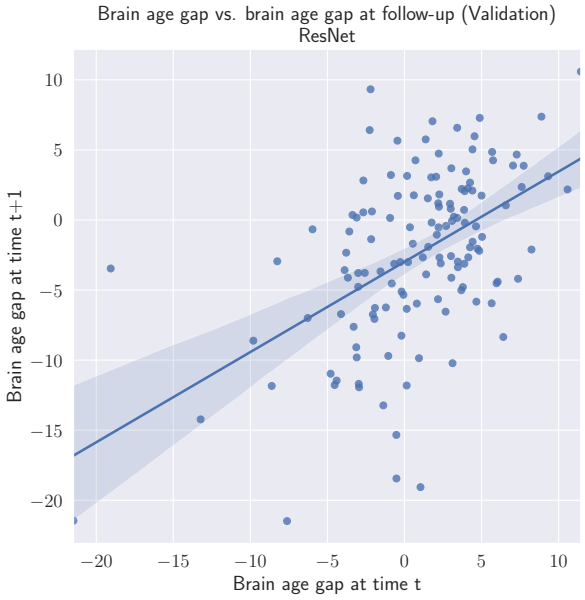
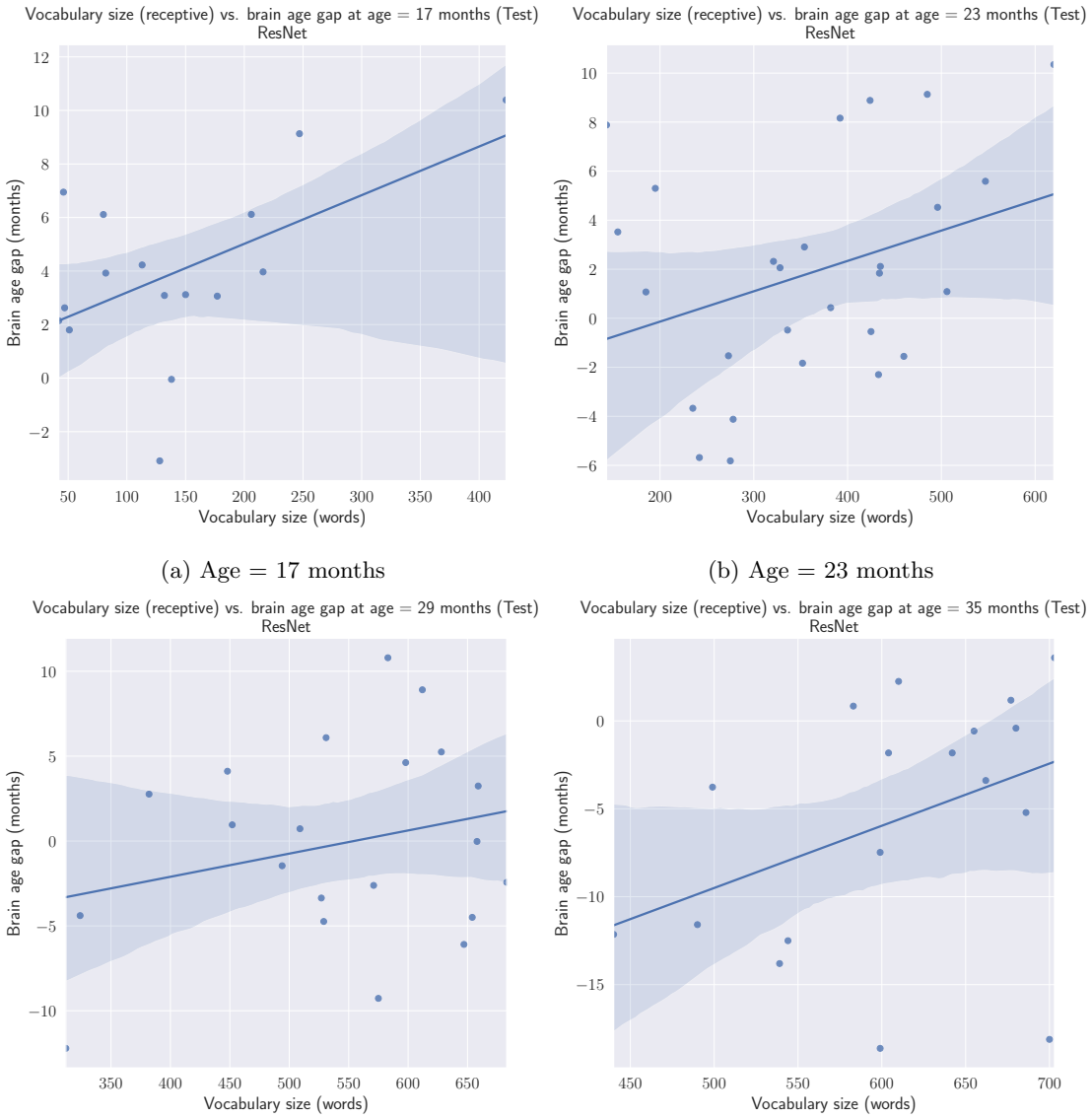



Figure 90: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the ResNet on the validation set.



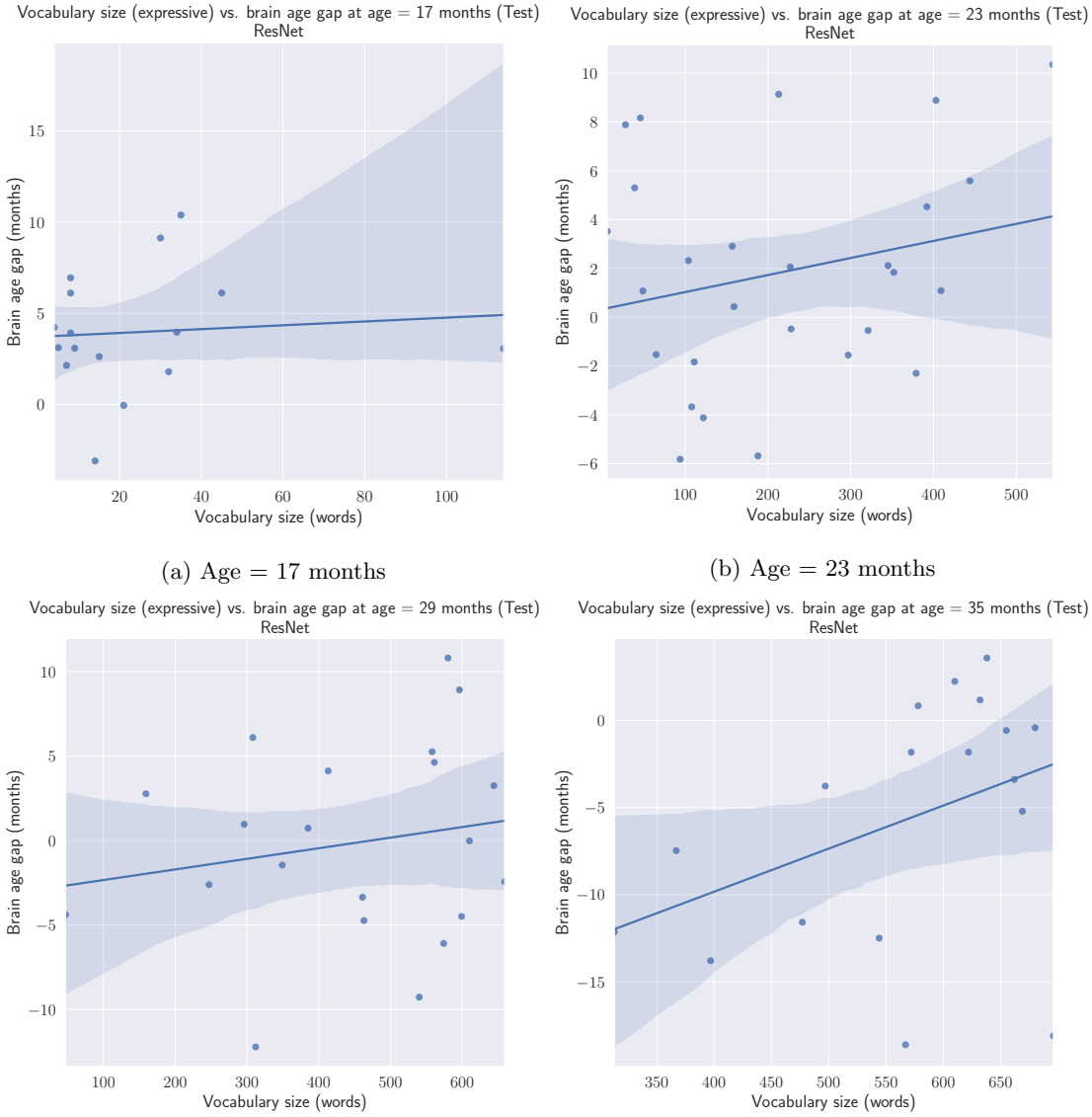

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 91: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the test set.



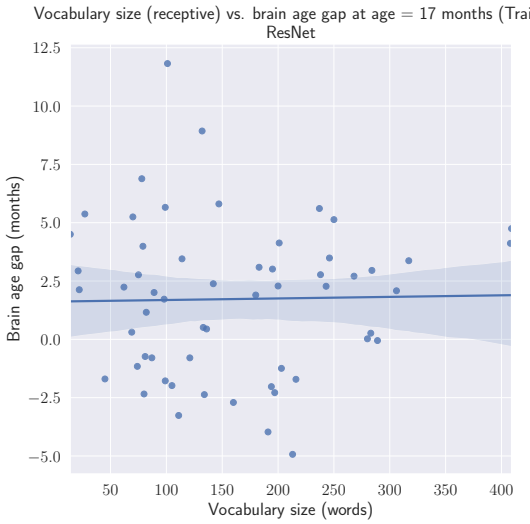

(a) Age = 17 months

(b) Age = 23 months

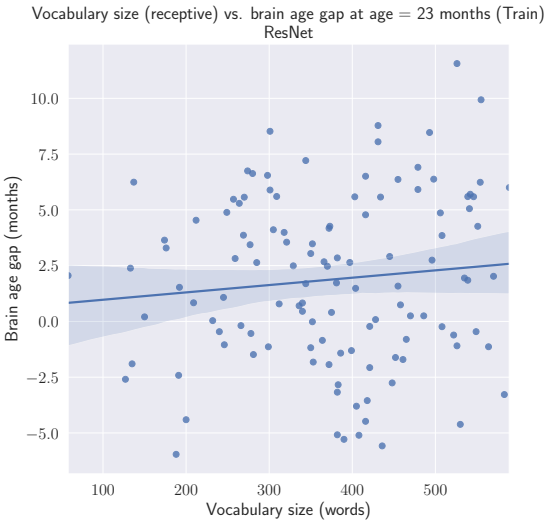
(c) Age = 29 months

(d) Age = 35 months

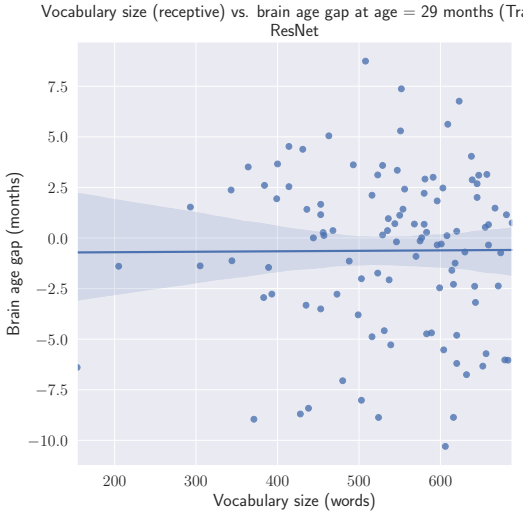
Figure 92: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the test set.



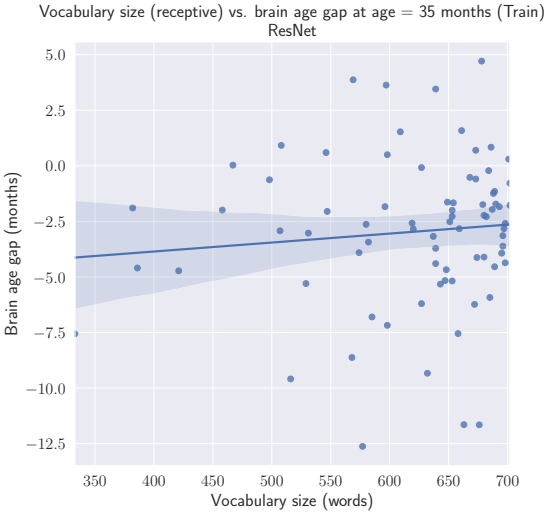

(a) Age = 17 months




(b) Age = 23 months

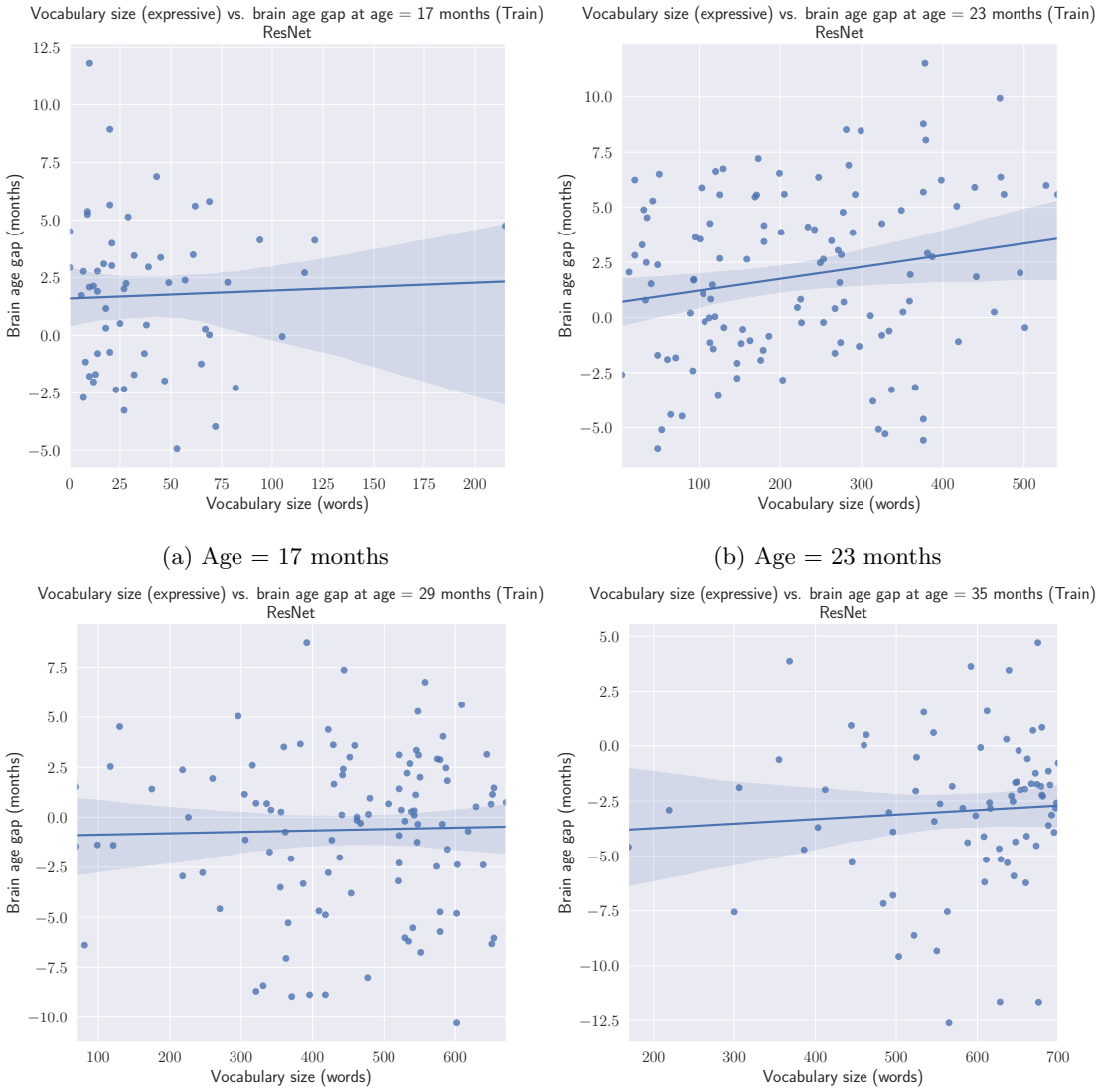



(c) Age = 35 months




(d) Age = 35 months

Figure 93: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the train set.



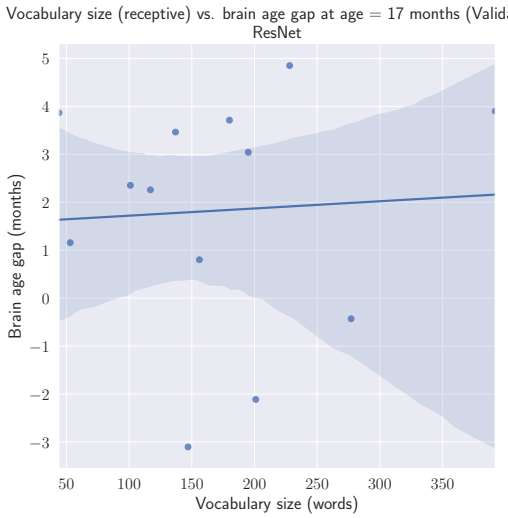

(a) Age = 17 months

(b) Age = 23 months

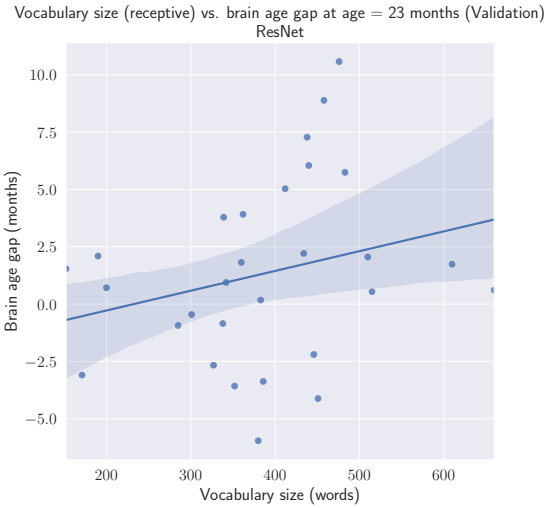
(c) Age = 29 months

(d) Age = 35 months

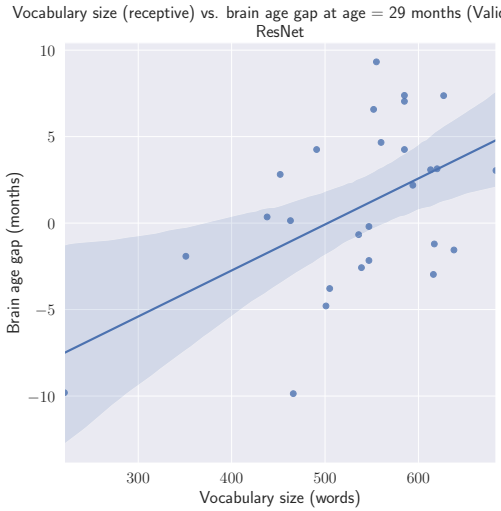
Figure 94: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the train set.



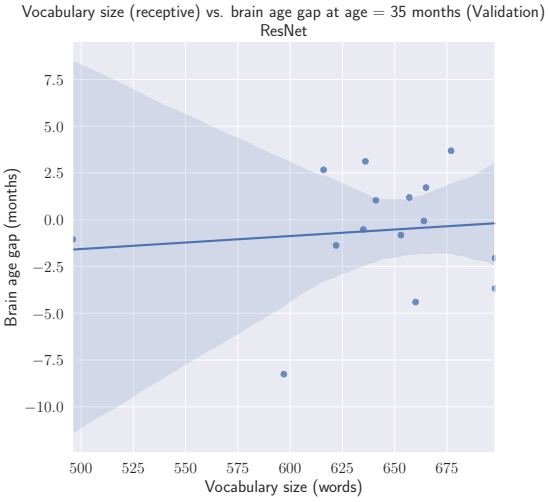

(a) Age = 17 months




(b) Age = 23 months

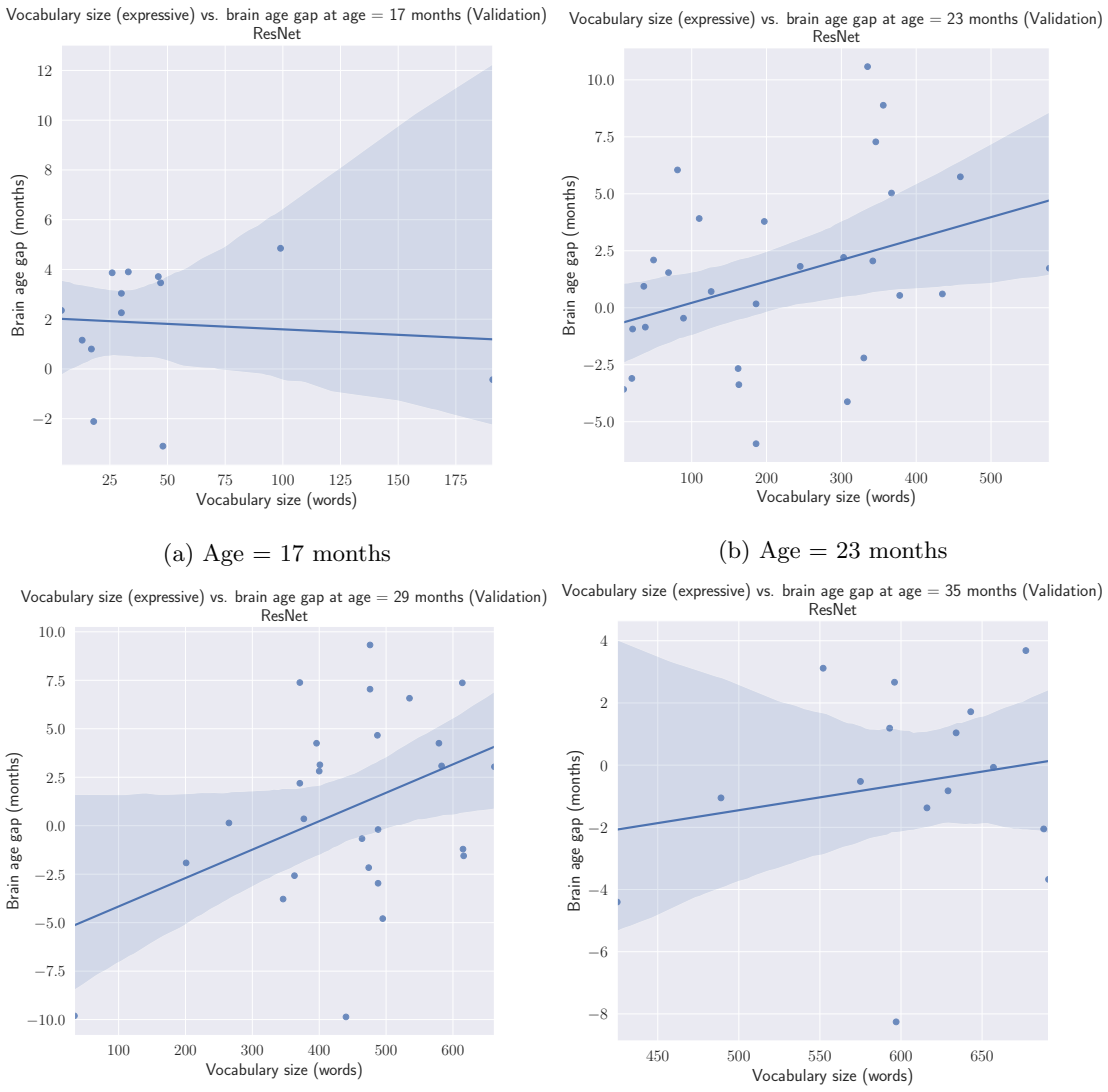



(c) Age = 35 months




(d) Age = 35 months

Figure 95: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 96: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the ResNet's brain age predictions on the validation set.

## Encoder

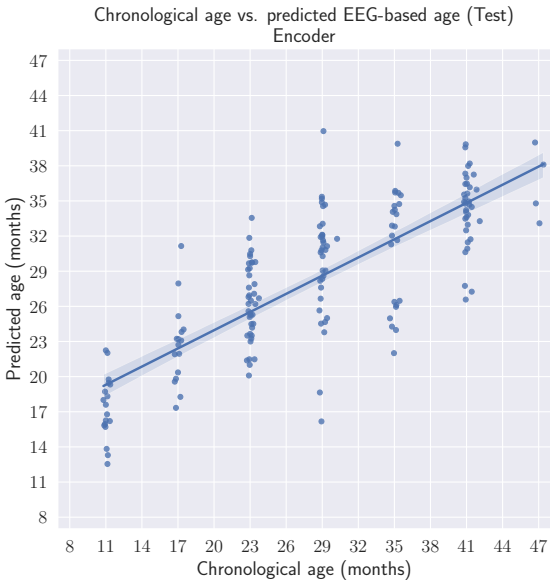



Figure 97: The predictions of the Encoder model compared to the true chronological ages of the subject on the test set.

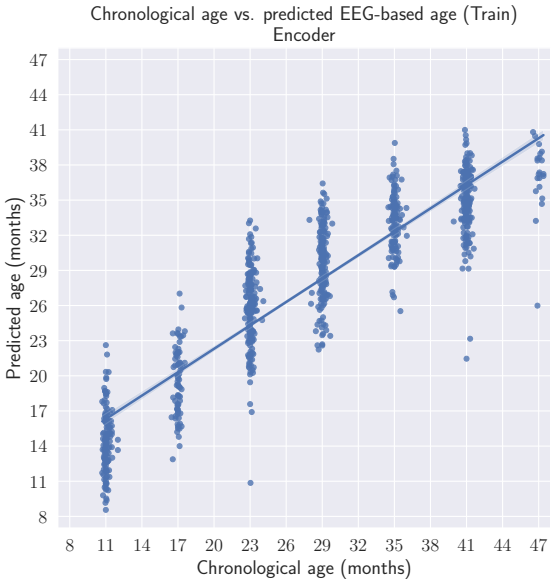



Figure 98: The predictions of the Encoder model compared to the true chronological ages of the subject on the train set.

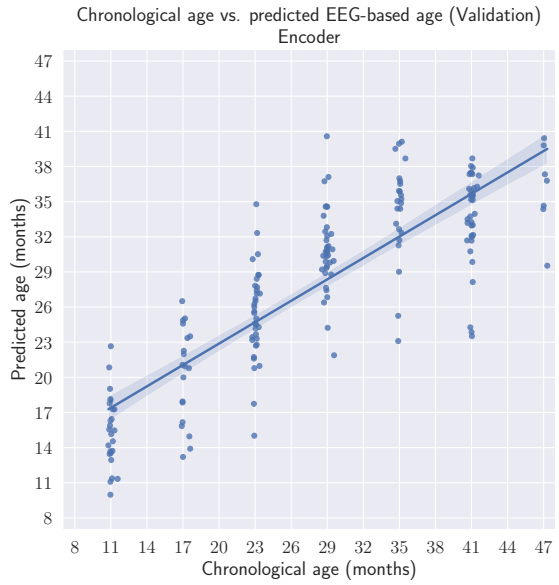



Figure 99: The predictions of the Encoder model compared to the true chronological ages of the subject on the validation set.

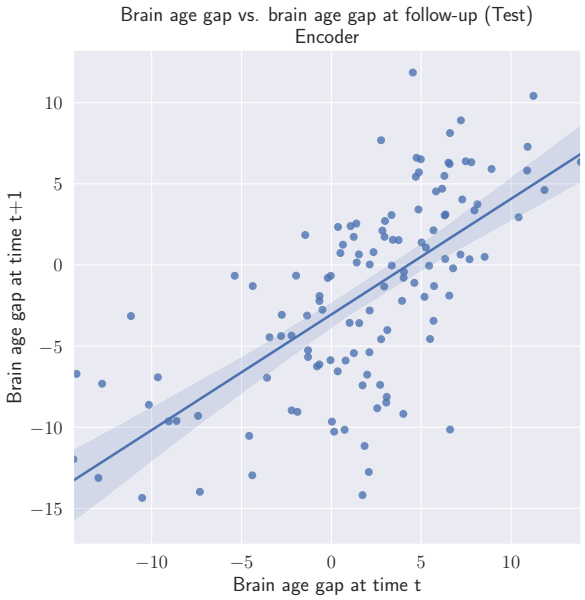
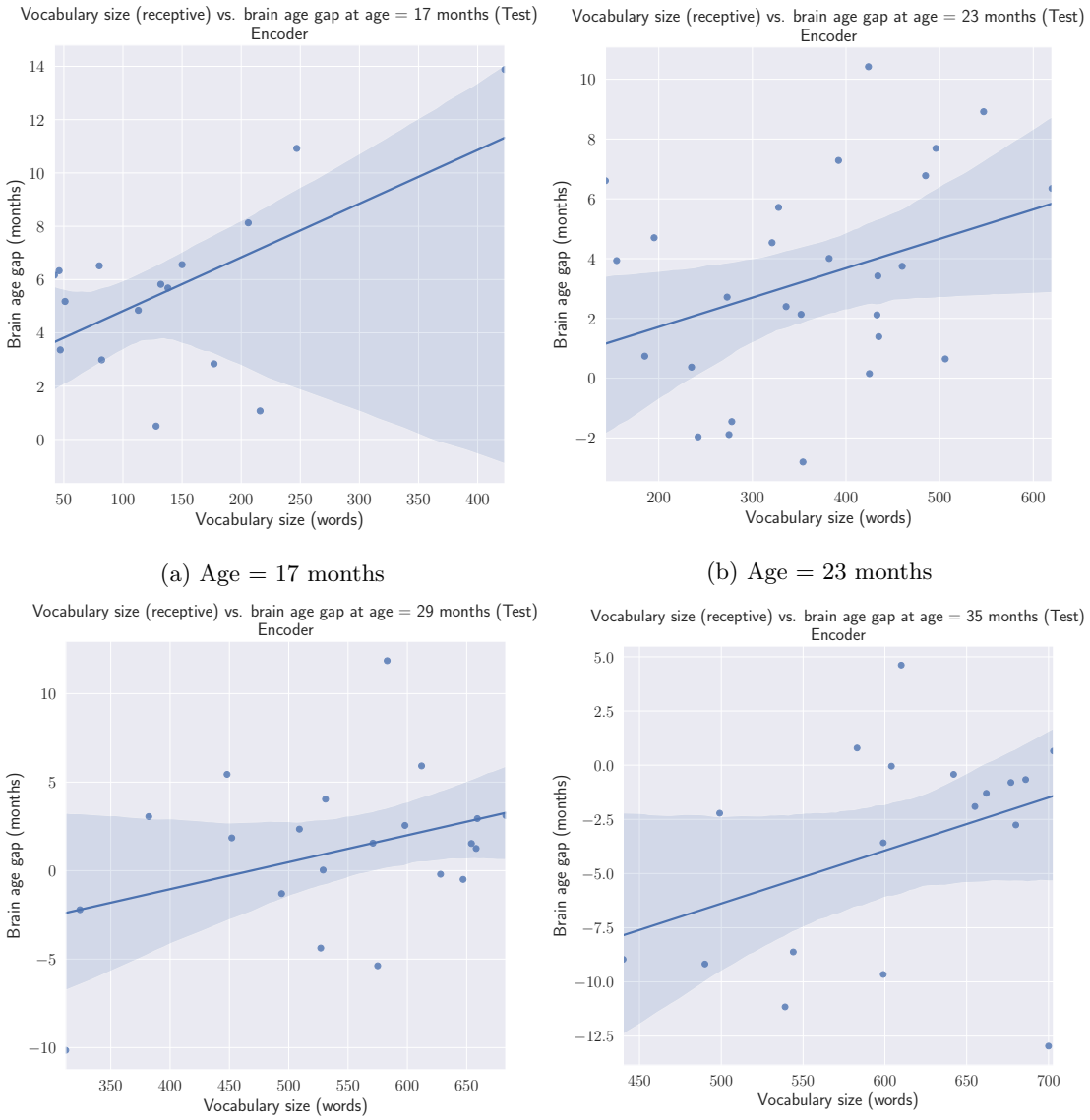



Figure 100: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the Encoder model on the test set.


Figure 101: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the Encoder model on the train set.


Figure 102: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the Encoder model on the validation set.



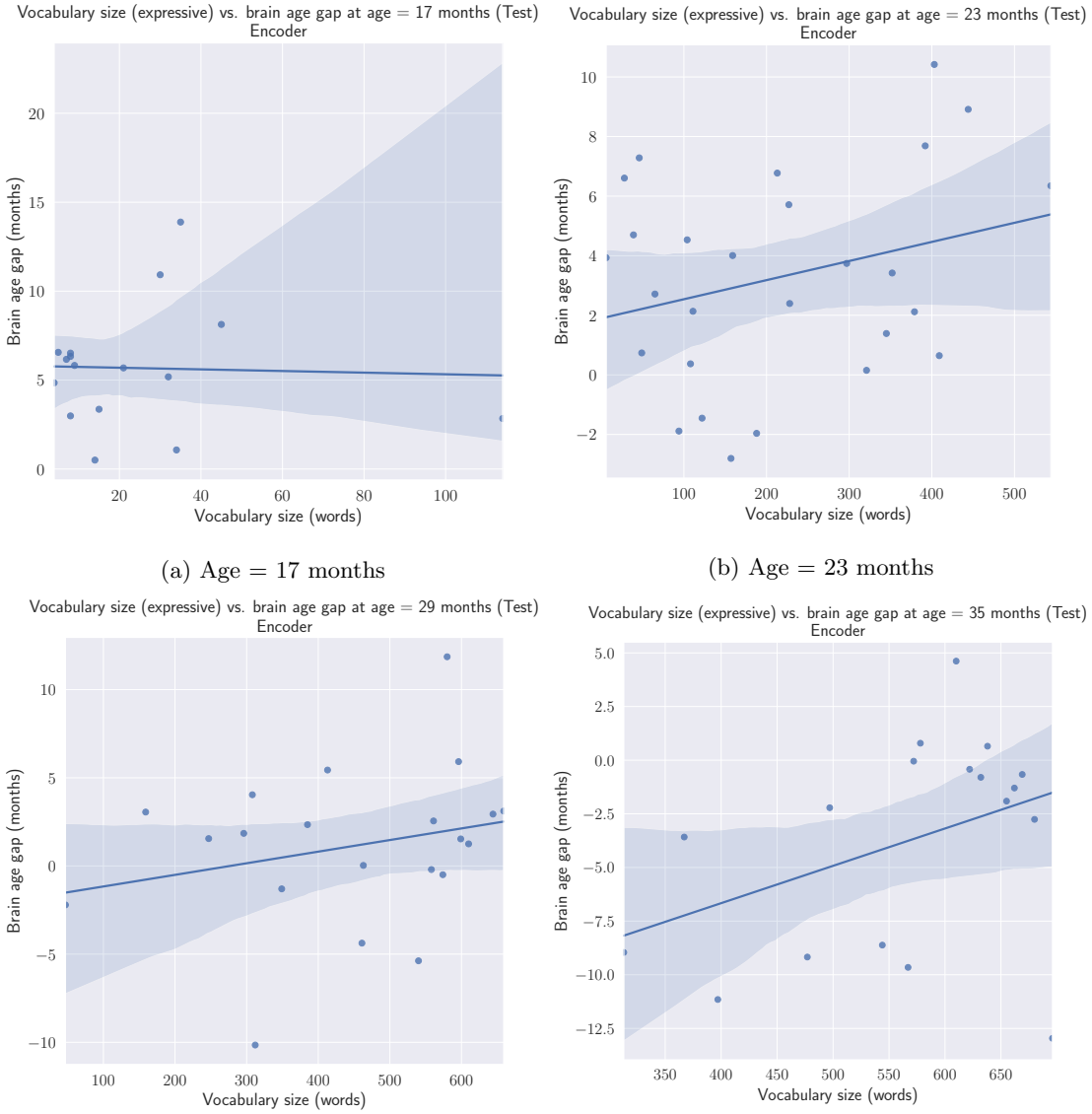

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 103: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the test set.



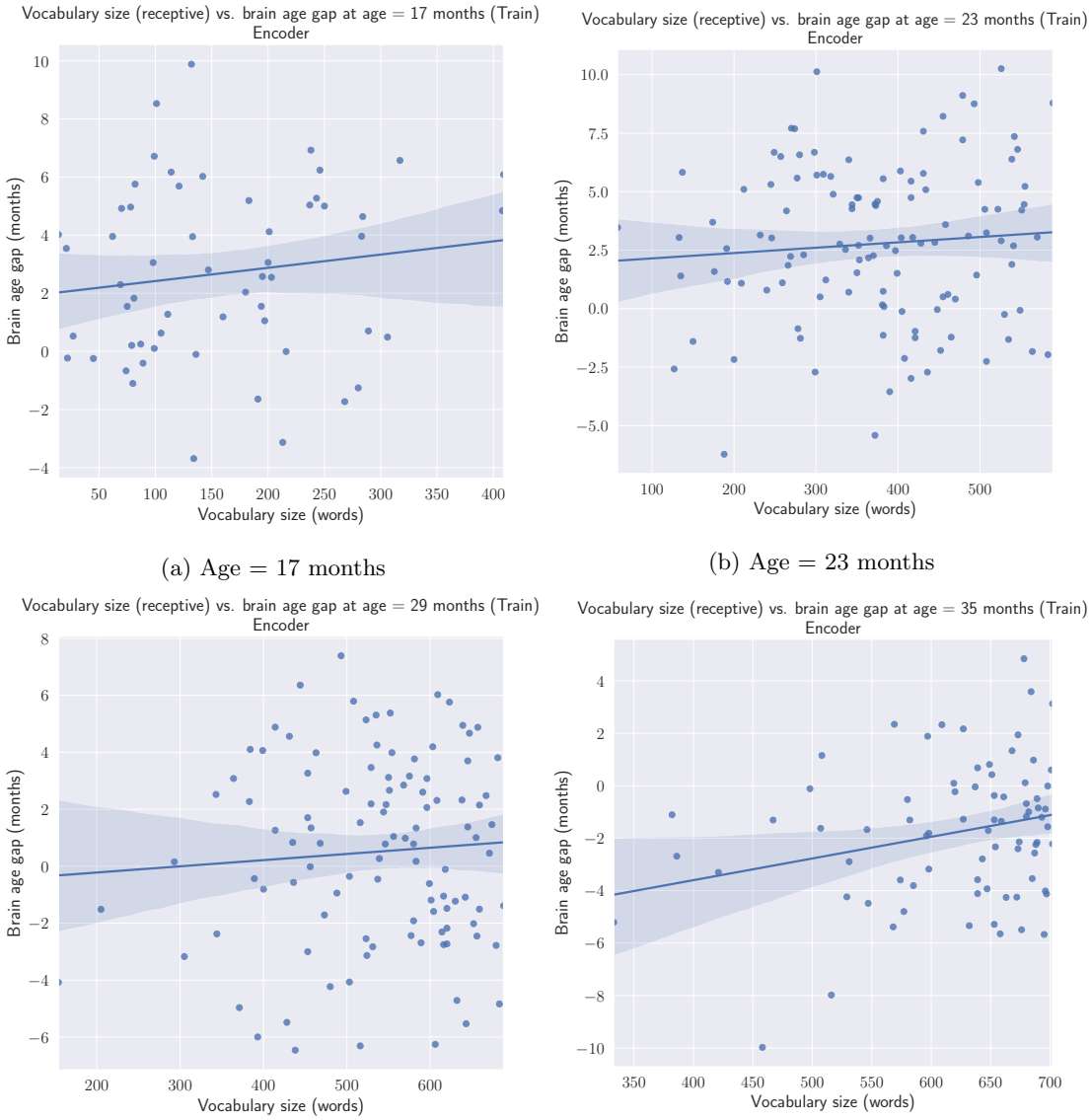

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 104: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the test set.



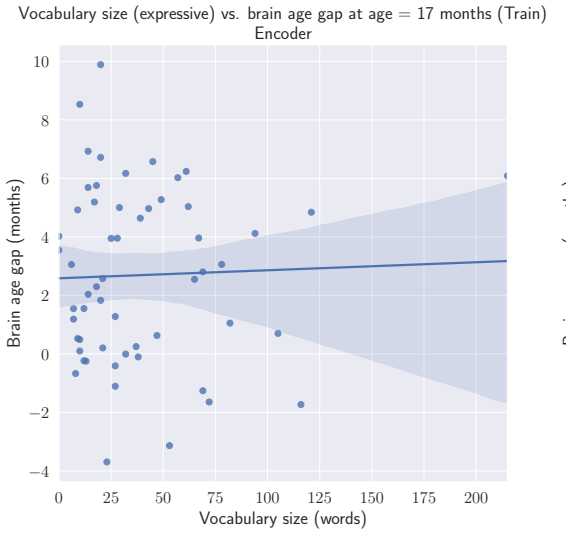

(a) Age = 17 months

(b) Age = 23 months

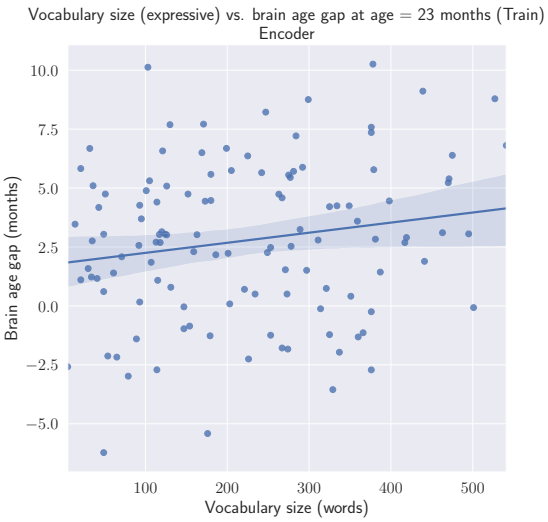
(c) Age = 35 months

(d) Age = 35 months

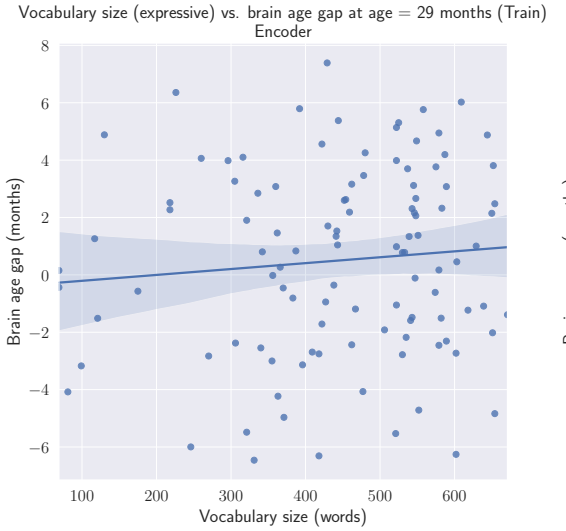
Figure 105: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the train set.



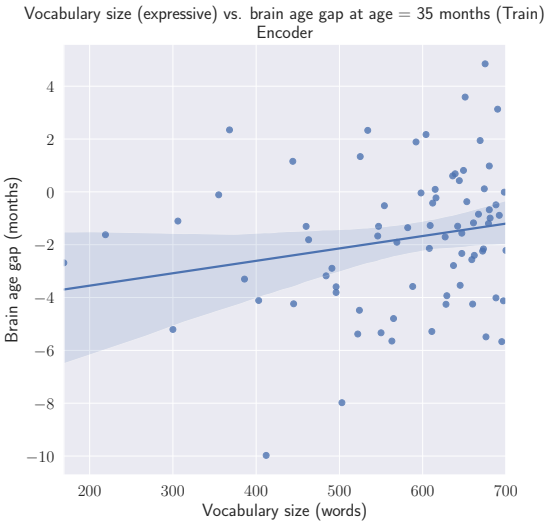

(a) Age = 17 months




(b) Age = 23 months

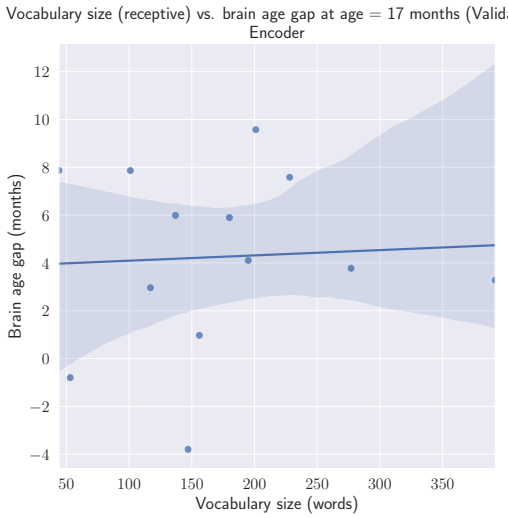



(c) Age = 29 months

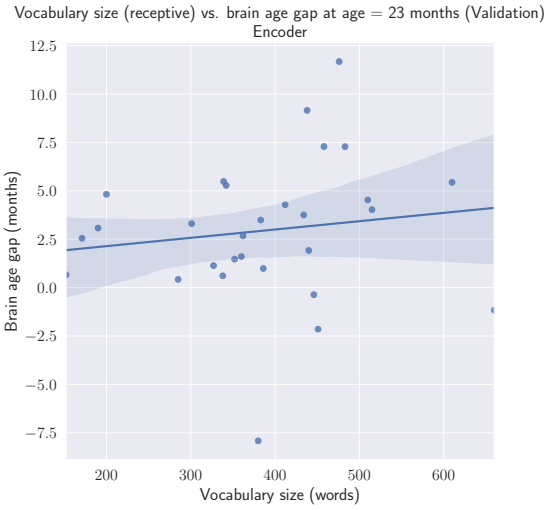



(d) Age = 35 months

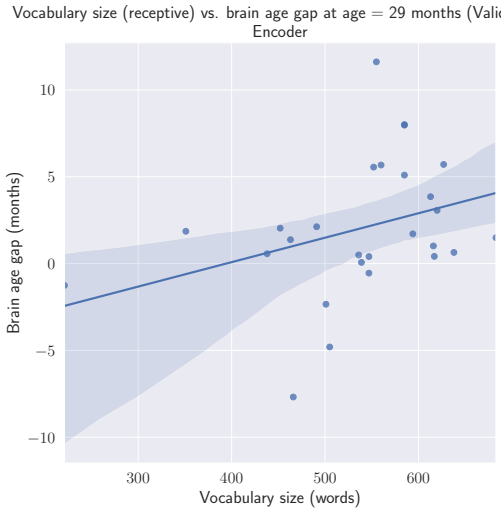
Figure 106: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the train set.



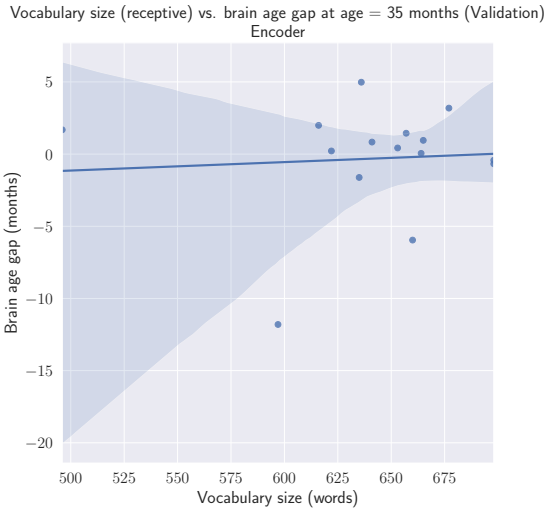

(a) Age = 17 months




(b) Age = 23 months

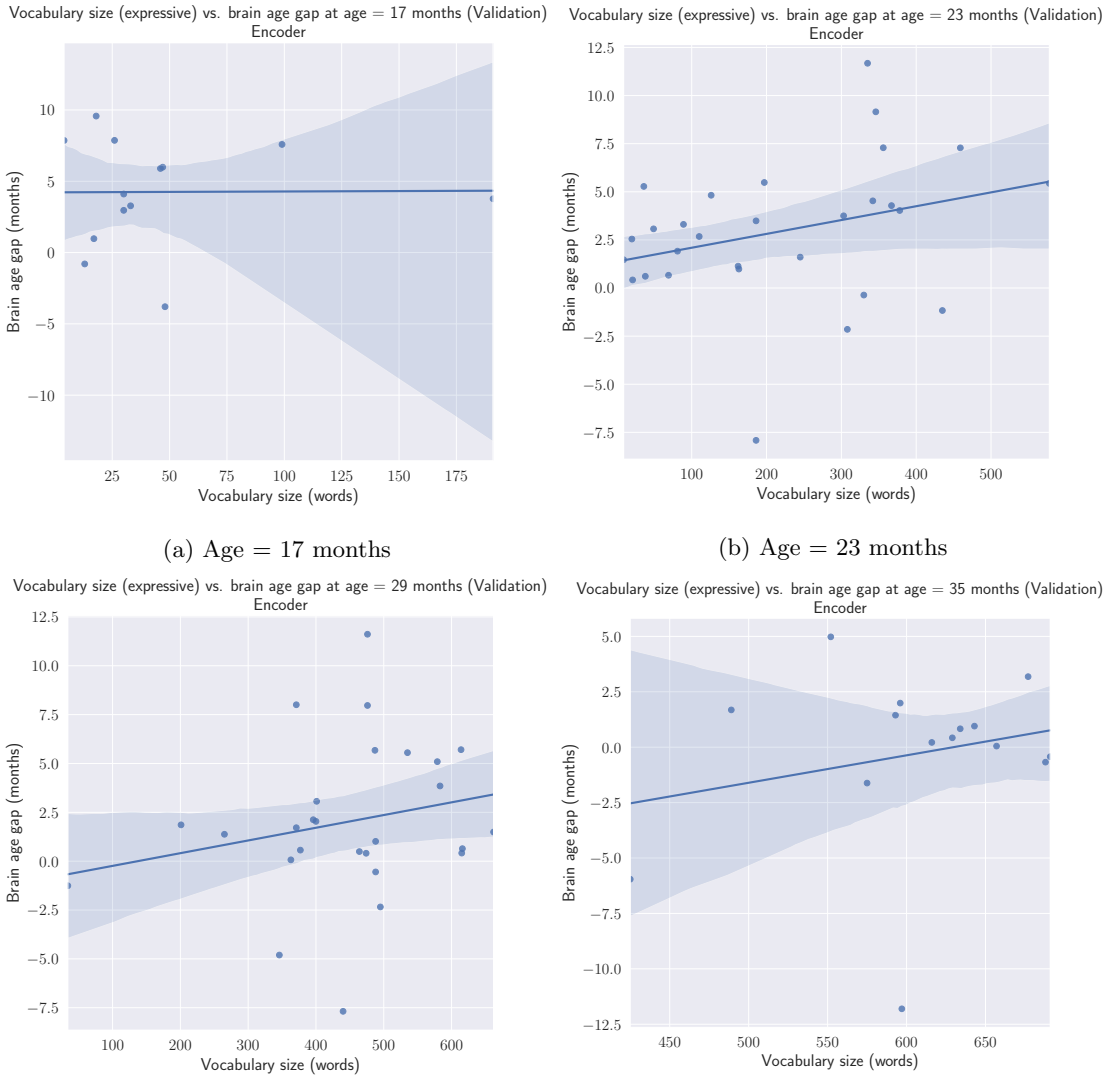



(c) Age = 35 months




(d) Age = 35 months

Figure 107: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 108: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the Encoder model's brain age predictions on the validation set.

## Time-CNN

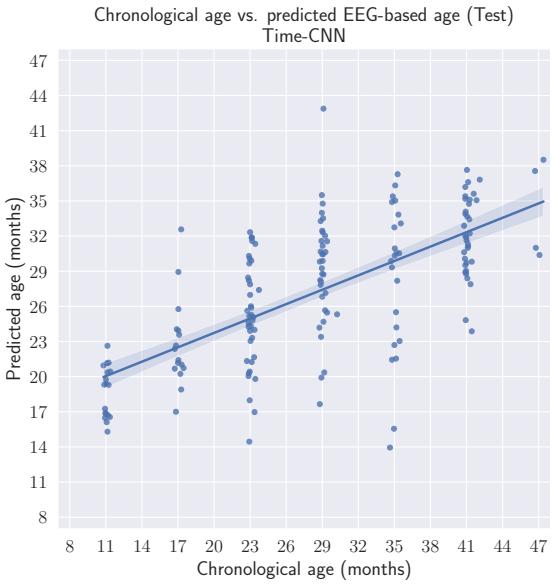



Figure 109: The predictions of the TimeCNN model compared to the true chronological ages of the subject on the test set.

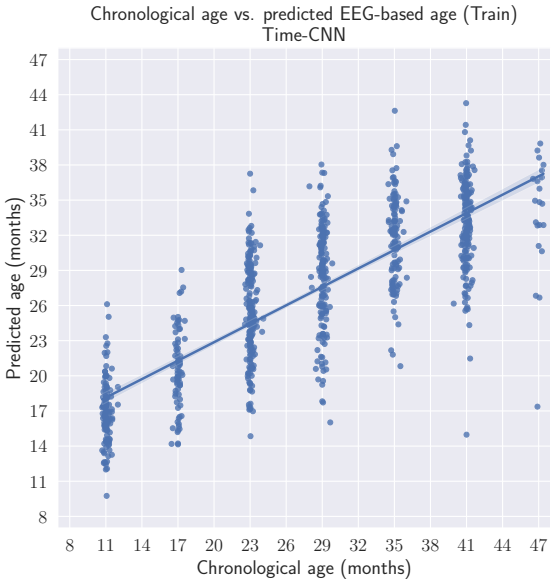



Figure 110: The predictions of the TimeCNN model compared to the true chronological ages of the subject on the train set.

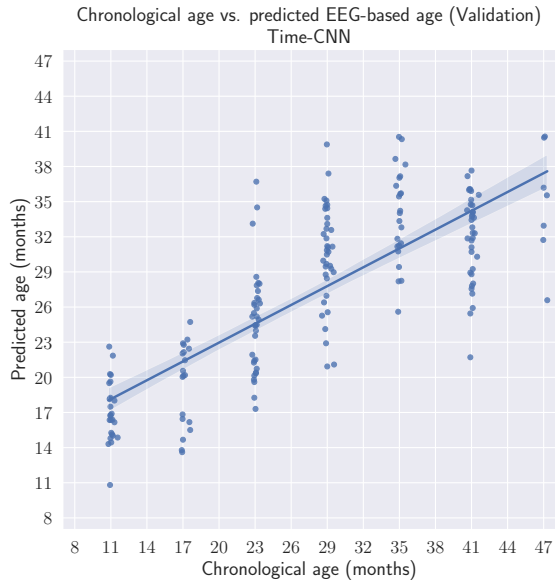



Figure 111: The predictions of the TimeCNN model network compared to the true chronological ages of the subject on the validation set.

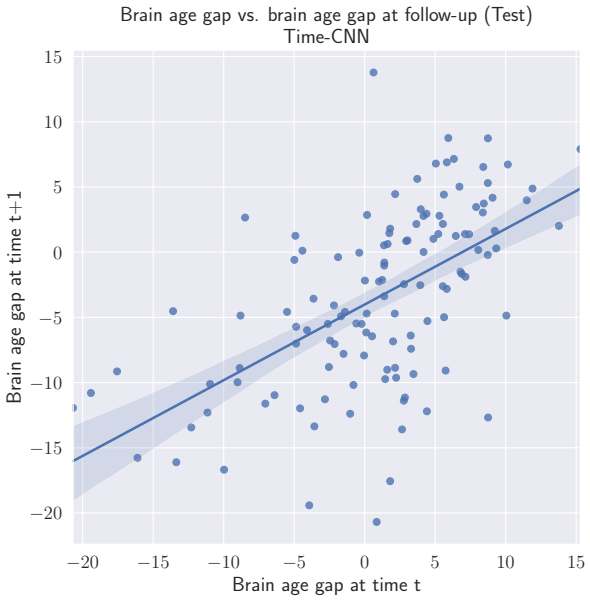



Figure 112: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the TimeCNN model on the test set.

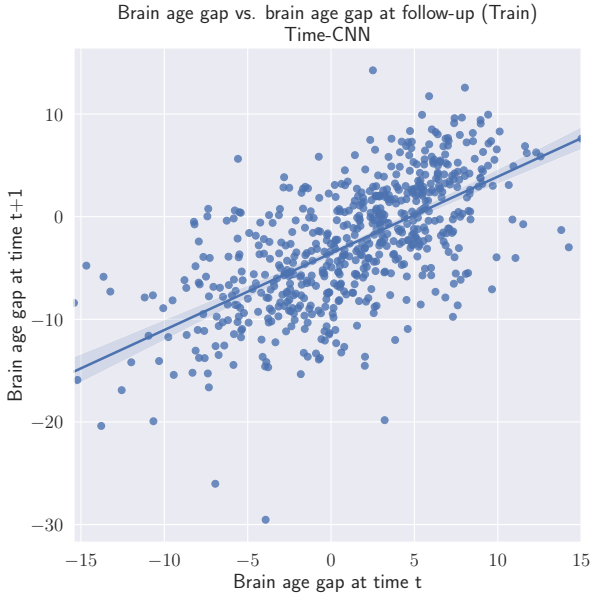



Figure 113: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the TimeCNN model on the train set.

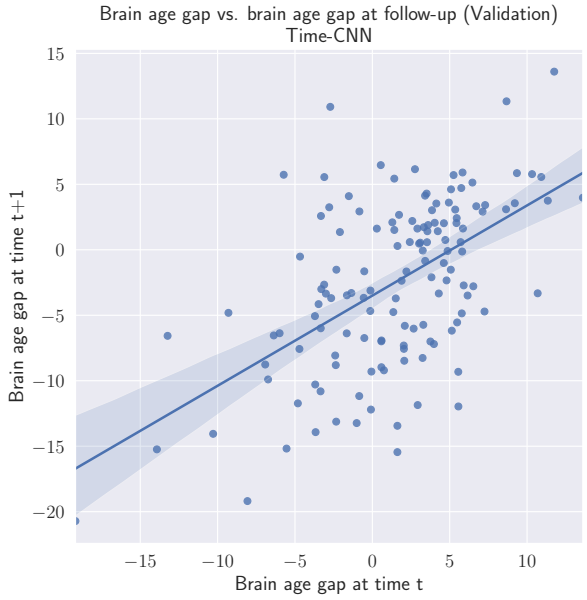
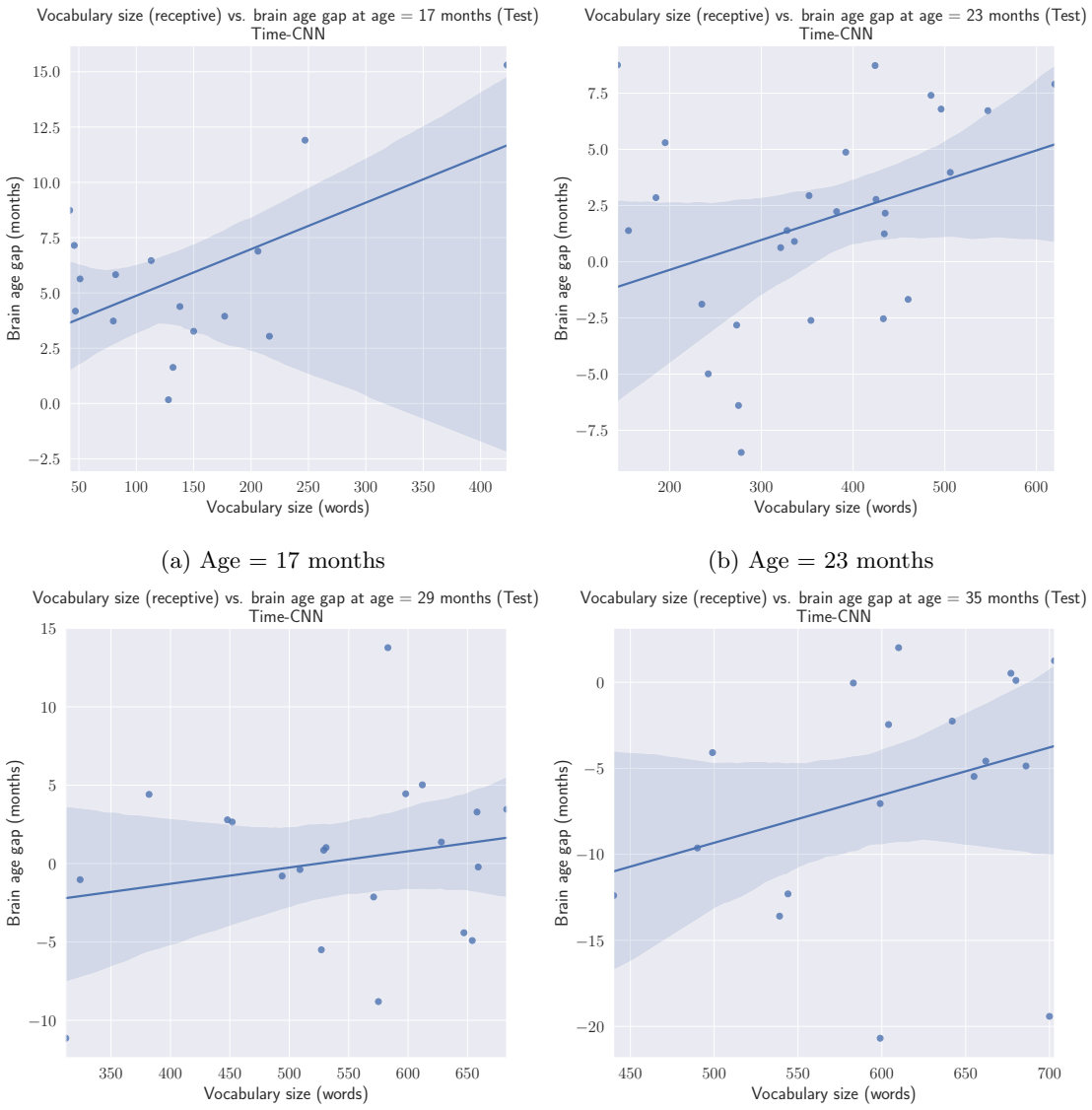



Figure 114: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the TimeCNN model on the validation set.



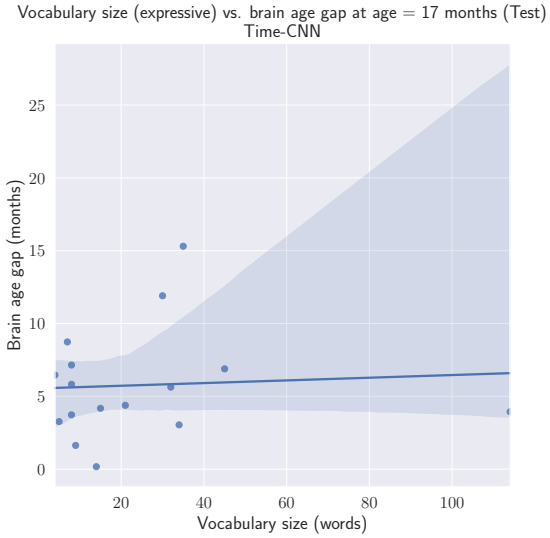

(a) Age = 17 months

(b) Age = 23 months

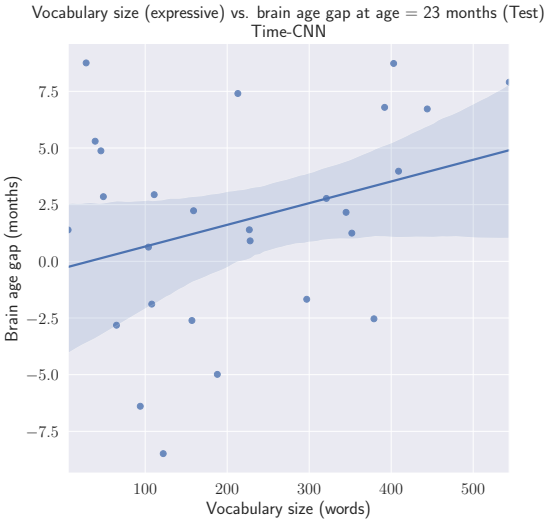
(c) Age = 35 months

(d) Age = 35 months

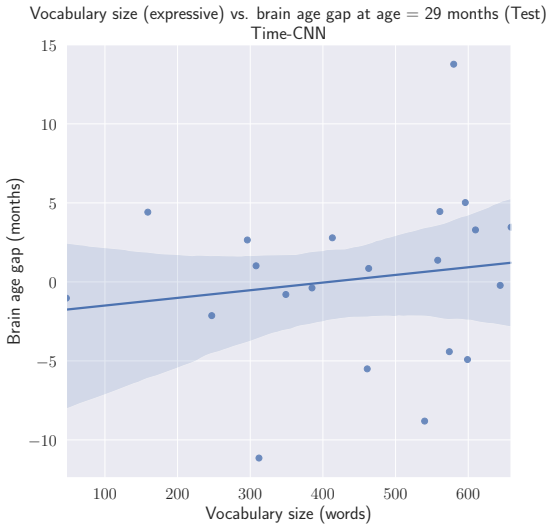
Figure 115: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the test set.




(a) Age = 17 months



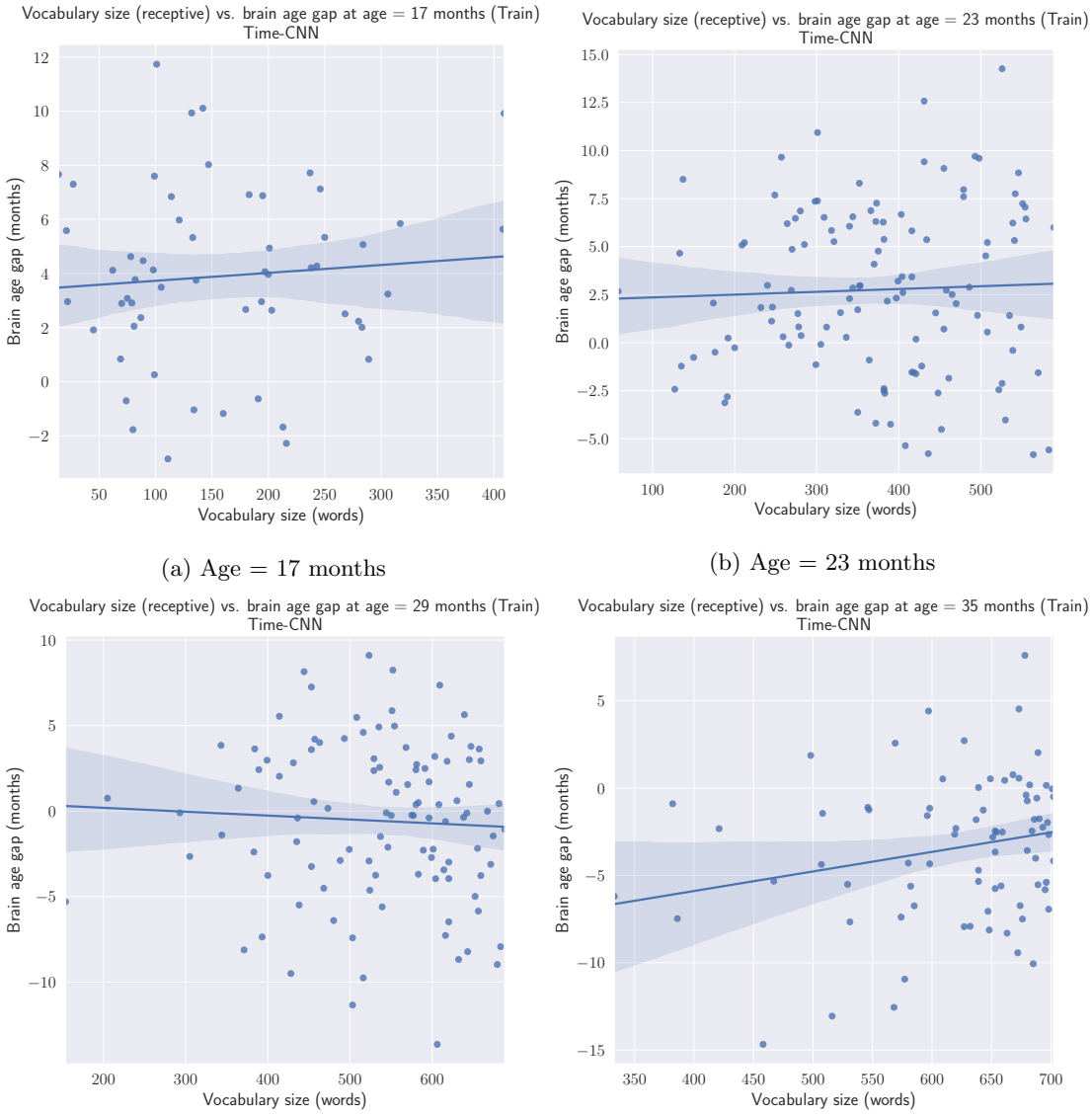

(b) Age = 23 months




(c) Age = 29 months


(d) Age = 35 months

Figure 116: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the test set.



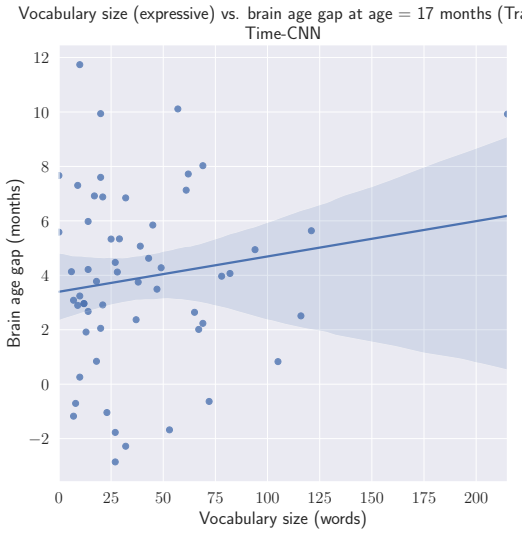

(a) Age = 17 months

(b) Age = 23 months

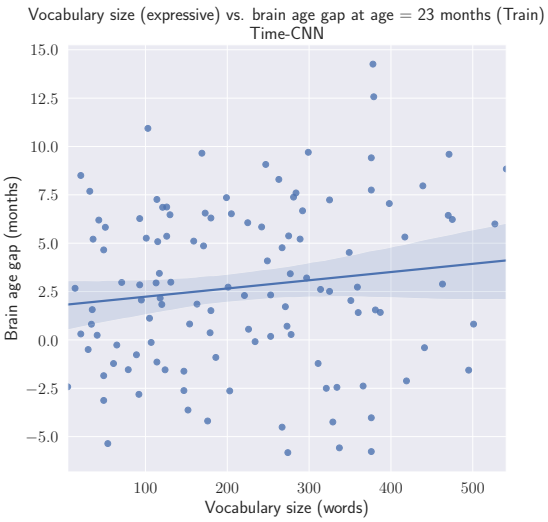
(c) Age = 35 months

(d) Age = 35 months

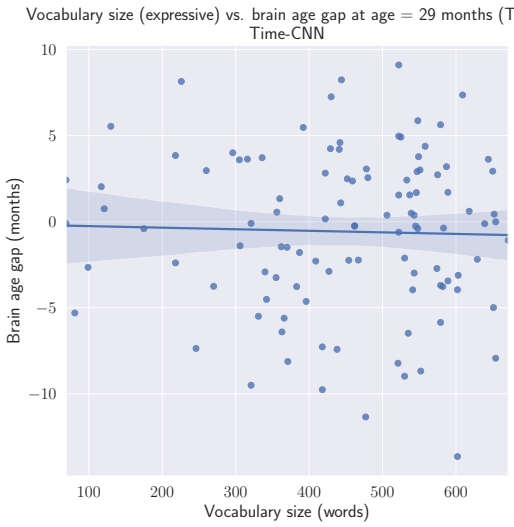
Figure 117: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the train set.



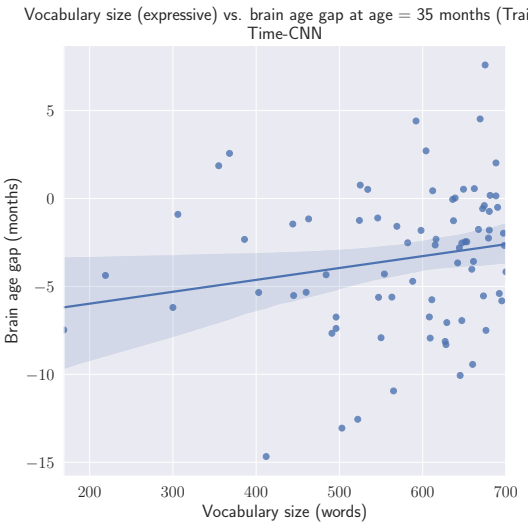

(a) Age = 17 months




(b) Age = 23 months

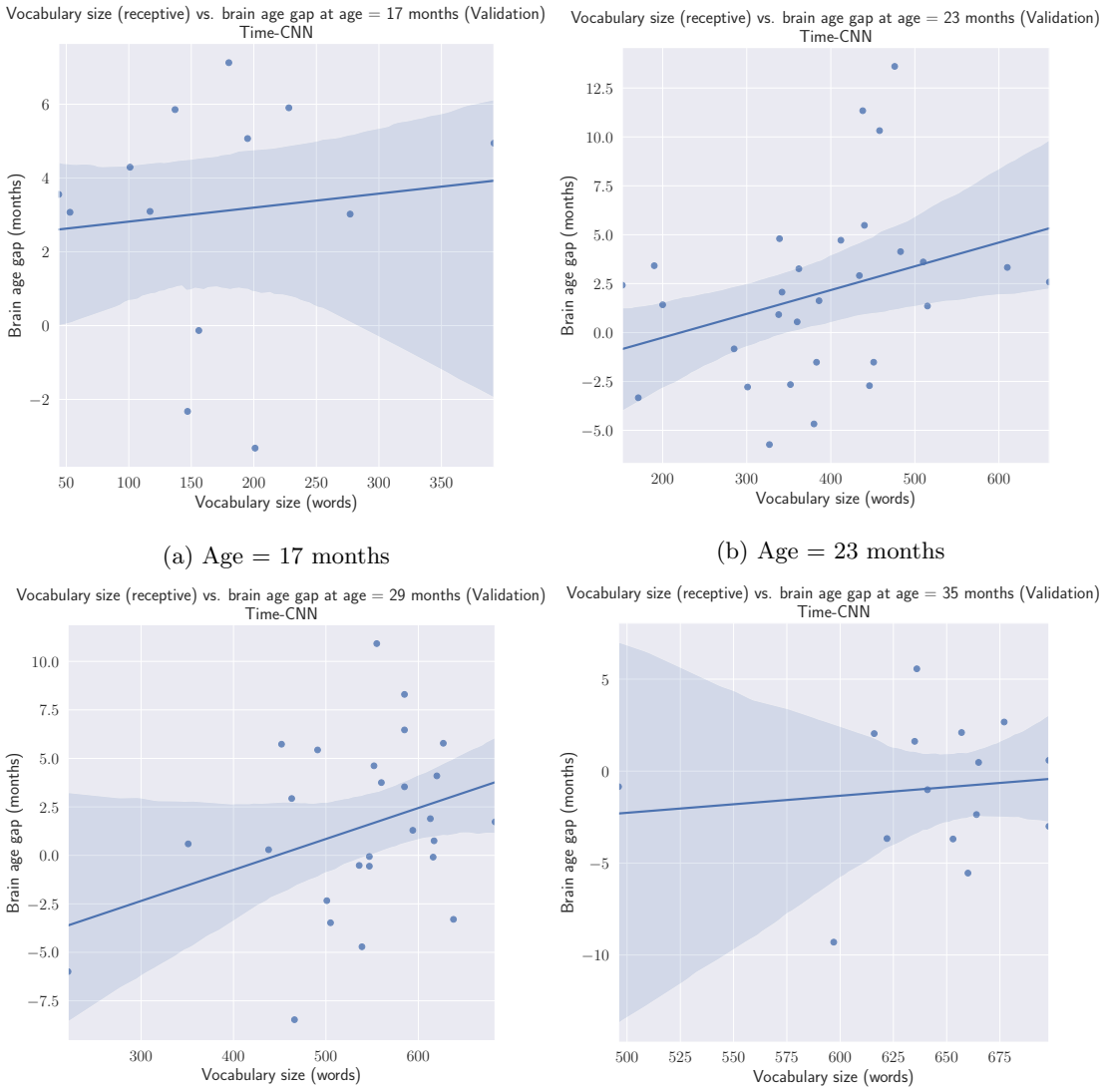



(c) Age = 29 months




(d) Age = 35 months

Figure 118: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the train set.



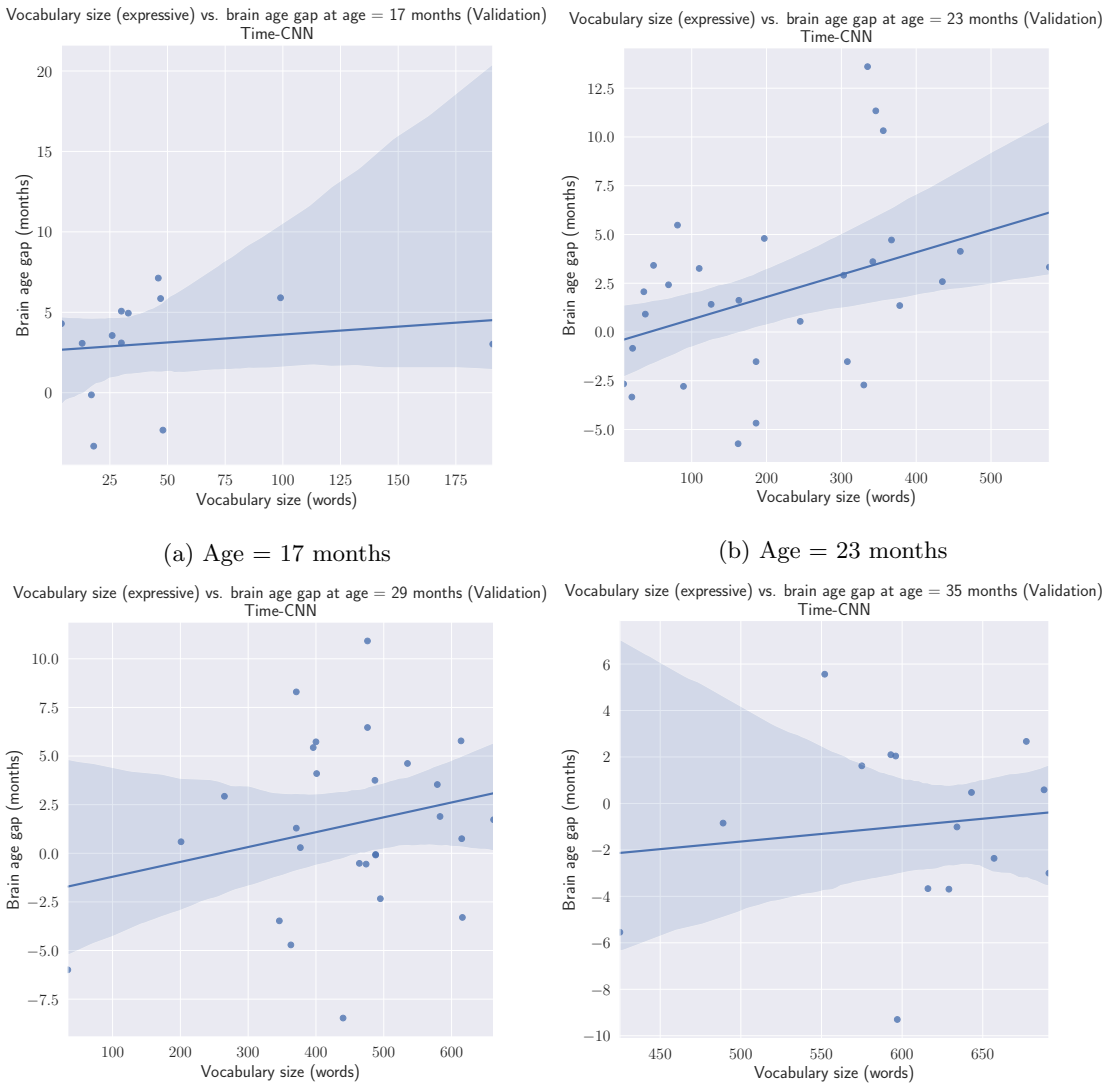

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 119: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 120: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the TimeCNN model's brain age predictions on the validation set.

## InceptionTime

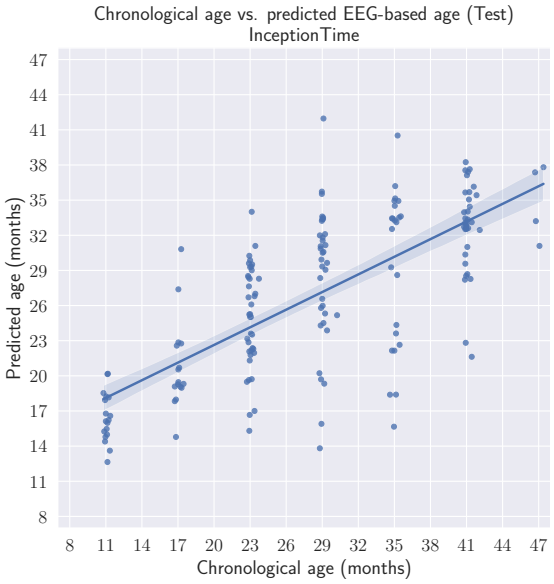



Figure 121: The predictions of the InceptionTime model compared to the true chronological ages of the subject on the test set.

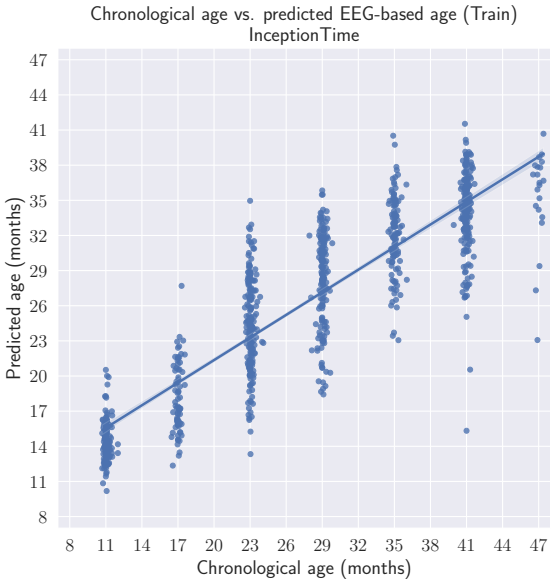



Figure 122: The predictions of the InceptionTime model compared to the true chronological ages of the subject on the train set.

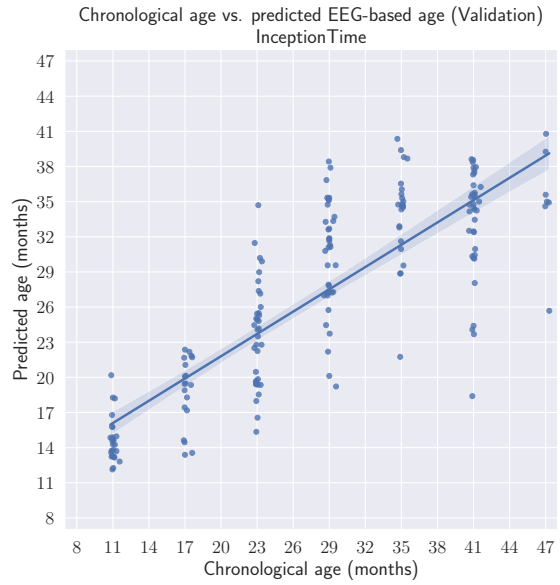



Figure 123: The predictions of the InceptionTime model compared to the true chronological ages of the subject on the validation set.

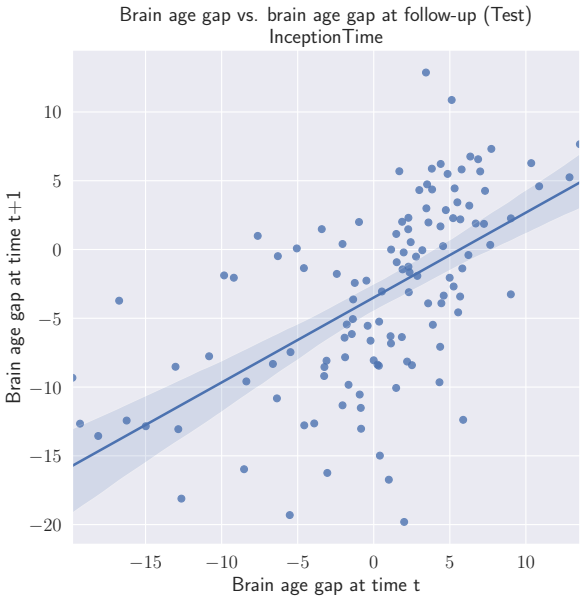



Figure 124: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the InceptionTime model on the test set.

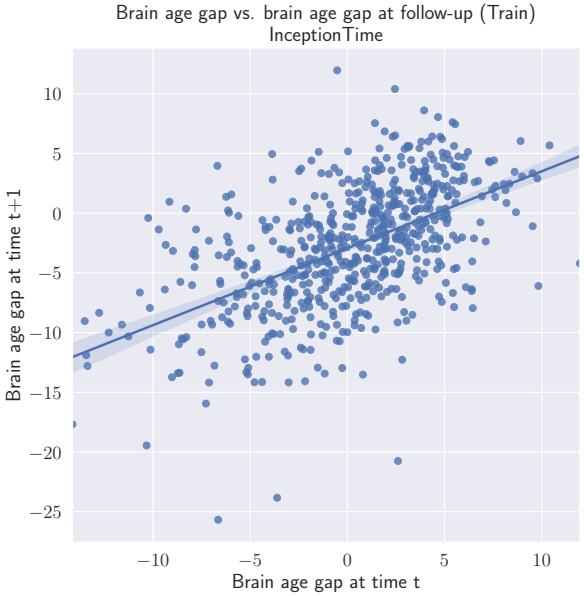
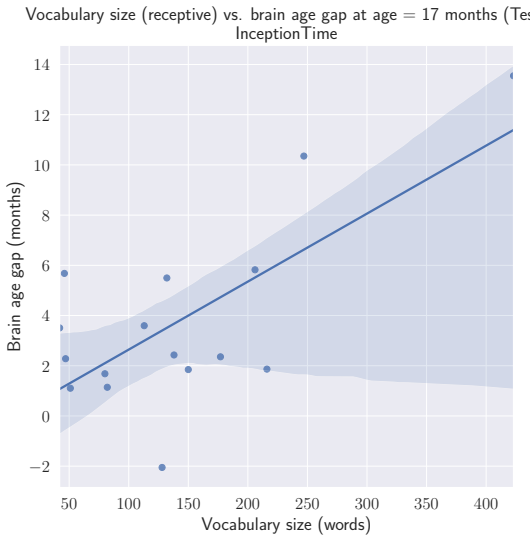
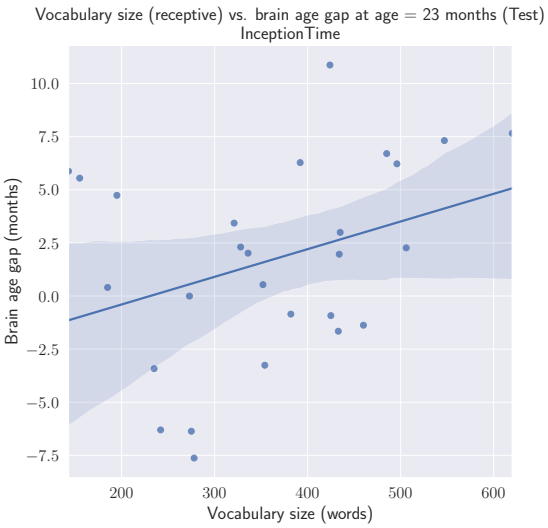



Figure 125: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the InceptionTime model on the train set.


Figure 126: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the InceptionTime model on the validation set.

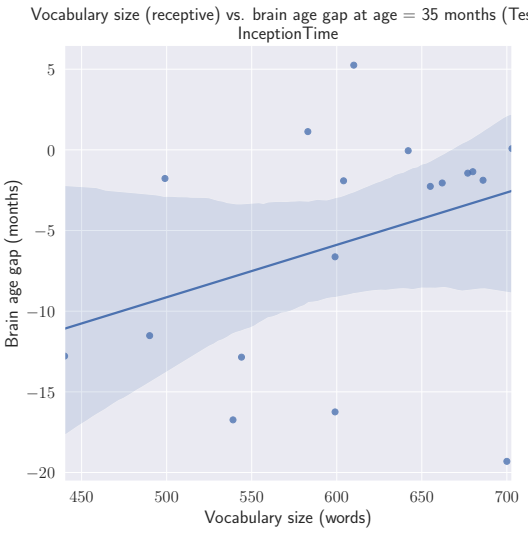



(a) Age = 17 months



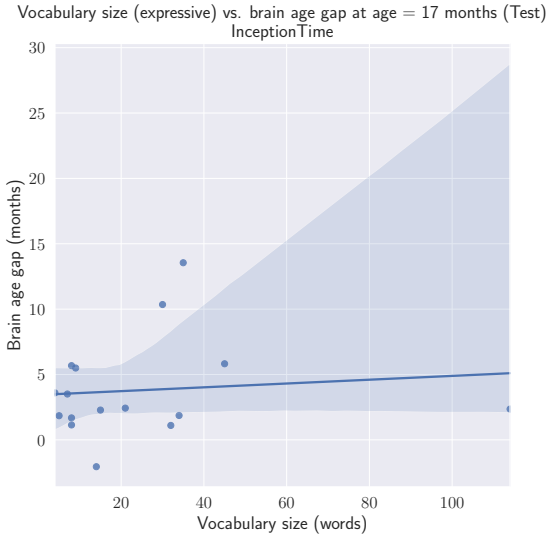

(b) Age = 23 months


(c) Age = 35 months

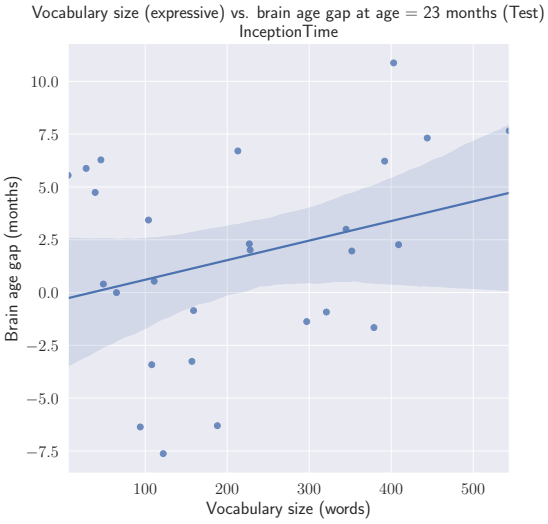



(d) Age = 35 months

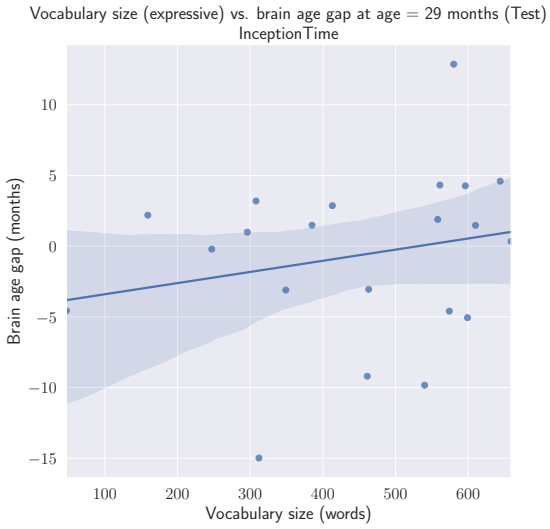
Figure 127: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the test set.



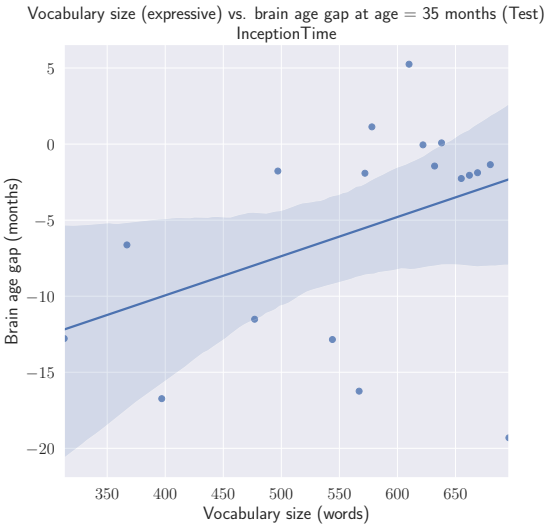

(a) Age = 17 months




(b) Age = 23 months

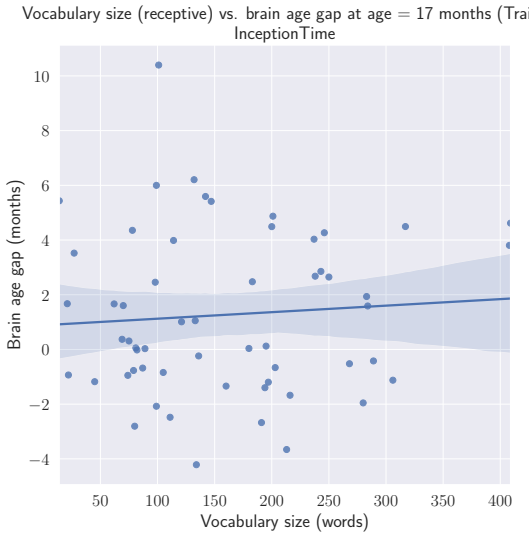



(c) Age = 29 months

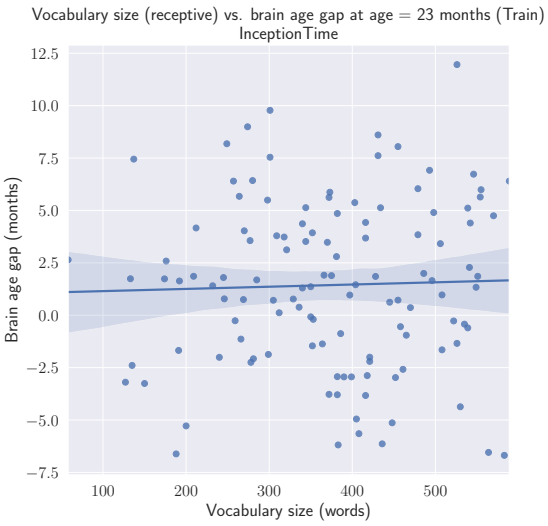



(d) Age = 35 months

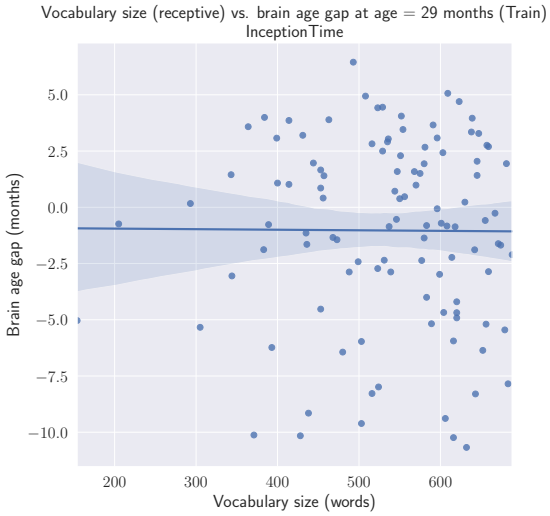
Figure 128: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the test set.



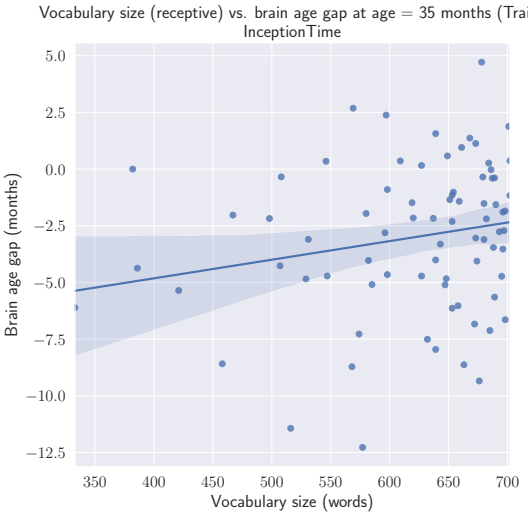

(a) Age = 17 months




(b) Age = 23 months

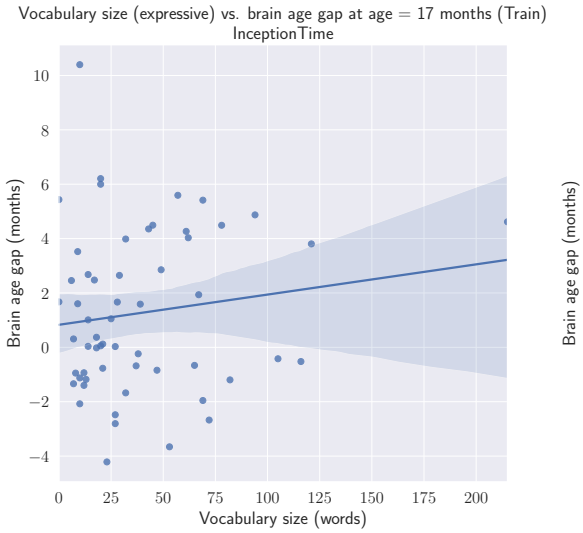



(c) Age = 35 months

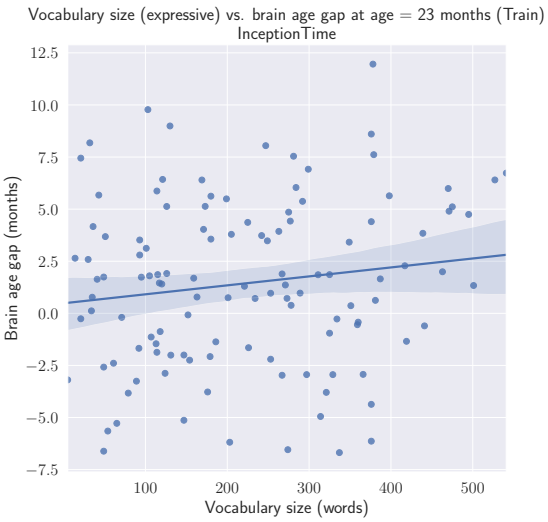



(d) Age = 35 months

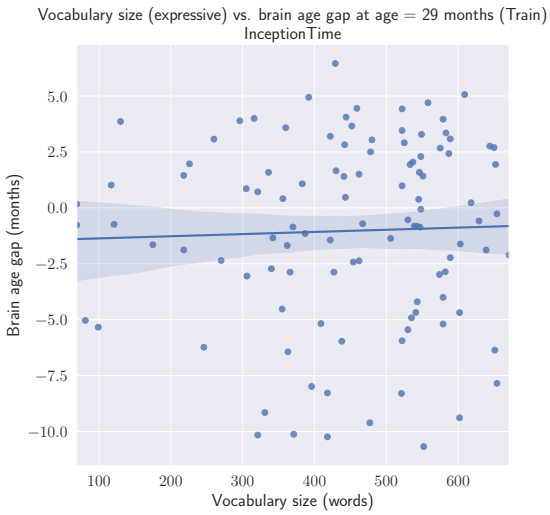
Figure 129: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the train set.



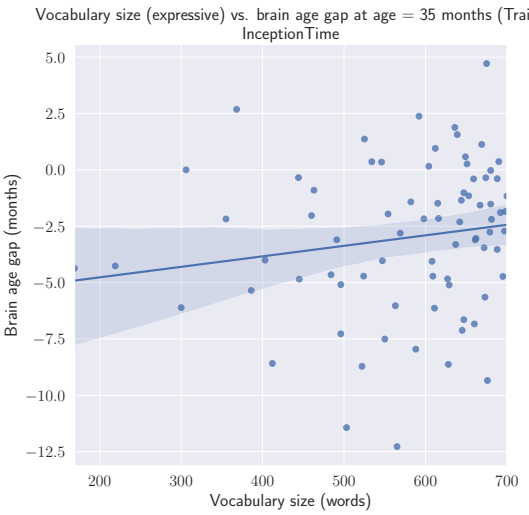

(a) Age = 17 months




(b) Age = 23 months

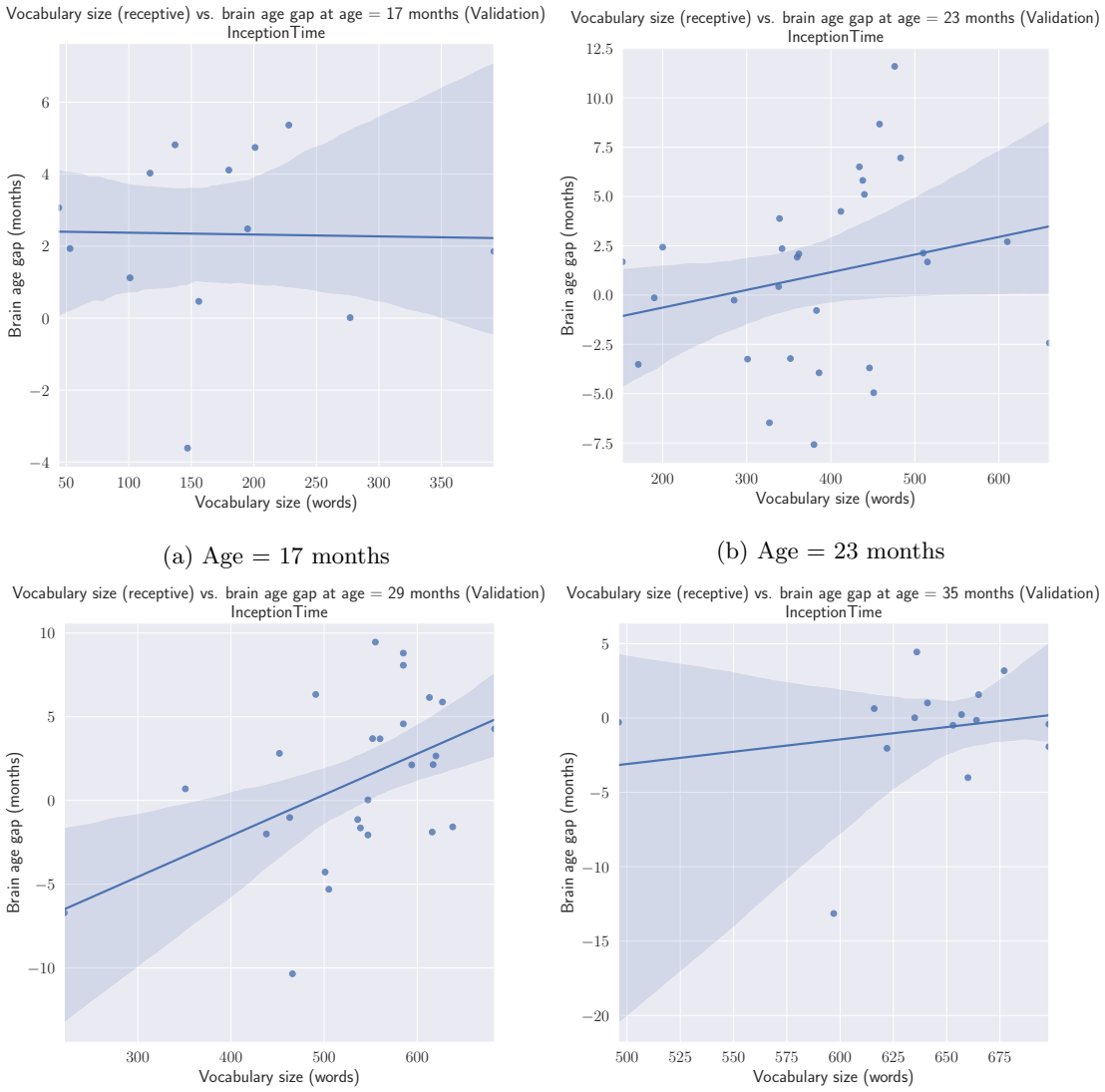



(c) Age = 29 months




(d) Age = 35 months

Figure 130: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the train set.



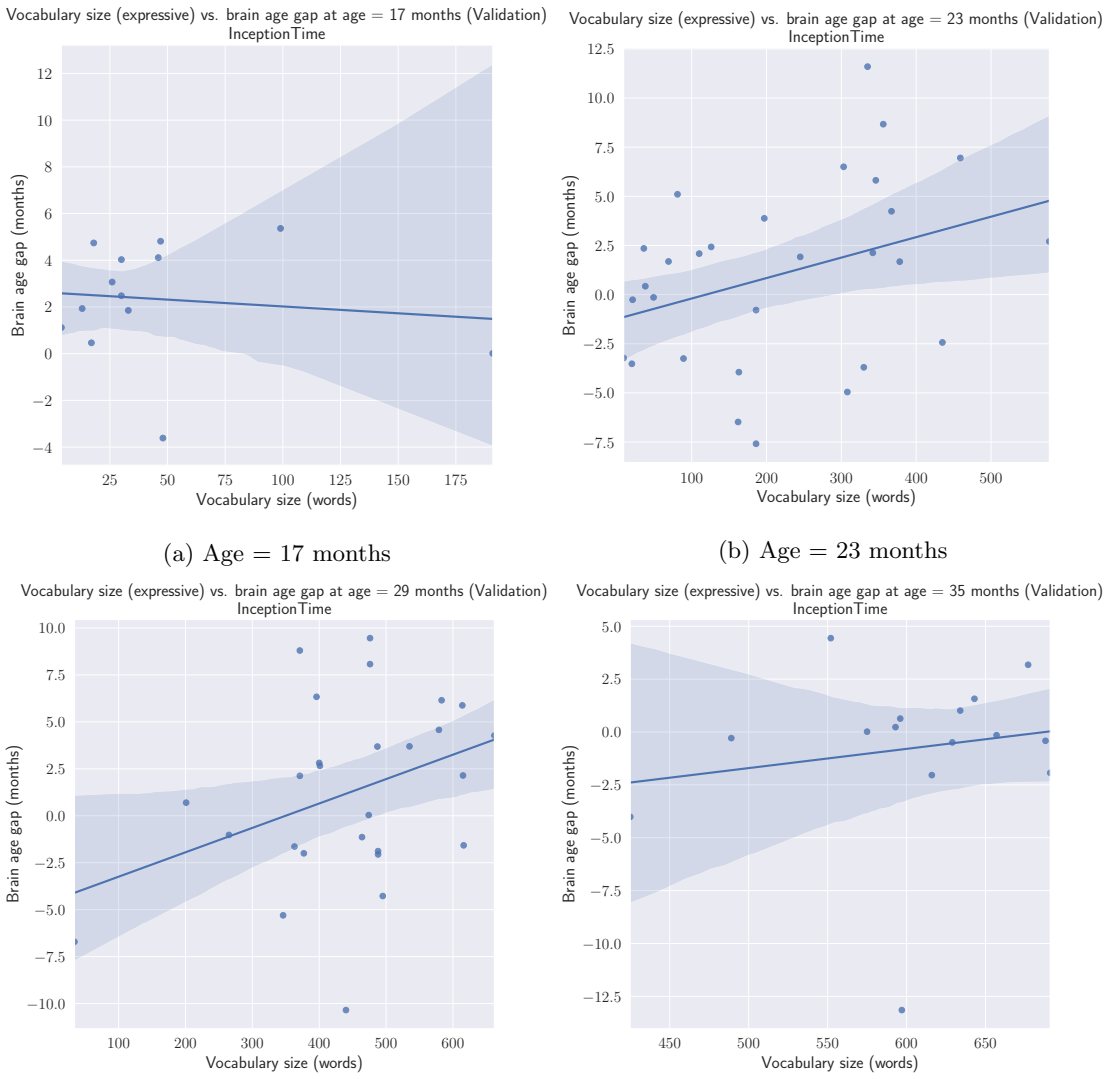

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 131: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the validation set.




(a) Age = 17 months

(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 132: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the InceptionTime model's brain age predictions on the validation set.

## BLSTM-LSTM

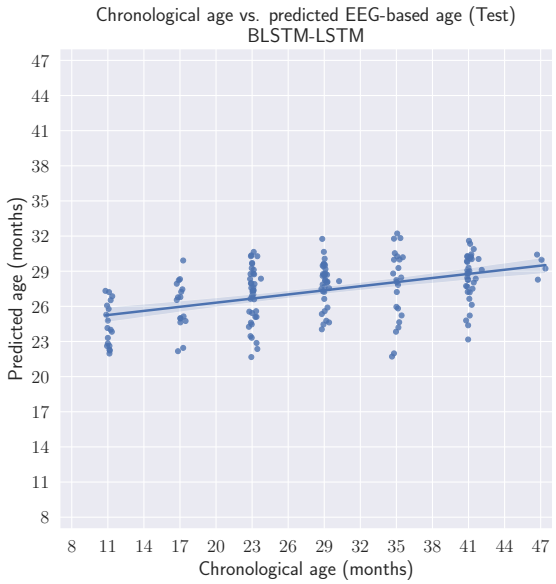



Figure 133: The predictions of the BLSTM-LSTM model compared to the true chronological ages of the subject on the test set.

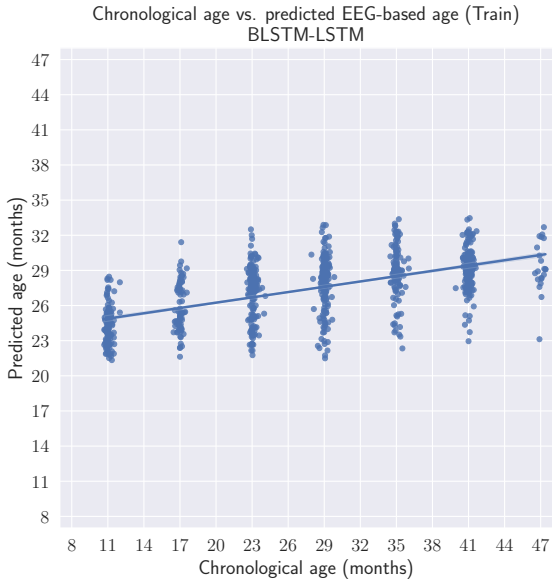



Figure 134: The predictions of the BLSTM-LSTM model compared to the true chronological ages of the subject on the train set.

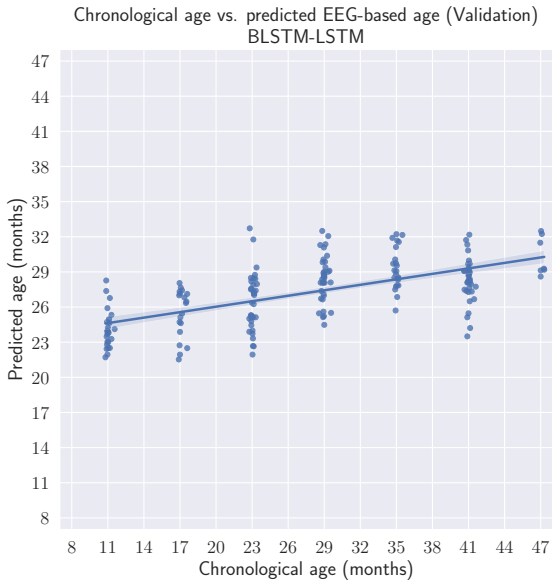



Figure 135: The predictions of the BLSTM-LSTM model compared to the true chronological ages of the subject on the validation set.

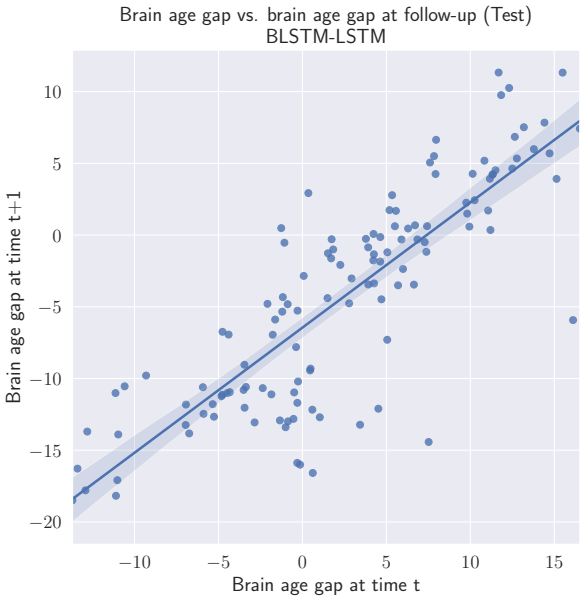



Figure 136: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the BLSTM-LSTM model on the test set.

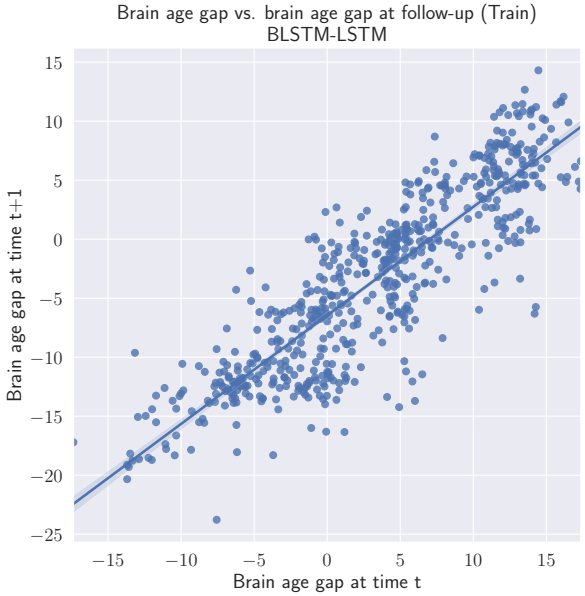



Figure 137: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the BLSTM-LSTM model on the train set.

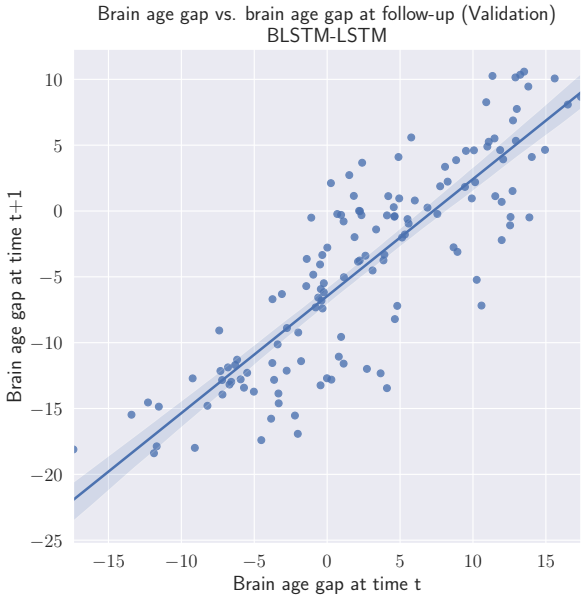
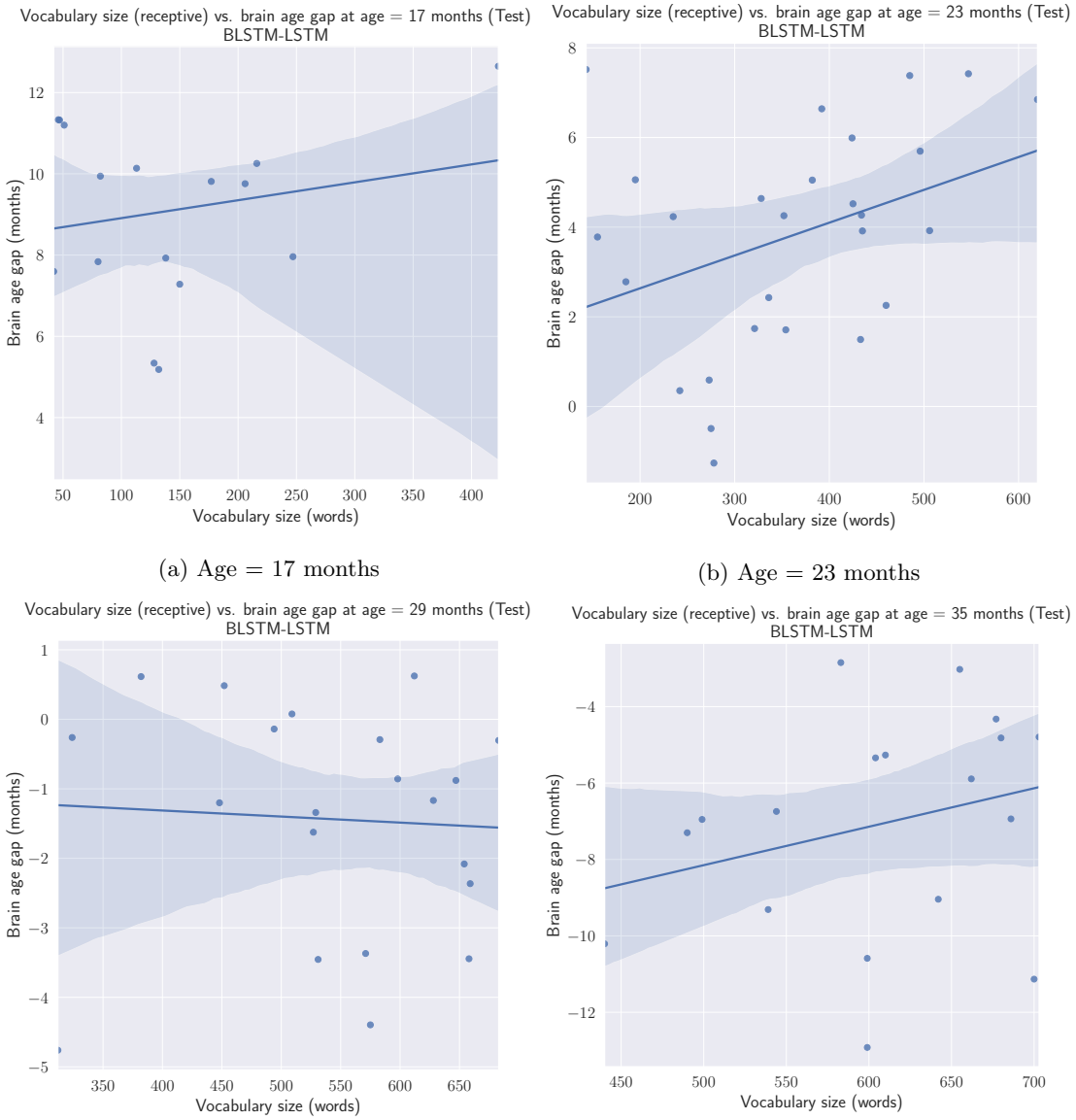



Figure 138: The brain age gap at time $t$ compared to the brain age gap at follow-up, time $t+1$ for the BLSTM-LSTM model on the validation set.



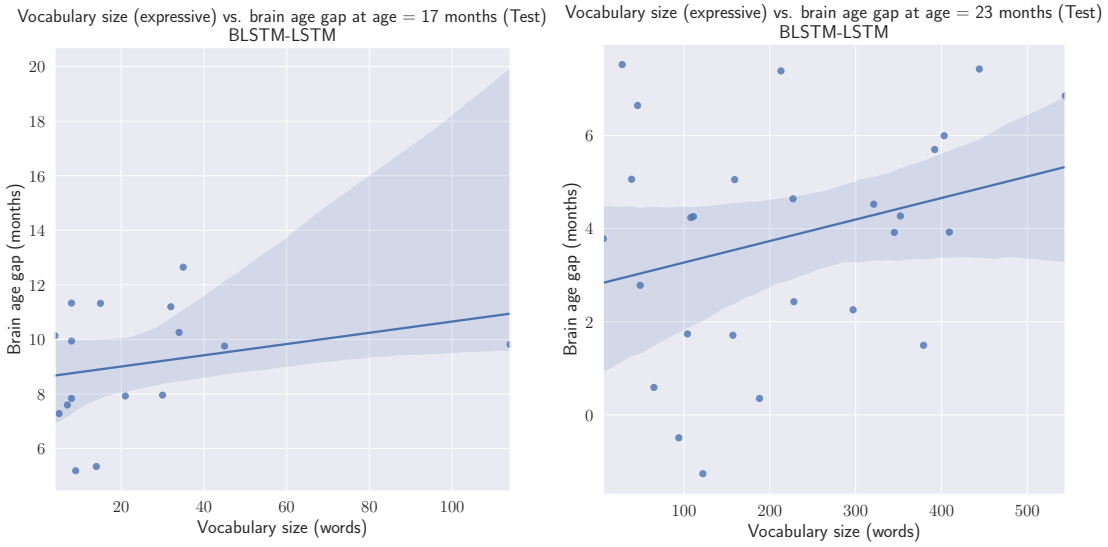

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

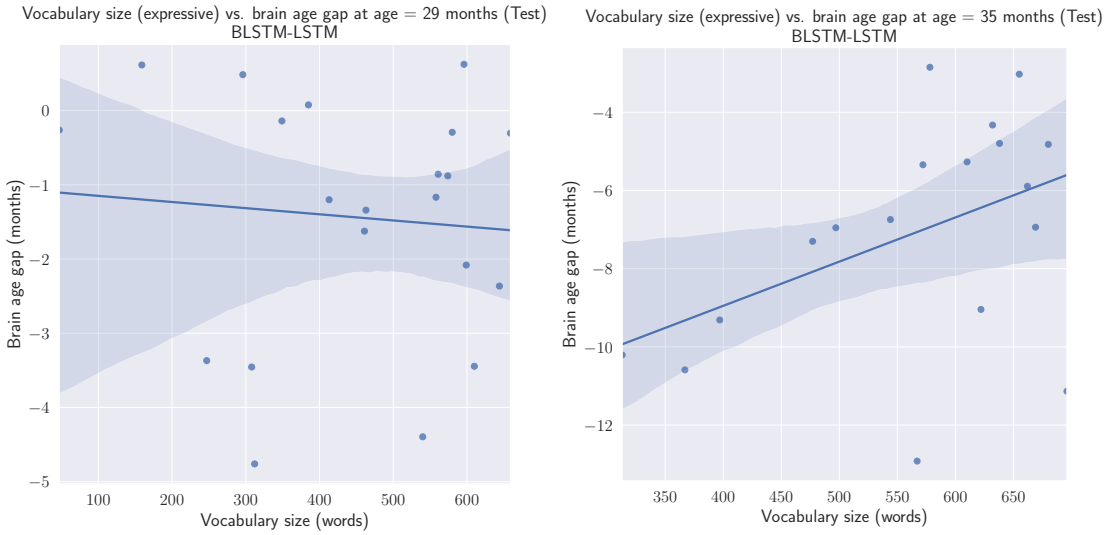
(d) Age = 35 months

Figure 139: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the test set.




(a) Age = 17 months

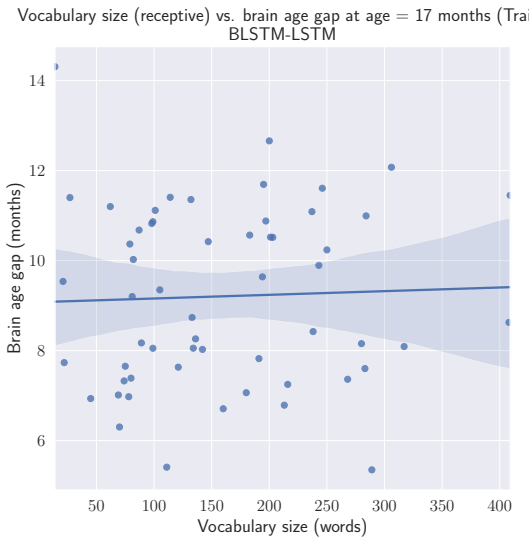
(b) Age = 23 months



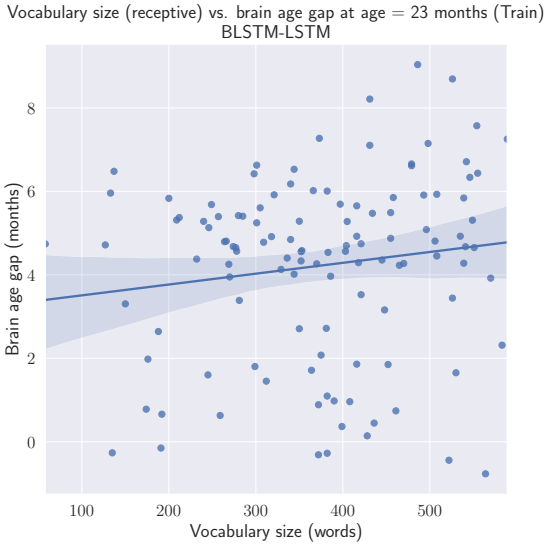

(c) Age = 29 months

(d) Age = 35 months

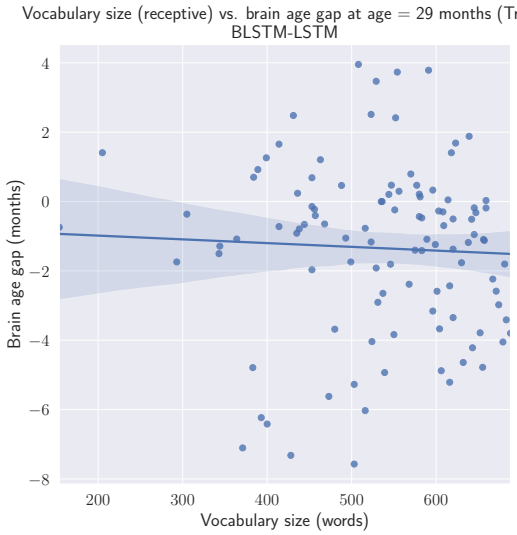
Figure 140: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the test set.



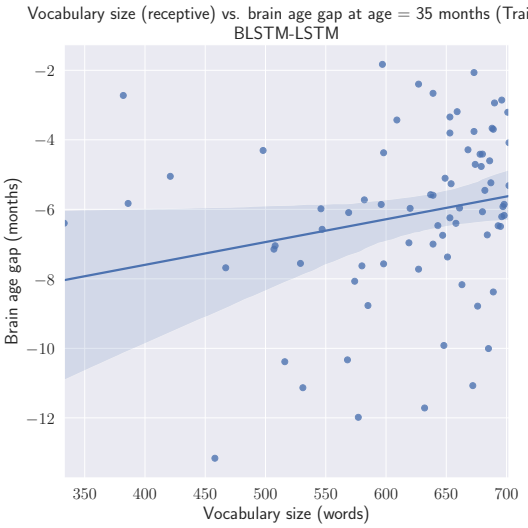

(a) Age = 17 months




(b) Age = 23 months

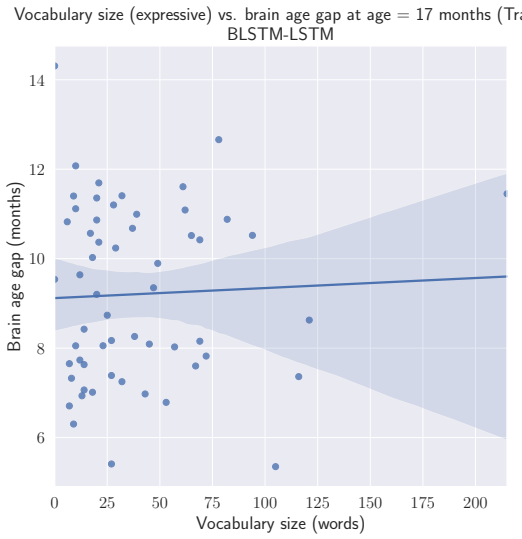



(c) Age = 35 months

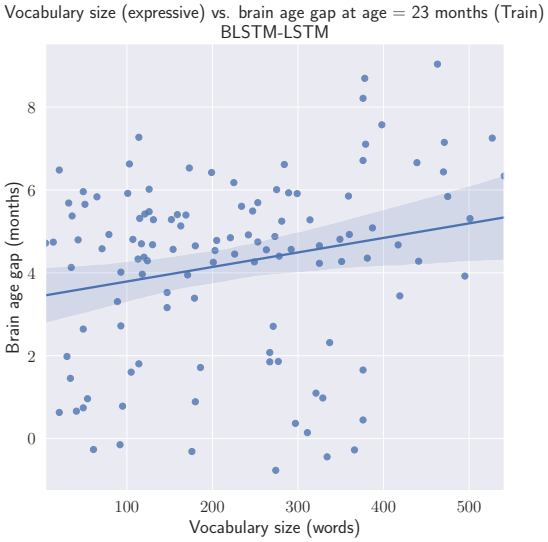



(d) Age = 35 months

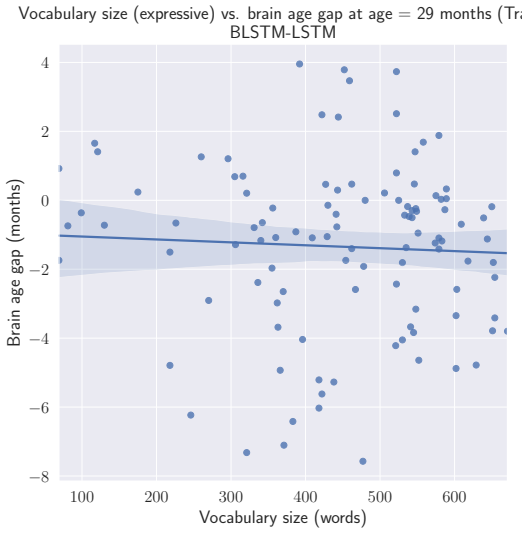
Figure 141: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the train set.



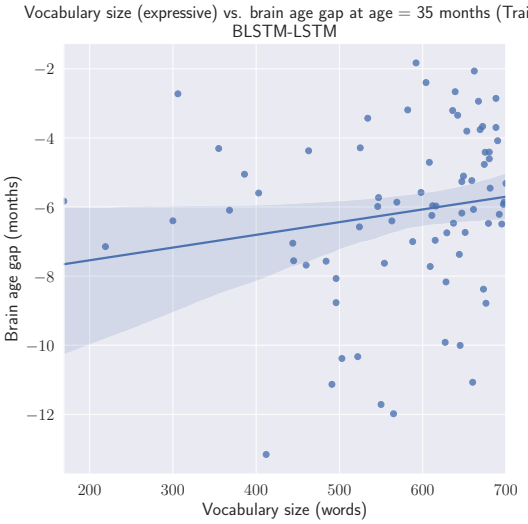

(a) Age = 17 months




(b) Age = 23 months

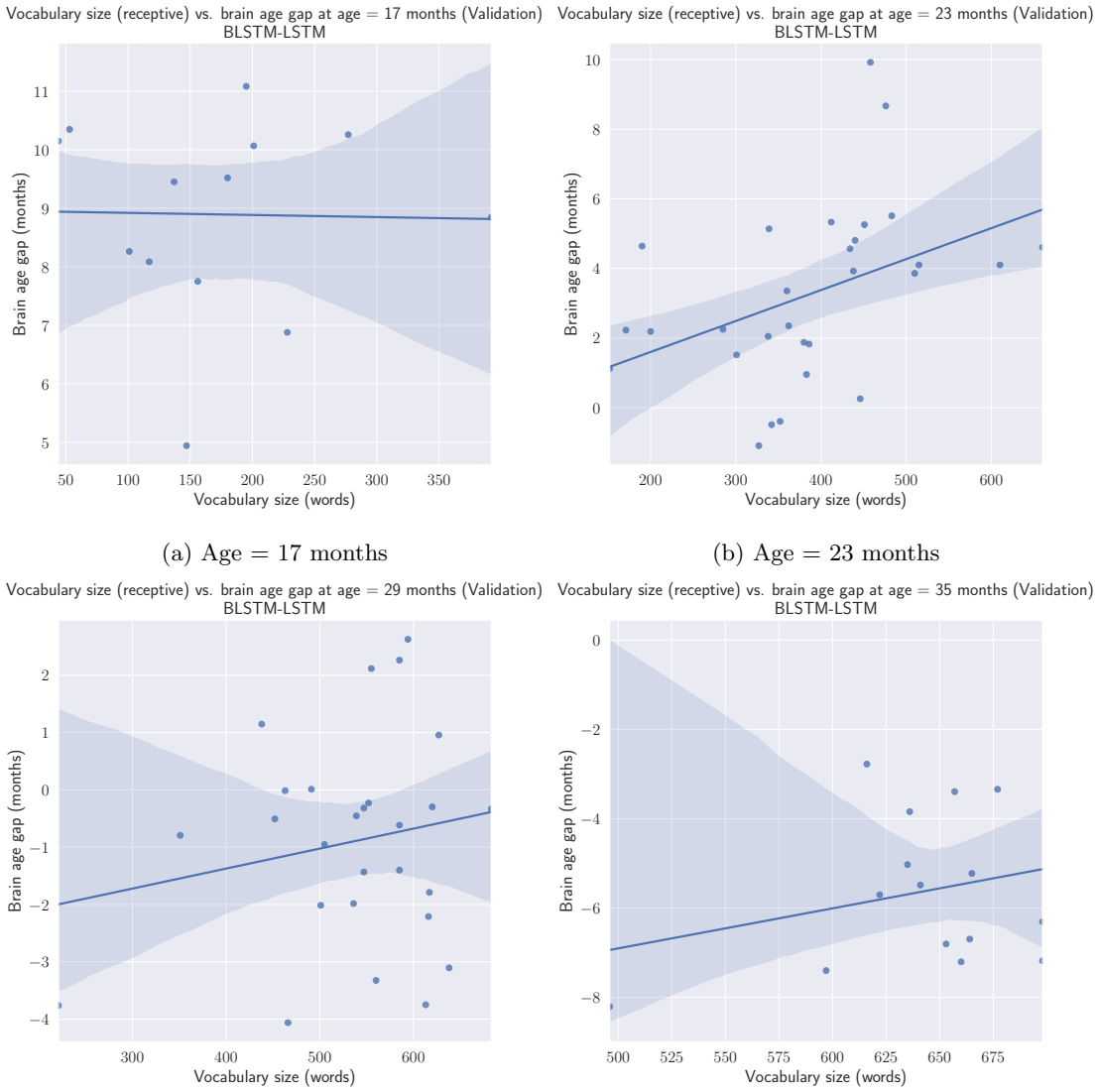



(c) Age = 29 months




(d) Age = 35 months

Figure 142: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the train set.



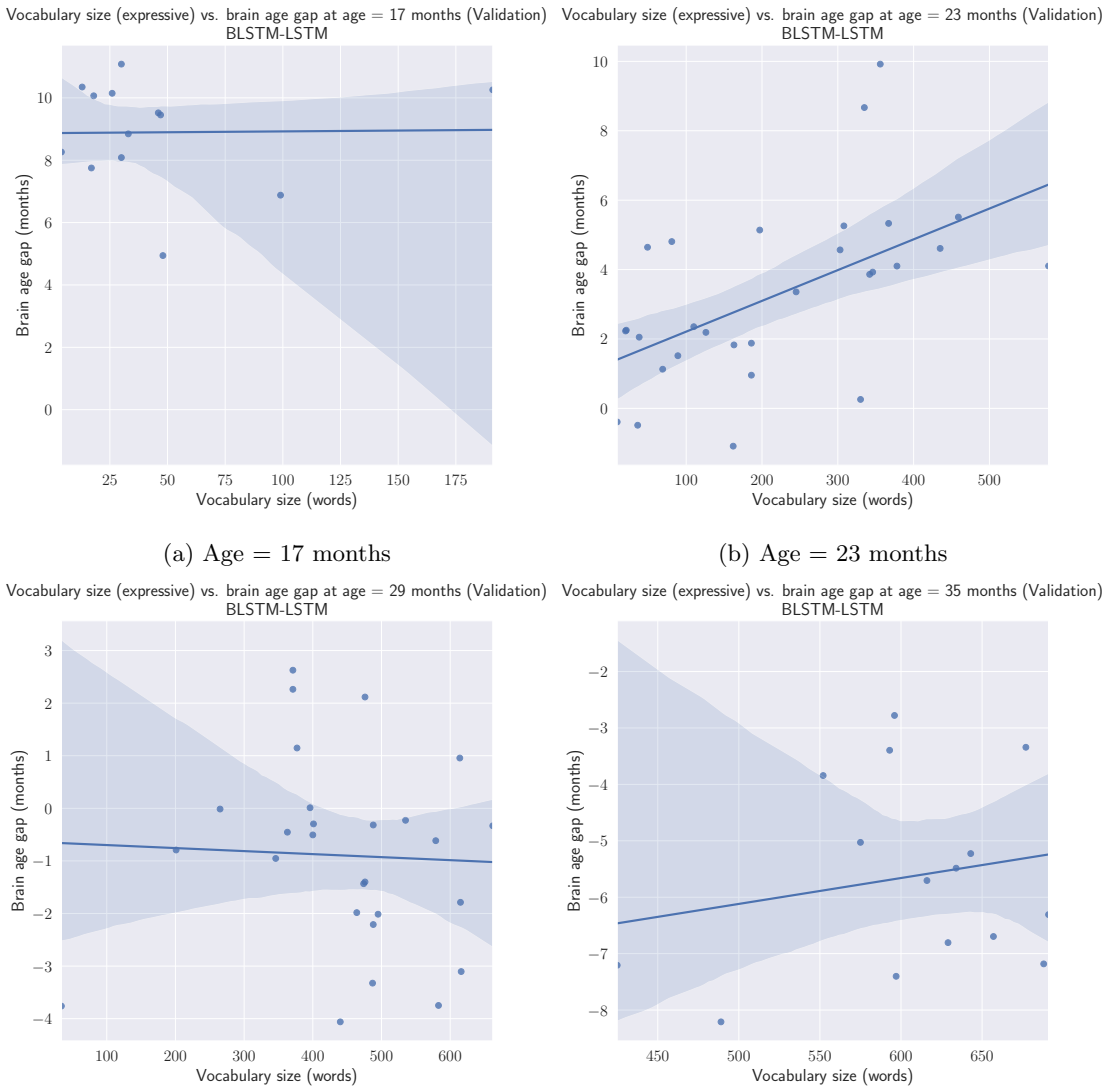

(a) Age = 17 months

(b) Age = 23 months

(c) Age = 35 months

(d) Age = 35 months

Figure 143: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the validation set.




(a) Age = 17 months

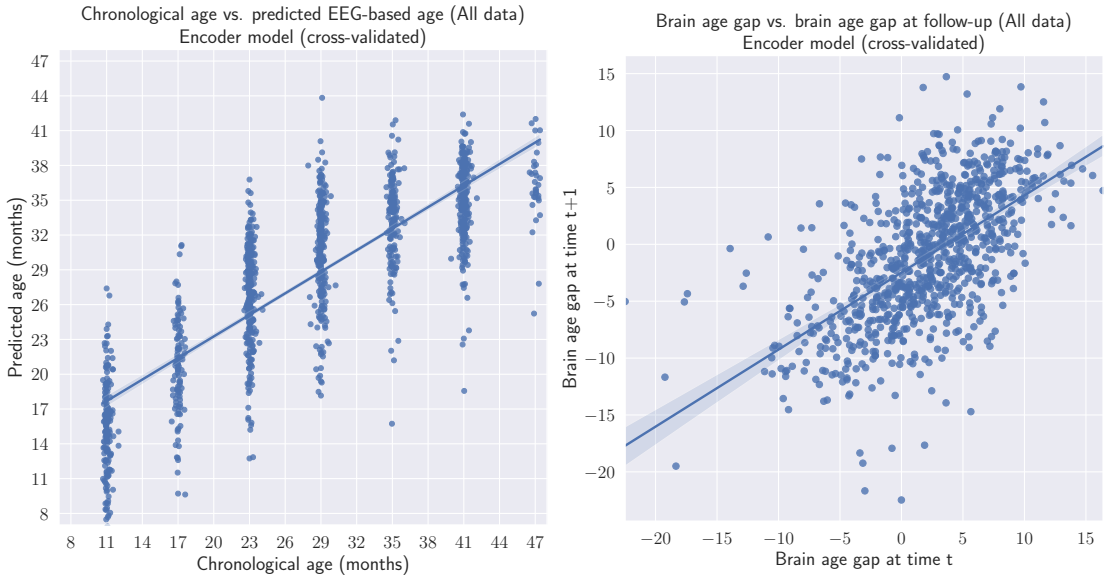
(b) Age = 23 months

(c) Age = 29 months

(d) Age = 35 months

Figure 144: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the BLSTM-LSTM model's brain age predictions on the validation set.

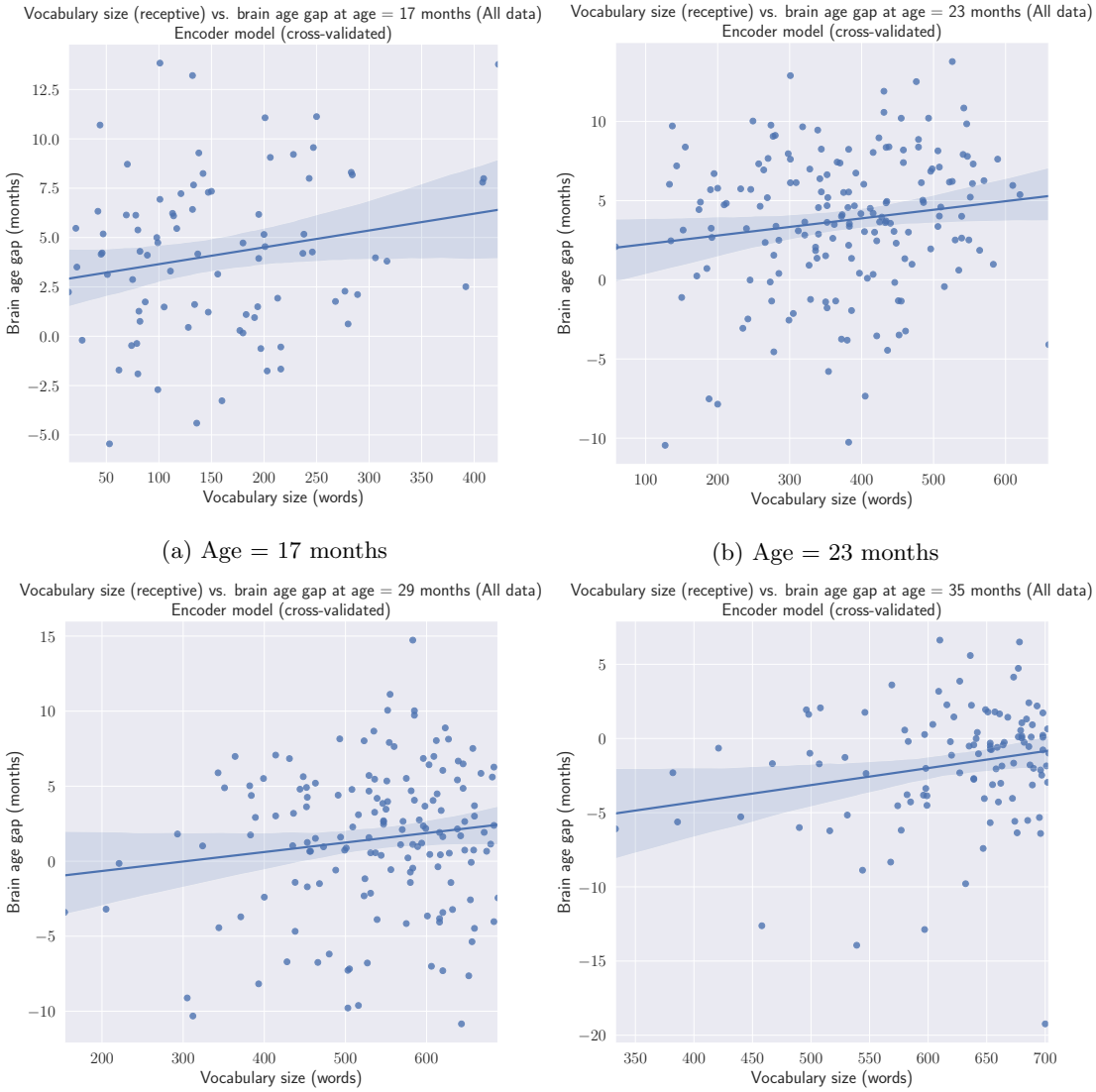
## Cross-validated Encoder model




(a) Predicted age vs. chronological age

(b) Error stability

Figure 145: The predictions compared to the true chronological ages of the subject (left), and the brain age gap at time $t$ compared to the brain age gap at follow-up, time $t + 1$ (right).



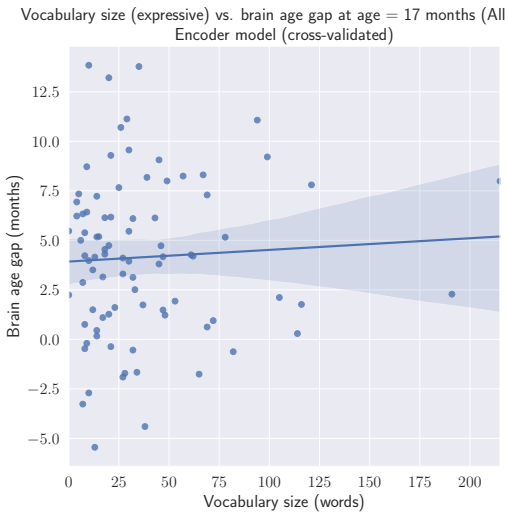

(a) Age = 17 months

(b) Age = 23 months

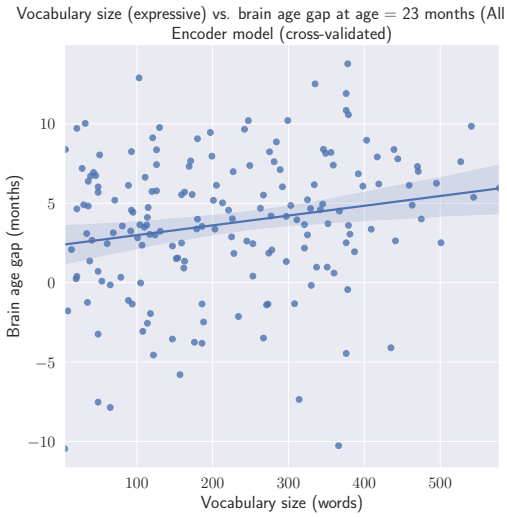
(c) Age = 29 months

(d) Age = 35 months

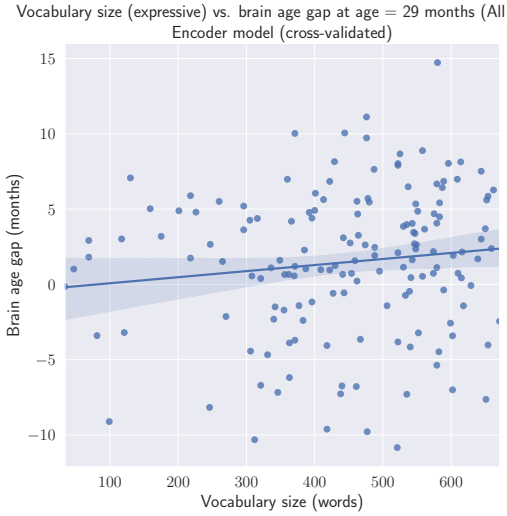
Figure 146: Vocabulary size (receptive) compared to the brain age gap of the subject at different ages, using the cross-validated Encoder model's brain age predictions on the full data set.



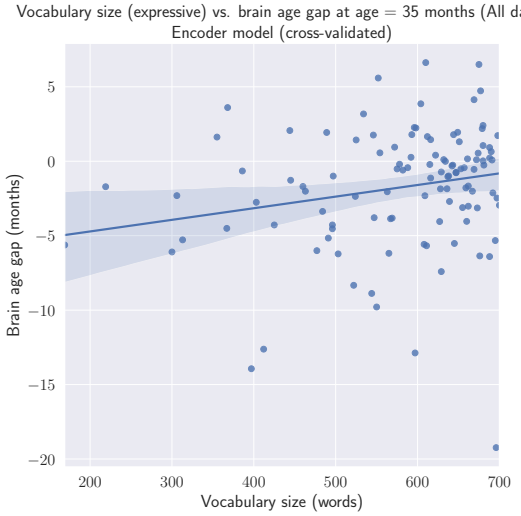

(a) Age = 17 months




(b) Age = 23 months




(c) Age = 29 months




(d) Age = 35 months

Figure 147: Vocabulary size (expressive) compared to the brain age gap of the subject at different ages, using the cross-validated Encoder model's brain age predictions on the full data set.

## Weights visualization

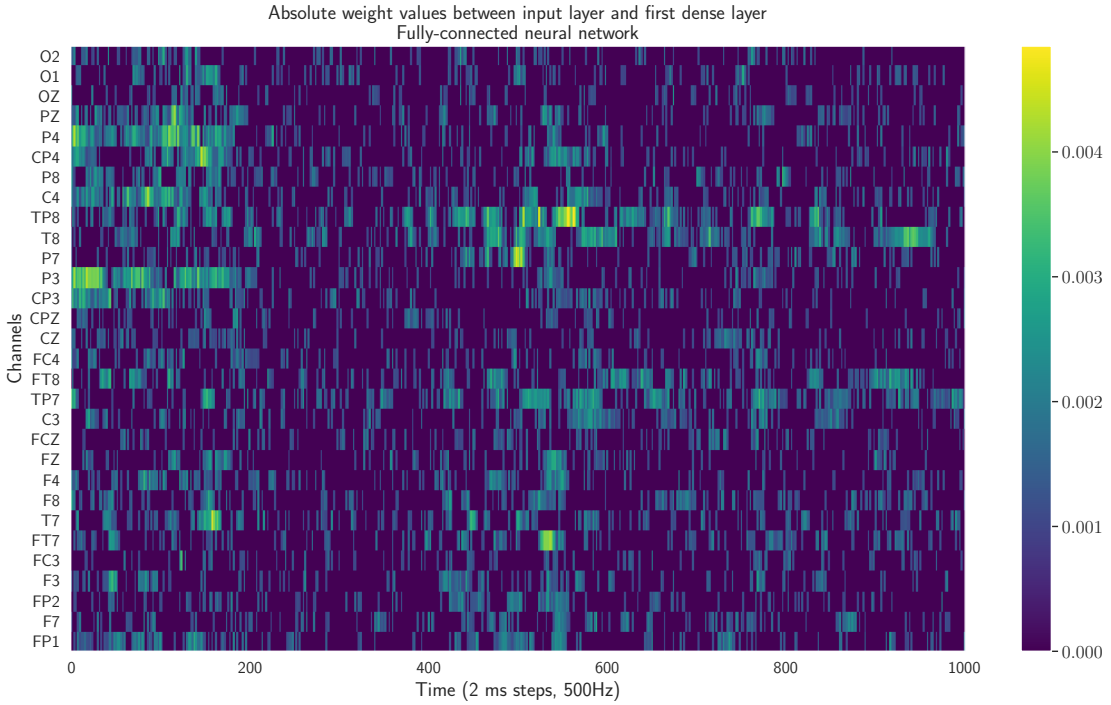



Figure 148: The mean absolute weight for each connection between the input features and the first dense layer of the trained fully-connected neural network. Clipping has been applied, weights below 0.001 are set to 0 for a clearer image.

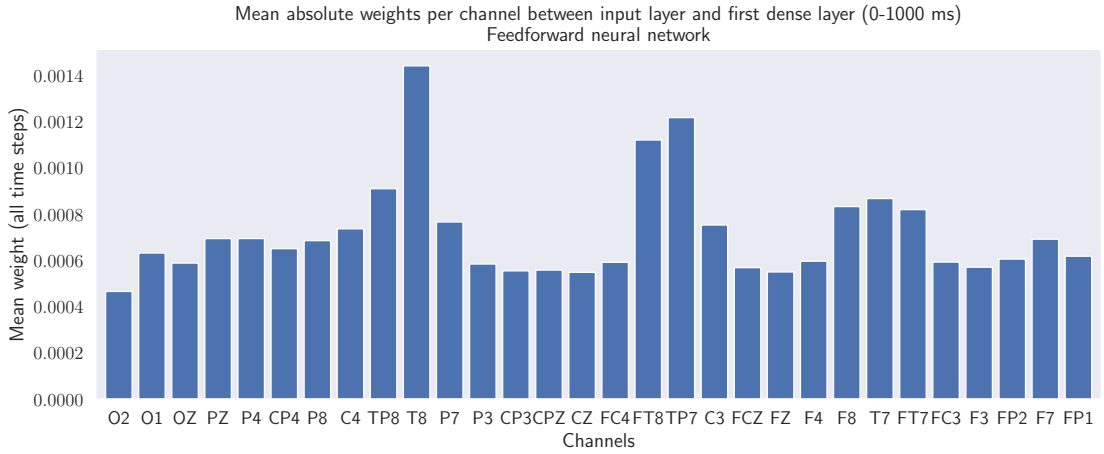



Figure 149: The mean absolute weight per channel (average across time steps) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

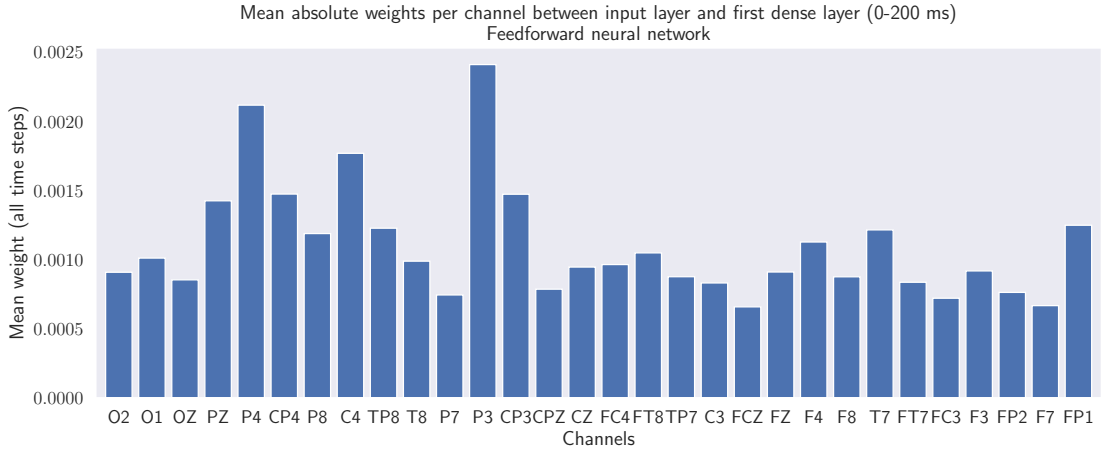



Figure 150: The mean absolute weight per channel (average over time steps between 0-200 ms) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

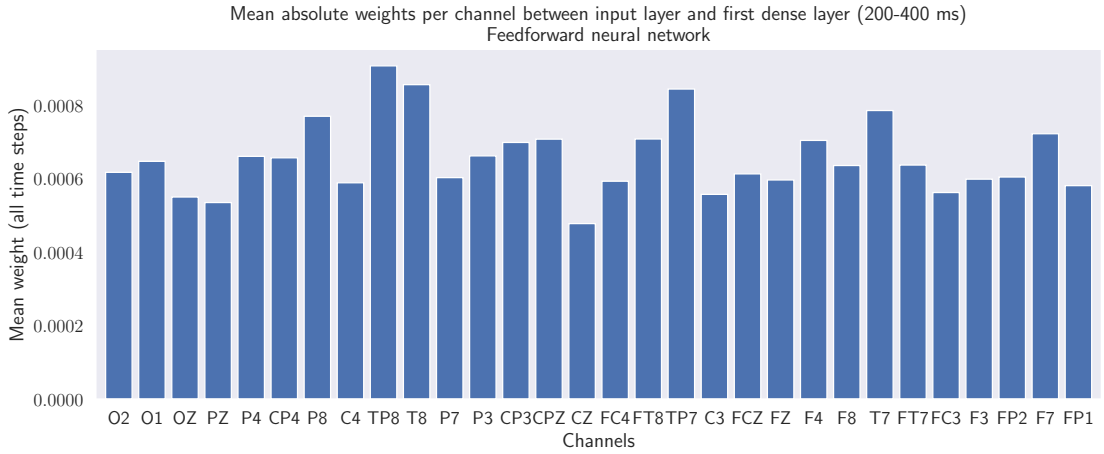



Figure 151: The mean absolute weight per channel (average over time steps between 200-400 ms) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

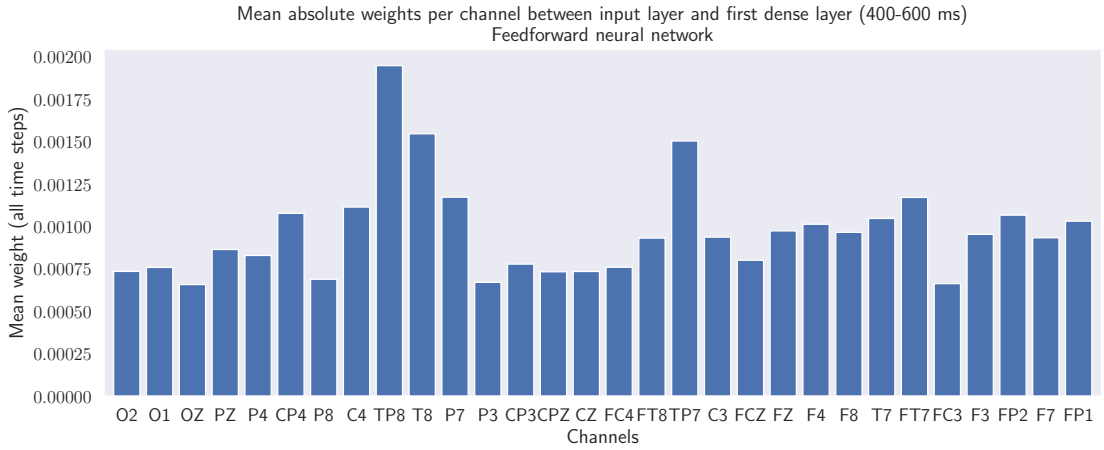



Figure 152: The mean absolute weight per channel (average over time steps between 400-600 ms) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

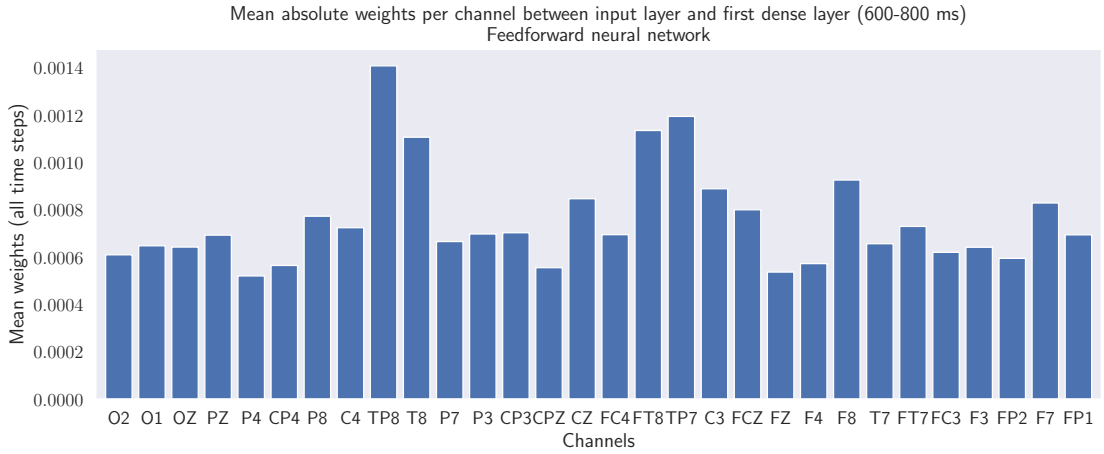



Figure 153: The mean absolute weight per channel (average over time steps between 600-800 ms) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

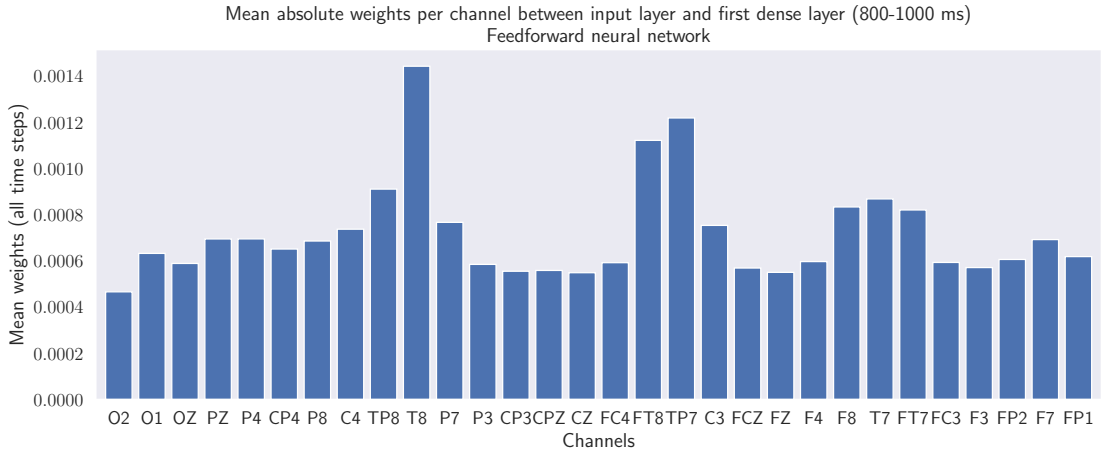



Figure 154: The mean absolute weight per channel (average over time steps between 800-1000 ms) for each connection between the input features and the first dense layer of the trained fully-connected neural network.

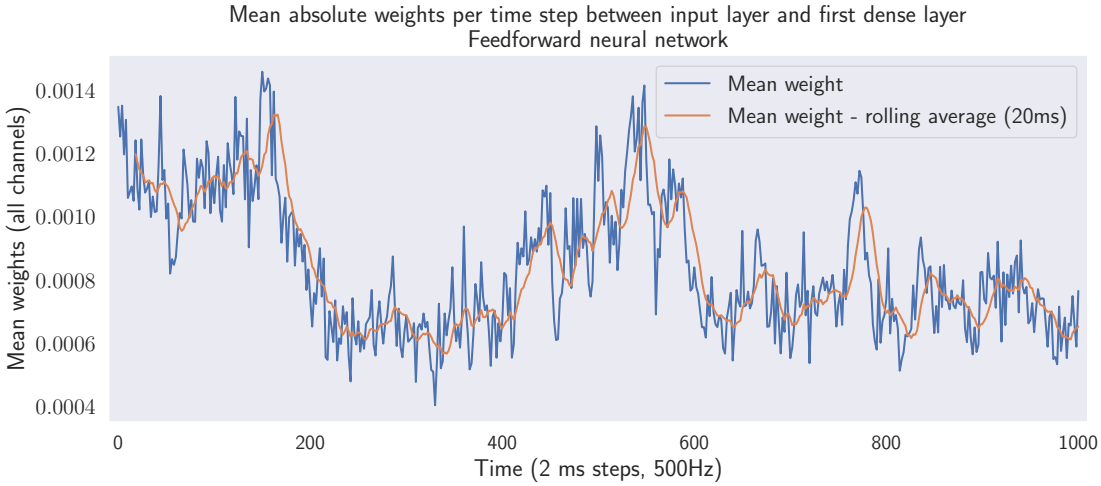



Figure 155: The mean absolute weight per time step (average across all channels) for each connection between the input features and the first dense layer of the trained fully-connected neural network.