ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Econometrics & Management Science
Specialization: Econometrics

Using Bayesian Neural Networks and multi-task learning to predict the clinical disease markers of MS patients and form a recommendation regarding the possible therapy switches.

Author: Marieke de Schepper (411826)
Supervisor: prof.dr. R. Paap
Second assessor: prof.dr. D Fok
Supervisor PwC: Tom Peters
Date: 28-1-2021

Abstract This research uses Bayesian Neural Networks (BNNs) to simultaneously predict two clinical markers for the disease course of multiple sclerosis (MS) patients. We examine the number of relapses and the confirmed disability progression (CDP). We construct one combined BNN, which predicts the two clinical markers simultaneously, and two single BNNs that predict the markers separately. The research uses a synthetic data set which contains demographic and clinical information about patients who decide to switch to another therapy. By modelling the two markers simultaneously, we can potentially benefit from the advantages of multi-task learning. BNNs are flexible models that require no assumptions on the data, which makes them suitable models to predict the two markers simultaneously as the relationships between the clinical markers are complex. Since BNNs can easily be applied to form an ensemble of NNs, we can form a distribution of recommendations which can express the uncertainty of the model regarding the possible therapy switches. The results show that the combined BNN does not consistently improve the performance, compared to the single BNNs. Next to that, less complex generalized linear models outperform the BNNs. Furthermore, we observe that the uncertainty of the BNN with respect to the recommended therapy switch is quite large. It is possible that the flexibility of the BNN is not fully employed as the data set is relatively small and the BNN cannot learn all complex patterns in the data. Although the BNNs do not improve the performance in this particular application, the characteristics of the BNN remain valuable in the application of medical decision making and it is interesting to further investigate the performance of the BNN in other applications with larger data sets.

Keywords: Bayesian Neural Networks, Multi-task learning, multiple sclerosis, medical decision making

Contents

1 Introduction

Multiple sclerosis (MS) is a chronic neurological disorder that damages the central nervous system (CNS). Relapsing remitting multiple sclerosis (RRMS) is a type of MS in which patients experience relapses, during which the symptoms of the disease worsen for a period of time, after which they partially improve again. Disease modifying drugs (DMDs) aim to slow down the progression of the disease on the long-term and reduce the number of relapses on the short-term. Research provides prove that several DMDs are effective in both matters (Goodin et al., 2002). However, many of those conclusions are drawn on group level, while it is shown that MS is a highly heterogeneous disease regarding the disease course and treatment response of patients (Derfuss, 2012). Even if DMDs are effective on average, individual patients can show no or adverse responses to the drugs (Miller et al., 2008). It is evident that DMDs are destructive for patients who suffer from adverse responses. However, ineffective DMDs are damaging for patients without any response as well since ineffective medication might permit the disease to progress more rapidly and cause permanent damage. Research shows that the disability of patients progresses faster if patients have more relapses in the first two years after the onset of RRMS (Scalfari et al., 2010). Next to that, it is shown that the damage of axons, which are essential features of the CNS, is most prevalent in the first years of the disease and may cause irreversible disabilities (Trapp et al., 1998). As a result, it is highly relevant to treat patients with effective DMDs to slow the progression and symptoms of MS.

Apart from the progression of the disease, ineffective DMDs are problematic as the social and economic costs of MS are substantial, making ineffective treatments expensive. Kobelt et al. (2006) describe that there are multiple direct costs such as the costs of medication, health care or home care. Moreover, the indirect costs of MS are of great importance as well. Many MS patients are no longer able to work, creating a large loss of productivity. This is a sizable cost, especially since many patients develop the disease early in life, while being in the most productive times of their lives. Although the costs vary between countries, the research of Kobelt et al. (2006) estimates the average costs per year for a European MS patient with a severe level of disability at 62000 euros. Both direct and indirect costs increase with the level of disability, amplifying the importance of

finding an effective DMD to slow down disease progression. Finally, MS influences other aspects that are more difficult to assess through monetary units. The research of Kobelt et al. (2006) shows that the quality of life reduces for MS patients. MS is a burdensome disease, both physically and mentally. Patients often suffer from pain, stress or anxiety. Therefore, it is essential to eliminate the distressing process of finding suitable DMDs through trial and error as much as possible.

The need for effective medicine on an individual level resulted in more research in personalized medicine (Schork, 2015). Personalized medicine tailors the recommended medication to individual characteristics such as gender, genetic information or previous disease course. Hence, it assumes that the treatment responses of individuals or groups of individuals can differ. Models that can capture the different treatment responses on an individual level are highly relevant for MS, since the heterogeneity of treatment responses makes it problematic to assign all patients to an effective 'one-fits-all' DMD. Models that account for individual information could explain the differences in treatment responses and generate more accurate predictions of the disease course of individuals. Based on these predictions, the expected effectiveness of the DMDs can be evaluated per patient. However, in order to correctly evaluate the effectiveness of DMDs, it is crucial to find disease markers that accurately represent the state of the disease. This remains challenging as many aspects of MS are not fully understood. The cause of MS is unknown and the search for suitable biological and clinical disease markers is continuing (Reich et al., 2018). There are several biological and clinical markers for RRMS that are generally accepted in practice. Although biological markers such as MRI scans are less subjective than clinical markers, in practice it can be infeasible to measure them frequently. On the contrary, clinical markers are relatively easy to measure, which causes them to be more available. Nevertheless, none of the existing clinical markers completely describe the activeness and progression of MS on its own. As Goodin et al. (2002) describe, it is uncertain which short-term clinical markers have the highest correlation with the aim of slowing down the long-term progression of MS. Hence, the authors state that research that uses multiple disease markers to evaluate the effectiveness of DMDs, should be considered as more robust evidence. It is thus relevant to base the choice of recommended DMD on multiple disease markers. Jointly, multiple markers possibly give a more complete representation of the disease course of patients, which could improve the evaluation of the effectiveness of DMDs.

The research of Stühler et al. (2020) is an example of personalized medicine and uses multiple disease markers to make treatment decisions for RRMS patients. The authors use real-world data from MS patients who are switching to another therapy cycle with a different DMD. The research examines the individualized prediction of two clinical disease markers. Two separate models aim to predict either the number of relapses or the confirmed disability progression (CDP) during the new therapy cycle. The CDP is a binary variable that represents the progression of disability, based on changes in the Expanded Disability Status Scale (EDSS). The EDSS ranks the disability of a patient on a range from 1 till 10. The two models of Stühler et al. (2020) incorporate information about the individual characteristics and the previous disease course of the patients and are used to recommend

a therapy switch to a particular patient who is starting a new therapy cycle. This recommendation is based on the predicted outcomes of the two models. The two models rank the outcomes of the possible therapy switches. For example, the expected number of relapses is computed for each therapy switch, which can show a preference regarding certain therapies. However, since the models consider the two clinical markers separately, it is possible that the preference for certain therapies based on the expected outcomes of the two clinical markers differs. Naturally, one clinical marker can be assigned as dominant marker in the choice between the different therapies. However, this introduces difficulties since it is preferred to use multiple clinical markers, as discussed before. Furthermore, the separate models do not include the other marker in their predictions. In other words, the predicted value for CDP does not depend on the number of relapses. If the two variables are related, this could mean that valuable information is lost. Therefore, it is important to develop models that produce one recommended therapy switch, based on multiple possibly related disease markers.

Neural Networks (NNs) are flexible models that can capture complex structures and patterns in the data (Nielsen, 2015). This makes NNs useful models for personalized medicine and MS, as MS is a heterogeneous disease with complex relations. Next to that, NNs are capable of predicting multiple types of variables at once without making explicit assumptions about the relationship between the multiple variables. This is favorable since Scalfari et al. (2010) show that the relationship between the number of relapses and the EDSS (which underlies the CDP variable of Stühler et al. (2020)) is complex. Nevertheless, if the two clinical markers of the research of Stühler et al. (2020) are simultaneously predicted in one NN, the model will still result in distinct predictions for both markers. Ensemble learning techniques can be used to combine the two predictions into one recommended therapy switch. Such techniques create multiple models that all differ slightly and can be used to form different predictions for the same observation. This does not only generalize the model, but it also results in a distribution of predictions. In this case, the ensemble of models results in many different predictions of the two clinical markers. Consecutively, the predictions all result in a different recommendation regarding the best possible therapy switch. Finally, the majority vote can be used to recommend one optimal therapy switch. Unfortunately, the computational time that is needed to train a NN makes the ensemble learning techniques infeasible in practice.

Bayesian Neural Network (BNN) are a type of NN that train the network by estimating the probability distributions of the weight parameters instead of their point estimates. The estimated probability distributions can be used to create an ensemble of NNs, by drawing a set of weights from the distributions that together form a new model. Thus, one trained BNN can be used to create an infinite number of NNs (Blundell et al., 2015). On the contrary, regular NNs estimate point estimates of the weights, which means that in order to generate an ensemble of models, each model in the ensemble is trained separately. Apart from this practical convenience, BNN have another feature that is favorable in its application in personalized medicine. For regular NNs, it is not possible to address any uncertainty in the outcomes of the model. However, in BNNs the

probability distributions of the weights make it possible to express uncertainty regarding specific predictions. This is highly relevant information since the decision to switch to another therapy is an important decision, in which the opinion of the patient is of influence. If the model shows to be uncertain in its choice between several possible therapy switches, this might affect the decision of a patient. For this reason, it is relevant to recognize and quantify the uncertainty in the predictions. This further argues for the use of a BNN instead of a regular NN.

In this research we investigate if BNNs can be used to extend the research of Stühler et al. (2020) on the recommendations regarding the possible therapy switches for MS patients. First, we examine if BNNs can accurately predict the number of relapses and the CDP, on an individual level. We employ two separate BNNs to predict the two clinical markers. Additionally, we examine if a combined BNN, that is used to predict the two clinical markers simultaneously, is able to improve the performance. We compare the performance of the separate and combined BNNs with the models of Stühler et al. (2020) and two benchmark models, that aim to predict the same clinical markers. Finally, we examine if the combined BNN can be applied to form a distribution of recommendations, based on both clinical markers, which can be used to recommend one optimal therapy switch for a particular patient.

The rest of this research is structured as follows. In section 2 we discuss previously done research concerning the two clinical markers. Next to that, we describe the possible challenges of NNs in personalized medicine and discuss the advantageous characteristics of BNNs and the combined BNN. Afterwards, section 3 addresses the data used in this research. Firstly, we discuss the original data set of Stühler et al. (2020), after which we introduce the synthetic data set that is employed in this research. Section 4 describes the methodology of the research. The intuition and training of regular NNs is explained in section 4.1. Next, the methodology of BNNs, including the probability distributions of the weights, is introduced in section 4.2. Section 4.3 discusses the choices made regarding the architecture of the two separate BNNs and the combined BNN. Subsequently, the methods used to evaluate the performance of the models are discussed in section 4.4. Finally, we describe how the combined BNN can be applied to form a distribution of recommended therapy switches in section 4.5. The results are discussed in section 5. The first three sections address how several choices regarding the architecture of the BNNs influence the performance of the BNNs. Afterwards, the performances of the separate BNNs and the combined BNN are compared with each other and the benchmark models in section 5.4. Section 5.5 presents how the combined BNN can be applied to recommend a therapy switch for four patients. Finally, the conclusion of this research is addressed in section 6.

2 Literature review

DMDs are a more recent type of drug that regulate the auto immune system and aim to postpone the progression of a patient's disability by reducing the number of lesions in the CNS and the

number of relapses. The effectiveness of DMDs would be most visible in tissue samples of the CNS. However, this is highly impractical which means that other disease markers are often considered for the evaluation of DMDs (Reich et al., 2018). As discussed before, clinical markers are frequently measured which makes them accessible for research in personalized medicine. Nevertheless, it is preferred to consider multiple clinical markers in order to combine their strengths. In this research we will consider the number of relapses and the progression of disability, that is, the CDP. Research shows that the relationship between these markers is not consistent through the whole disease course of a patient. First of all, the research of Confavreux et al. (2003) examines the effect of several clinical markers on the time it takes to reach irreversible disability. The clinical markers that are considered in the research are all measured shortly after the onset of the disease. The authors consider the progression of disability through the EDSS. For this, the time between the onset of MS and reaching an EDSS of 4, 6 and 7 is assessed. The authors observe that a higher number of relapses in the first 5 years shortens the time till these levels of disability are reached. However, once these levels are reached, the number of relapses in the first 5 years are no longer of influence on the time it takes to reach higher levels on the EDSS. In other words, while there seems to be a relationship between the number of relapses in the first 5 years and the early progression of MS, this relation cannot be observed on the long term for more disabled patients. The findings of Scalfari et al. (2010) as well illustrate the uncertainty in the relationship between the progression of MS and the number of relapses. The authors observe that the time from RRMS to secondary progressive MS (SPMS) is shorter if patients experience more relapses in the first two years after the onset of MS. SPMS is a stage of MS that often follows after RRMS. In this stage, the disability of a patient steadily worsens with time which means that the transition to SPMS is undesirable (Rovaris et al., 2006). In addition, the results show that the probability of transitioning from RRMS to SPMS increases for patients with more relapses in the first two years. Next to that, the time from the onset of MS till more severe EDSS levels of 6, 8 and 10 is shorter for those patients. On the contrary, the research shows that the number of relapses between the third year after the onset of MS and the transition to SPMS has no effect on the time till these EDSS levels are reached.

The research of Confavreux et al. (2003) and Scalfari et al. (2010) shows that the relationship between the number of relapses and the progression of disability is not that straightforward and might alter throughout the disease course of a patient. Next to that, it is unclear if the relationship is causative or if the two are associated through another factor. Even if there is a causative relationship, it is unsure which way this causation flows. As the relationship between the number of relapses and progression of disability is complex and not constant over time, care should be taken when making assumptions about the joint structure of the two markers. For this reason, NNs can be appropriate models for the joint prediction of the markers. Parametric models would require the researcher to make assumptions about the multivariate distribution of the two markers, in order to estimate them jointly. On the contrary, NNs can be constructed in such a manner that the models can be used to predict multiple different variables simultaneously, without making explicit

assumptions about the relationship between the different variables (Caruana, 1997). Hence, we could construct and train a NN such that the two clinical markers can be predicted simultaneously, without the need to define the type of relationship between the two markers.

The use of NNs in personalized medicine is researched in several fields. For example, Itchhaporia et al. (1996) report that NNs are applied to different areas of cardiovascular medicine, such as making the correct diagnosis or optimizing the dozing of drugs. Next to that, Zhou et al. (2002) use NNs to identify lung cancer cells in images. For this purpose, the authors develop two ensembles of NNs. The first ensemble of models is applied to decide if a cell can be classified as a normal cell or a cancer cell. Subsequently, the second ensemble is used to classify the type of cancer in the cancer cells. The model incorrectly classifies cancer cells as normal cells in 2,7% of the cases. Moreover, Bejarano et al. (2011) develop personalized models to predict the short-term disease course of MS patients. They examine several classification models, among which NNs and logistic regressions. The input variables used for the models include personalized information about the patients, including variables about their previous disease course. The models aim to predict three clinical disease markers: the change of the EDSS in 2 years, a binary variable for disability progression and a binary variable indicating if a patient experiences a relapse or not. When comparing the predictions of the models, the authors conclude that the overall performance of the NNs is better than that of the regression models. Nonetheless, the authors note that the precision of the predictions made by the NNs might be too low for the application in medical decision making as this requires high performing models. Apart from the limited performance of the models, it should be noted that the predictions of the three markers in the research of Bejarano et al. (2011) are made by using three separate models with three different output variables. This makes it difficult to base the medical decisions on all three markers while this is preferable.

The wide application of NNs in personalized medicine can partly be explained by the flexibility of the models. NNs can detect both linear and nonlinear patterns in the data. Next to that, the models require little assumptions on the data (Nielsen, 2015). This can be favorable in the application of medicine, since not all characteristics of diseases are completely understood. As discussed by Reich et al. (2018), this is as well the case for the disease course of MS. However, NNs have some disadvantages which might be of importance in the context of personalized medicine. First of all, it is recognized that NNs often need large amounts of data to take advantage of the flexibility in the network. If the training data set is too small, it is possible that the bias of the predictions increases as the unknown relationships are not properly learned by the NN. This problem is often solved by reducing the depth and complexity of the NN. However, this reduction as well diminishes the flexibility of the NN (Neal, 2012). Secondly, the interpretability of NNs is lower compared to statistical methods like ordinary least squares. Especially for deep NNs, it is difficult if not impossible to explain why certain relations are found by the model. This does not have to be an issue if the model performs well. However, depending on 'black-box' models which are hard to explain, such as NNs, can be problematic if the predictions are not that accurate.

As Dilsizian & Siegel (2014) explain, mistakes made by technology might be less accepted than human mistakes made by doctors. On the other hand, the authors note that several human biases result in unnecessary mistakes made in medical decisions. NNs are potentially less susceptible for these biases and can be used as an additional consult to the decision making of doctors. For this reason, the models should not be disregarded because of their black-box nature. Third, NNs have an absence of any measure of uncertainty in the models (Blundell et al., 2015). This is a large issue in the context of personalized medicine as it leads to two possible problems. First of all, patients can dislike some treatments because of side effects. If such a treatment is recommended by the model, the patient or doctor will probably want to assess the certainty of the recommendation. Furthermore, NNs can experience problems while handling new data that is too different from the training data (Nielsen, 2015). This issue, called overfitting, occurs when the model fits the training data too well, whilst it is not able to explain new data. The NN will always return a prediction or classification, even when the new data is entirely different. For example, if a NN is trained to classify pictures of dogs versus cats and it is given a picture of a bird, it will nonetheless classify this picture as either dog or cat. The lack of uncertainty in this classification implies that the model is overconfident in its decision. Ideally, the model should express its uncertainty when classifying the picture of the bird. Since many diseases are heterogeneous and new patients can have more extreme symptoms compared to the training data, this imposes issues. Hence, since the decisions made by the model can be highly influential on the patient's life, the uncertainty in the predictions needs to be visible. Finally, an additional complication of medical data in the evaluation of NNs is that the importance of identifying different type of patients is often unequal. For example, it is more important to correctly identify a patient with MS, than it is to identify a patient who does not have MS. In addition to this varying importance of correct classifications, the occurrence of different type of patients is often unequal. A data set in which the different values of the output variable occur with unequal frequency, is referred to as an imbalanced data set. For example, it is much more common to experience no relapses in the time span of a year than it is to have relapses. When a NN is trained on an imbalanced data set, Saito & Rehmsmeier (2015) explain that the performance measures should be chosen with care, as traditional measures can be deceptive and lead to incorrect conclusions. For example, the accuracy shows the percentage of individuals that are correctly classified by the model. This measure can be uninformative in imbalanced data sets, as a high accuracy might correspond to a model which solely estimates the more frequently observed output value. However, this issue is not inherent to NNs, but as well arises in different classification models.

BNNs have the potential to be more suitable NNs for the purpose of personalized medicine and solve multiple of the previously described issues concerning NNs. BNNs aim to estimate the probability distribution of the weights in the network instead of the point estimates (Neal, 2012). This implies that the uncertainty of the estimated weights is incorporated in the model. The uncertainty of the weights can be used to express the uncertainty of the BNN regarding a

particular observation (Blundell et al., 2015). For example, if the model is used to predict the disease progression for a new patient, whose previous disease course has values that are out of the range of the training data, the outcomes of the BNN can possibly express the uncertainty about the predictions. This implies that a BNN will be less overconfident than a NN in these situations. This characteristic is advantageous in the context of personalized medicine. Next to that, while the interpretability of the BNN itself might not improve compared to a NN, the output of a BNN is more informative. This is explained by the fact that BNNs can easily be transformed into an ensemble of models, which can be applied to form a distribution of predictions (Blundell et al., 2015). As these predictions can express the uncertainty of the model, the outcomes contain valuable information. In addition, the use of an ensemble of models can be seen as model averaging, which can greatly improve the generalization of the model and reduce overfitting. Finally, the research of Neal (2012) shows that it might not be needed to reduce the complexity of BNNs for smaller data sets. This implies that when the training data is relatively small, BNNs might be a better choice than NNs to describe the data. However, the results of Neal (2012) are based on the use of specific prior distributions in the training process of the BNN. To conclude, BNNs have some advantageous characteristics for the application in personalized medicine.

Furthermore, the proposition to model multiple clinical markers simultaneously can be beneficial as well. A model which combines multiple output variables can potentially result in more generalized models. Caruana (1997) discusses the concept of multi-task learning, which allows several types of variables to be learned in one model. Modelling all variables in separate models means that information which could improve the training of the model is ignored. On the contrary, by training one single model for multiple output variables simultaneously, the output variables seem to learn from each other. As Caruana (1997) describes, multi-task learning can improve the generalization of the model since the errors of the estimated output variables can be considered as extra noise to the other output variables. In separate models, the network can more easily learn the patterns in the training data which can lead to overfitting. However, if the network has to fit two types of output variables, this could reduce the problem of overfitting. This advocates the use of the combined model which predicts the two clinical markers simultaneously. To conclude, the use of BNNs combined with multi-task learning has the potential to result in models that are applicable in the sensitive decision-making process of personalized medicine.

3 Data

In this section we discuss the original data set which contains the variables that are used to model the two clinical markers discussed in section 2. Subsequently, we address the sampling procedure of the synthetic data set, which is based on the original data of Stühler et al. (2020). Finally, we investigate the validity of the synthetic data and examine several of its characteristics.

3.1 Original data set

The research of Stühler et al. (2020) uses real-world data from the NeuroTransData (NTD) MS registry. NTD is a German network of physicians practicing in neurology and psychiatry. The MS database consists of a large number of variables, among which demographic and clinical variables. The data set in the research of Stühler et al. (2020) is collected in the period between 1999 and October 2019. The authors select the input variables based on clinical expertise and scientific research. Next to that, several inclusion criteria and quality checks are applied to each observation in the data (for detailed information on the criteria, see additional file 1 in Stühler et al. (2020)). The resulting data set is used to develop two models in their research. The output variables of those models are the two clinical markers discussed in section 2, that is, the *number of relapses* and the *CDP*. To compare the results of this research analysis with the research of Stühler et al. (2020), we use the same output variables. Moreover, except for the variable *clinical site*, our analysis employs the same input variables as Stühler et al. (2020), which are shown in table 1. The motivation for excluding the variable *clinical site* in our analysis is discussed in section 3.2. All patients in the data set are changing the DMD they are currently taking, meaning that a new

Table 1: Description of input variables of the data set.

Variable name	Description
Age	Age at start of the index therapy
Gender	Gender
EDSS	EDSS (measured at most 6 months before or 3 months after the start of the therapy cycle, and at least 84 days after a relapse)
Index therapy	DMD taken during the therapy cycle
Current therapy	DMD taken prior to the start of the therapy cycle
DMD count	Number of DMD taken prior to the start of the index therapy
Second-line	Whether a second-line DMD has been taken before the start of the index therapy
Relapse count	Number of relapses in the year prior to the start of the index therapy
Relapse distance	Time elapsed between the last relapse preceding the start of the index therapy and the start of the index therapy
Diagnosis distance	Time elapsed between MS diagnosis and start of index therapy
Index duration	Duration of the index therapy
Current duration	Duration of the current therapy
Clinical site	Clinical site where the course of MS is observed

The descriptions of the variables in this table are taken from Stühler et al. (2020).

therapy cycle is started. Since each patient in the data set decides to switch to another therapy, the models in this research are exclusively used for this purpose. In other words, the models are not applied to predict the disease course for patients who keep the same therapy. The variable *current therapy* refers to the original DMD of the patient, while the *index therapy* refers to the newly taken DMD. There are six different index therapies in the data set: Dimethylfumarat (DMF), Fingolimod

(FTY), Glatirameracetat (GA), Interferon-ß1 (IF), Natalizumab (NA) and Teriflunomide (TERI). The current therapies include these six DMDs, as well as the option that no previous DMD was taken (NoDMD).

The outcomes of the output variables *number of relapses* and *CDP* are measured during the time span of the new therapy cycle. During our analysis, we use the same definition for the variable *CDP* as given by Stühler et al. (2020). Stühler et al. (2020) base the definition of the binary variable *CDP* on changes in the EDSS of the patient. For patients with a previous EDSS measurement of 5.5 or lower, an increase of 1 point or higher on the EDSS will be characterized as CDP. For patients with a higher initial EDSS, the increase should be at least 0.5. In these cases, the *CDP* will take the value of one. Otherwise, its value will be zero. To ensure that the observed rise in EDSS is not due to a temporal relapse, the increase should last for at least three months and should be observed in at least one of the subsequent measurements as well.

3.2 Synthetic data set

In this research we use a synthetic data set, which ensures that the privacy of the patients is preserved. We develop the synthetic data in a manner that accounts for the underlying distributions and relationships of the original data set. This is done by using the original data for the variables *number of relapses*, *CDP* and *index therapy*. For each combination of these three variables, which as well exists in the original data set, we create separate data sets. For each combination $c \in (1, ..., C)$, the synthetic data set is created in the following manner:

1. The values of *number of relapses*, *CDP* and *index therapy* are set to the values of c in the original data set. In this manner, the existing relationships between these three variables is preserved.

2. The sample size of the data set depends on the proportion of individuals that have combination c in the original data set.

3. The remaining input variables are sampled by considering their distribution, conditional on combination c. In other words, each input value is drawn from the pool of values in the original data set, observed for the individuals with the current combination of *number of relapses*, *CDP* and *index therapy*. This maintains the relations between each input variable and the three variables of combination c.

Step 1 and 2 in the creation of the synthetic data set ensure that the values and proportion of the most important variables are equal to the original data set. For example, if 5% of the individuals in the original data set have zero *number of relapses*, a *CDP* of zero and switch to the *index therapy* DMF, the proportion of those individuals in the synthetic data set will be similar. On the contrary, if no individual in the original data experiences seven *number of relapses* in combination with a *CDP* of zero, this combination does not appear in the synthetic data set. In addition, the values

for the remaining input variables will be set to values that actually exist for the individuals in the original data set with the particular combination c.

The synthetic data set consist of 3007 individuals. The sampling procedure preserves the relationships between the variables *number of relapses*, *CDP* and *index therapy*, as they are equal to the observed values in the original data. In addition, the procedure aims to preserve the relationships between the input variables and the former three variables. This includes both linear as nonlinear relationships. It is important to verify if the synthetic data set is a valid representation of the original data set. For this purpose, we re-estimate the models of Stühler et al. (2020) with the synthetic data. This is done by comparing the mean squared error (MSE) of the models. Following the modelling procedure as explained in the research of Stühler et al. (2020), we re-create the two hierarchical Bayesian generalized linear models (GLMs) for the *number of relapses* and *CDP*. This is accomplished with the library `rstanarm` in R. Similar to Stühler et al. (2020), we use 10-fold cross validation to assess the MSE of the models. For this, we randomly split the synthetic data into 10 equal folds. Each fold is used as test data, while the 9 remaining folds are used to train the model. Subsequently, the average of the MSE over the 10 folds is compared. The results are shown in table 2. Note that the MSE's of the models constructed with the synthetic data set are

Table 2: Comparison of MSE between synthetic and original data.

	Synthetic data		Original data	
	CDP	Relapse	CDP	Relapse
MSE testing data	0.135	0.898	0.125	0.731
MSE training data	0.138	0.878	0.121	0.705

The models are estimated with the use of 10-fold cross validation, after which the average MSE is shown in this table.

consistently higher. This might be due to the sampling of the input variables. While step 3 in the sampling procedure ensures that the relationships between the input variables and output variables is preserved, the relationships between the input variables themselves is not considered. Although this is done to protect the privacy of the patients, it might cause a small difference in performance as some relationships are lost in the sampling. However, since the MSE's of the models are comparable and the use of a synthetic data set is preferred, we continue our analyses with the synthetic data set.

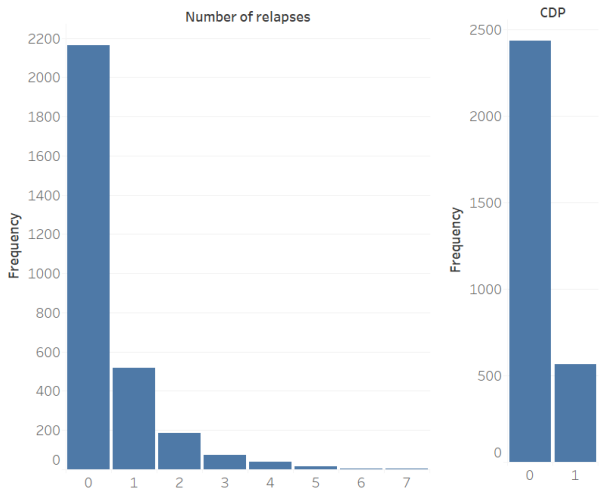
BNNs can only process numerical input values, which means that categorical variables need some type of transformation. One-hot encoding is regularly used for this purpose. It transforms all possible categories within a categorical variable into binary variables and uses the number 1 to represent to which category the individual belongs. For example, for the variable *index therapy*

this will imply the following transformation,

$$\begin{bmatrix} ID & Index\ therapy \\ 1 & DMF \\ 2 & IF \\ 3 & DMF \\ 4 & TERI \\ \dots & \dots \end{bmatrix} \rightarrow \begin{bmatrix} ID & DMF & FTY & GA & IF & NA & TERI \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

One-hot encoding is used to transform all categorical and ordinal variables. The values of the variables *age*, *DMD count*, *EDSS* and *relapse distance* are given in bins of values, which means that they need to be transformed through one-hot encoding as well. The ranges of these bins for the four variables equal: *age*: {(18,30], (30,40], (40,50], (50,inf]}, *DMD count*: {(0,1], (1,2], (2,3], (3, inf]}, *EDSS*: {(0,1.5], (1.5,2.5], (2.5,3.5], (3.5,10]} and *relapse distance*: {(0,0.25], (0.25,1], (1,3], (3,inf]}. The variables *current therapy*, *index therapy*, *age*, *DMD count*, *EDSS* and *relapse distance* are transformed, which increases the actual number of input variables for the BNN, that is, the number of input neurons. After the transformations, the input variables result in 35 input neurons. The variable clinical site is not added as input variable as it will create 70 extra input neurons using one-hot encoding. This large increase of input neurons substantially increases the complexity of the network, relative to the benefits of including the variable. Therefore, we choose to exclude it. Finally, the continuous input variables, *diagnostic distance*, *relapse count*, *index duration* and *previous duration* are standardized to have a zero mean and a standard deviation of one.

As discussed in section 2, it is important to employ proper performance measures if the data set is imbalanced (Jain & Nag, 1997). For this reason, we examine the frequency with which the different values of the output variables are observed. It is visible in figure 1 that the frequency with



(a) Number of relapses (b) CDP

Figure 1: Histogram for the *number of relapses* and *CDP*.

which the possible values of the output variables occur is quite uneven. Firstly, it is much more

common to experience zero relapses during the therapy cycle compared to one or more relapses. Next to that, just 567 out of 3007 patients have a *CDP* value of 1.

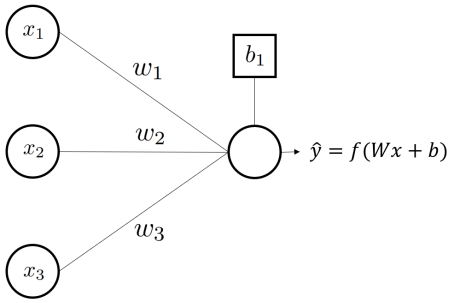
4 Methodology

This section describes the methodology employed to model the two clinical markers discussed in section 2. First, the intuition and training of a regular NN is addressed in section 4.1. Secondly, the theory regarding BNNs is introduced in section 4.2. Afterwards, we discuss the architecture of the BNNs in section 4.3. This includes modeling choices such as the assumed distributions, but as well incorporates the structure of the separate and combined BNNs, including the input and output variables that are used to train the BNNs and are previously described in section 3. In addition, we describe the validation methods that are applied to compare the performance of the models in section 4.4, including the use of several benchmark models. Finally, we discuss how the BNN can be adopted to express uncertainty in the recommendation of different therapy switches for MS patients in section 4.5.

4.1 A regular Neural Network

In this section we discuss the notation and methods that are used to train a NN. Although the exact notation in this section differs, we build on the examples given in Hastie et al. (2009) and Nielsen (2015). A simple form of a NN is shown in figure 2. This network consists of an input layer with three neurons and an output layer with one neuron. Next to that, there is a bias neuron connected to the output neuron. The aim of the NN is to retrieve the relationship between the input neurons and the output neuron. Subsequently, this relationship can be used to estimate the value of y for new data. The three weights in figure 2, that is, w_1, w_2 and w_3, scale the values of the input neurons x_1, x_2 and x_3, after which they are added with the bias neuron b_1 to retrieve an estimate of y. In other words, $\hat{y} = Wx + b$, where $W = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$, $x' = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$ and $b = b_1$. This type of NN implies that the relation between x and y is linear. However, it is possible to assume other types of relations between the weighted input neurons and output neuron. In this case, the pre-activation value $Wx + b$ is transformed by the function $f()$. The function $f()$ is called the activation function and can be set to several functions. It transforms the pre-activation value into a desired range. For example, if the output neuron concerns a chance variable, a logistic function can be used. This function transforms the pre-activation value into the range from 0 to 1. After the pre-activation value is transformed, we retrieve the post-activation value, which equals $f(Wx + b)$. Given an estimated set of weights, $\hat{y}$ can be estimated. In order to quantify the accuracy of $\hat{y}$, a cost function is needed to compute the loss of the network. There exist many different cost functions and the most suitable choice depends on the type of output neuron. For instance, a commonly used cost function for continuous values is the sum of squared error (SSE). On the other hand, the cross-entropy cost function is often applied to binary classification problems (Hastie et al., 2009).

Figure 2: A NN with 3 input neurons, 1 output neuron and a bias neuron.



The loss that is computed with the cost function is used to determine the optimal set of weights, given the data. The NN algorithm is an iterative algorithm that starts with an initial set of weights and updates these weights in each epoch, that is, in each iteration. To find the best possible update for each weight, the optimization of the loss with respect to each weight is required. These partial derivatives show how sensitive the cost function is for changes in the weights. In other words, it shows how the accuracy of $\hat{y}$ can be improved by changing the weights. The optimization can be performed with gradient descent (GD). GD updates the weight matrix W, for each epoch k in the following manner:

$$W_{k+1} = W_k - \lambda \frac{\delta C}{\delta W_k} \tag{1}$$

where C is the loss of the network and λ is the learning rate. The parameter λ is a hyperparameter. Hyperparameters are parameters that are fixed to a particular value during the training of the network yet can take many different values. Hence, the aim is not to estimate the hyperparameters during the training of the NN, but to choose the best possible value that supports the training of the NN. Once the value of the learning rate is fixed, GD can be used to update the weight matrix W in each epoch. The number of epochs can be chosen before the start of the algorithm. In addition, it is possible to let the number of epochs depend on the training of the network. For example, a commonly used method is early stopping, which evaluates the performance of the trained NN on a validation data set in each epoch. When the performance does not improve for a couple of epochs, the training is stopped (Nielsen, 2015).

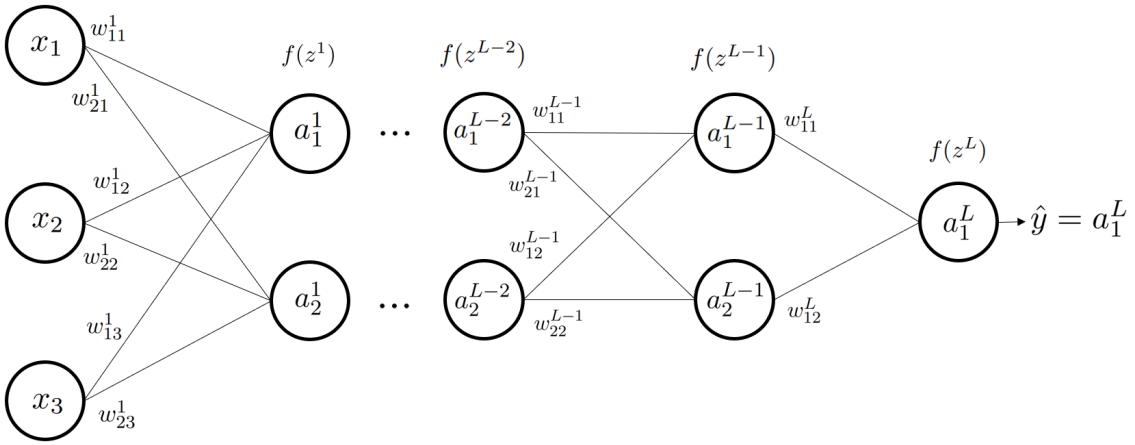
GD can be considerably slow for large data sets, as the algorithm computes the gradient of the cost functions for all observations. For this reason, machine learning techniques often apply stochastic gradient descent (SGD) instead (Nielsen, 2015). In each epoch, this algorithm randomly selects an observation $i \in (1, 2, ..., N)$ of the training data. The weights are then updated by only considering the gradient of the loss for observation i, instead of for the whole data set. Batch gradient descent (BGD) is another variant of GD, which uses a small batch of observations. The algorithm thus compromises between the full use of information in GD and the computational speed of SGD. BGD results in the following update,

$$W_{k+1} = W_k - \lambda \frac{\delta C_{s \in B}}{\delta W_k} \tag{2}$$

where the observations in the batch B are randomly drawn from the training data. This method is also called mini-batch gradient descent. The optimization and updates of the bias neuron b are performed in a similar way as the weights. Choosing a proper learning rate in the GD or BGD algorithm can be complicated, while it is of influence for the training of the network. If a learning rate is too small, it might cause the training of the NN to get stuck in local minima. On the other hand, a large learning rate can cause the training to pass over important minima. To solve this issue, there exist multiple algorithms that use an adaptive learning rate during the training. Adaptive Moment Estimation (Adam) is widely used in the training of NNs, as it is shown that Adam performs well in practice (Kingma & Ba, 2014). The exact algorithm of Adam is given in Appendix B.

The NN in figure 2 only consists of an input and output layer. Nonetheless, part of the strength and flexibility of NNs arises in the network of hidden layers (Hastie et al., 2009). These hidden layers can be added between the input and output layer. Leaving behind the simple NN of figure 2, we examine a NN with L layers. Each layer $l \in (1, 2, ..., L)$ has n^l neurons. Figure 3 shows a NN with L layers, that is, $L-1$ hidden layers and 1 output layer. Note that the input neurons are not counted as a layer. In this NN all neurons in two consecutive layers are connected. As for the output layer, it is possible to use an activation function for the neurons in the hidden layers. These functions are allowed to differ, yet they can as well be the same for all layers. Moreover, bias neurons can be added to all neurons in the hidden layers. Assume for simplicity that the activation

Figure 3: Neural Network with L layers.



functions of all layers are similar and that there are no bias neurons. Let z^l and a^l be the vectors of pre-activation and post-activation values for layer l. The NN can then be presented in matrix form. For example, the vectors for layer $L-1$ are as follows:

$$z^{L-1} = \begin{bmatrix} z_1^{L-1} \\ z_2^{L-1} \end{bmatrix} = \begin{bmatrix} w_{11}^{L-1} & w_{12}^{L-1} \\ w_{21}^{L-1} & w_{22}^{L-1} \end{bmatrix} \begin{bmatrix} a_1^{L-2} \\ a_2^{L-2} \end{bmatrix} = W^{L-1} a^{L-2} \tag{3}$$

$$a^{L-1} = \begin{bmatrix} a_1^{L-1} \\ a_2^{L-1} \end{bmatrix} = f(z^{L-1}) \tag{4}$$

The activation function $f()$ in equation (4) is applied element wise to the values in the vector z^{L-1}. As can be seen, the length of the vectors z^{L-1} and a^{L-1} correspond to the number of neurons in layer $L-1$, that is, n^{L-1}. The post-activation value of the output neuron in the last layer is equal to the prediction of y. Hence, we can write $\hat{y}$ as follows,

$$\begin{aligned} \hat{y} &= a^L \\ &= f(z^L) \\ &= f(W^L f(W^{L-1} f(...f(W^1 x))) \end{aligned} \tag{5}$$

As can be seen in equation (5), an essential difference compared to the former NN in figure 2 is the nested structure of the prediction of y. This implies that the complexity of the loss increases rapidly with the number of hidden layers, as $\hat{y}$ appears in this equation as well. For example, if we define the cost function as $h()$, the loss of the NN in figure 3 is as follows:

$$\begin{aligned} C &= h(y, \hat{y}) \\ &= h\left(y, f(W^L f(W^{L-1} f(...f(W^1 x)))\right) \end{aligned} \tag{6}$$

Recall that in order to update the weights in the GD algorithm, the partial derivatives of the loss with respect to all weights are required. These partial derivatives can be computed with the chain rule. For instance, the gradient with respect to the weight matrix in layer $L-1$ equals

$$\frac{\delta C}{\delta W^{L-1}} = \frac{\delta C}{\delta a^L} \frac{\delta a^L}{\delta z^L} \frac{\delta z^L}{\delta a^{L-1}} \frac{\delta a^{L-1}}{\delta z^{L-1}} \frac{\delta z^{L-1}}{\delta W^{L-1}} \tag{7}$$

In a similar fashion, the gradients with respect to the weight matrices of all layers can be retrieved. However, equation (6) and (7) clearly illustrate that this becomes more computationally burdensome when the depth of the NN increases. Since a NN can consist of millions of parameters for which the partial derivatives are needed, it is important to derive the gradient in an efficient manner. This is achieved by using the backpropagation algorithm in combination with an efficient GD algorithm, such as SGD or BGD (Nielsen, 2015). Backpropagation exploits the resemblance between the gradients of two consecutive layers. Using the chain rule, the gradients with respect to the weight matrix in layer l can be written as,

$$\frac{\delta C}{\delta W^l} = d^l \frac{\delta z^l}{\delta W^l} \tag{8}$$

with

$$d^l = d^{l+1} \frac{\delta z^{l+1}}{\delta a^l} \frac{\delta a^l}{\delta z^l} \tag{9}$$

Backpropagation uses the recursive relation in equation (9) combined with equation (8) to retrieve all gradients. The algorithm starts at the last layer L, by setting $d^L = \frac{\delta C}{\delta a^L} \frac{\delta a^L}{\delta z^L}$ and works back

to the first layer. Because of the recursive structure, none of the duplicate multiplications in the gradients of the network are computed more than necessary. This makes the algorithm as efficient as possible. A mathematical derivation of the recursive relationship of equation (9) is shown in Appendix C.

4.2 Bayesian Neural Networks

The NNs shown in figure 2 and 3 in the previous section consist of point estimates of the weights. On the contrary, BNNs aim to retrieve the probability distributions of the weights. In this section, we first discuss the theory of Bayesian statistics that is central to the probability distributions in a BNN. Afterwards, we introduce the Bayes by Backprop algorithm developed by Blundell et al. (2015), which is used to efficiently approximate the probability distributions of the weights.

4.2.1 Bayesian statistics

In Bayesian statistics the model parameter is assumed to be an unknown value which can be described with a probability distribution (Greenberg, 2012). The model parameter, which we define as β, can take different forms. For example, consider a coin toss in which throwing a head versus tails corresponds to $y = 1$ and $y = 0$, respectively. If we describe the probability that the toss results in head as $P(y = 1) = \beta$, the parameter β is a model parameter. As explained by Greenberg (2012), Bayesian statistics aims to find the probability distribution that describes the random variable β, conditional on the data. This distribution is called the posterior distribution and is central to Bayesian inference. Let us define the data as D. In the example of the coin toss, D incorporates the values of all coin tosses, that is, of all y. Then, using Bayes theorem, we can write the posterior distribution $p(\beta|D)$ as

$$p(\beta|D) = \frac{p(D|\beta)p(\beta)}{p(D)}$$

Since $p(D)$ is independent of β, the posterior distribution can be written as,

$$p(\beta|D) \propto p(D|\beta)p(\beta) \tag{10}$$

The first component of equation (10), $p(D|\beta)$, is the likelihood function. The likelihood function describes the probability of observing data D, given the parameter β. It follows that in this function, the data D is known. Moreover, the second component $p(\beta)$ is the prior distribution of β. This distribution contains the initial beliefs we have about β, before the data D is known. As Greenberg (2012) explains, these beliefs can for example be based on theoretical knowledge or empirical results. Hence, both the information about β in the data as the information we have before seeing the data is combined in the posterior distribution.

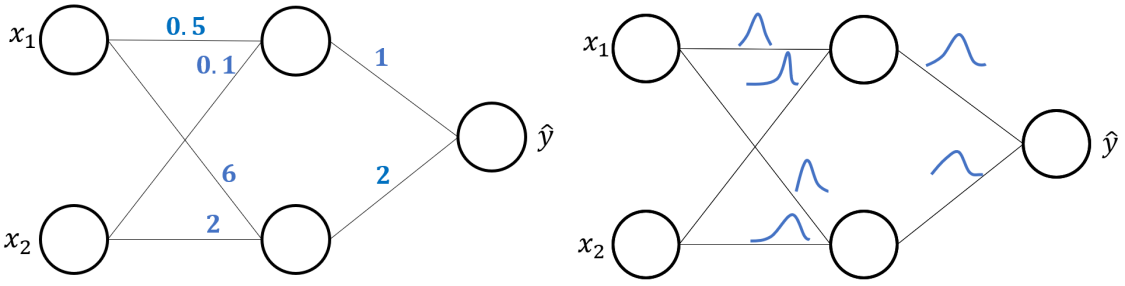
The posterior distribution takes a central role in Bayesian inference, including the prediction of unseen values of y. If D includes the observed data for observation $i = 1, ..., n$, the prediction of

y^{n+1} can be made by computing

$$p(y^{n+1}|D) = \int p(y^{n+1}|\beta)p(\beta|D)d\beta \tag{11}$$

4.2.2 Bayes by Backprop

Figure 4 visualizes the structural difference between NNs and BNNs. While regular NNs describe the weights of the network with point estimates, BNNs aim to find a probability distribution for the weights. Note that figure 4 does not include any bias neurons for simplicity. Nonetheless, the bias neurons in a BNNs are as well described with a probability distribution. The distributions in the BNN are the distributions of the weights, given the data. In other words, BNNs incorporate the theory of Bayesian statistics by estimating a posterior distribution for each weight. Hence, the posterior distribution in a BNN is defined as $p(w|D)$, in which w includes all weights and bias neurons of the network. The model parameter w can thus be seen as β in equation (10). The adoption of Bayesian statistics in BNNs results in two complications. First of all, in practice it is often not possible to find an analytical solution for the posterior distribution. Secondly, once we have an estimate of the posterior distribution, it is often infeasible to integrate over the posterior distribution (as in equation (11)) and predict the values of unseen data (Blundell et al., 2015).



(a) Point estimates in a NN

(b) Posterior distributions in a BNN

Figure 4: Visual representation of the difference between a regular NN and BNN for a simple network structure with two input neurons, one hidden layer and one output neuron. The neurons of a NN are connected with point estimates of the weights, while the probability distributions of the weights are used in the BNN.

To solve these issues, Blundell et al. (2015) propose the Bayes by Backprop algorithm that uses unbiased and efficient approximations to learn the posterior distributions of the weights in the BNN. The algorithm builds on the research of Graves (2011) and Hinton & Van Camp (1993), which utilizes variational inference to estimate the posterior distributions of the weights. As their research explains, variational inference employs an approximate distribution to replace the real posterior distribution. This approximation is called the variational distribution. The advantage of the variational distribution against the real distribution, which are both unknown, is that it is possible to use numerical methods to solve the optimization problem. This is needed as it is

often infeasible to find an analytical solution to the posterior distribution. Therefore, the posterior distribution $p(w|D)$ is approximated with the variational distribution $q(w|\theta)$. The parameter θ contains the distribution parameters of the weights and bias neurons, that is, w. Hence, θ depends on the assumed variational distribution. For example, if we assume that the variational distribution can be described with a Gaussian distribution, θ includes the distribution parameters μ and σ. To evaluate the distance between the variational distribution and the unknown posterior distribution, the Kullback-Leibler (KL) divergence is used. Variational inference aims to determine the optimal value of θ, which minimizes the KL divergence between the two distributions. Given two distributions q(x) and p(x), the KL divergence is defined by,

$$KL(q(x)||p(x)) = \int q(x) log\left(\frac{q(x)}{p(x)}\right) dx \tag{12}$$

Subsequently, the KL divergence between the variational distribution and the posterior distribution can be written in the following manner,

$$\begin{aligned} KL(q(w|\theta)||p(w|D)) &= \int q(w|\theta) log\left(\frac{q(w|\theta)}{p(w|D)}\right) dw \\ &= \mathbb{E}_{q(w|\theta)} log\left(\frac{q(w|\theta)}{p(w|D)}\right) \\ &= \mathbb{E}_{q(w|\theta)} log\left(\frac{q(w|\theta)}{p(D|w)p(w)} p(D)\right) \\ &= \mathbb{E}_{q(w|\theta)}\left[log\left(q(w|\theta)\right) - log\left(p(D|w)\right) - log\left(p(w)\right)\right] + log\left(p(D)\right) \end{aligned} \tag{13}$$

Minimizing equation (13) with respect to θ results in the estimate of the distribution parameter θ for the variational distribution. Since the last term, $log\left(p(D)\right)$, does not depend on θ, it can be ignored during the minimization of the KL divergence with respect to θ. Finding the optimal set of parameters thus reduces to

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w|\theta)}\left[log\left(q(w|\theta)\right) - log\left(p(w)\right) - log\left(p(D|w)\right)\right] \\ &= \underset{\theta}{\operatorname{argmin}}\, F(D, \theta) \end{aligned} \tag{14}$$

Equation (14) consist of three components. The first component corresponds to the variational distribution and the second to the prior distribution of w. Furthermore, the last component of the equation can be interpreted as the error loss function of the network (Graves, 2011). In other words, it can be seen as the likelihood function in a regular NN, which equals the cost function as discussed in section 4.1. An appropriate choice of this function depends on the data.

The Bayes by Backprop algorithm of Blundell et al. (2015) employs the theory of variational inference and aims to minimize equation (14). However, as the authors note, it is often infeasible to minimize this equation analytically. For this reason, Blundell et al. (2015) propose an unbiased approximation for the minimization in equation (14), which is applied in each epoch of the training of the BNN. Firstly, $F(D, \theta)$ is estimated by drawing a sample for each weight and bias neuron

from the current estimate of the variational distribution $q(w|\theta)$. These samples of w are used to approximate $F(D, \theta)$ with the following equation,

$$F(D, \theta) \approx log\left(q(w|\theta)\right) - log\left(p(w)\right) - log\left(p(D|w)\right) \tag{15}$$

Subsequently, the approximation of $F(D, \theta)$ is minimized with respect to the parameters in θ, after which they are updated by means of GD. The approximation of equation (15) can be considered as the loss of the BNN, which we need to optimize to retrieve the updates of the parameters. The gradient of the loss can be retrieved efficiently by means of the backpropagation algorithm described in section 4.1. The updated parameter θ is then used as distribution parameter in the updated estimate of the variational distribution, which is used in the next epoch to draw samples for the weights.

We continue with an example of the Bayes by Backprop algorithm, which uses the theory described in this section. Although the algorithm applies for non-Gaussian distributions as well, we continue with an example of the Bayes by Backprop algorithm where $p(w)$ and $q(w|\theta)$, the prior and variational distribution, are Gaussian distributions. Let the distribution of the prior be a standard normal distribution. Next, assume that the weights and bias neurons are independent and that their probability distribution is described with a different mean and variance. In this case, the two distributions are given by,

$$p(w) = \mathcal{N}(0, I) \tag{16}$$

$$q(w|\theta) = \prod_{j=1}^{W} \mathcal{N}(w_j|\mu_j, \sigma_j^2) \tag{17}$$

As can be seen in equation (17), the distribution parameter in this example is equal to $\theta = [\mu, \sigma]$, where μ and σ are vectors of length W. Note that W and w are two different parameters. W is a scalar and is equal to the total number of weights and bias neurons in the network. On the contrary, w is the model parameter containing all weights and bias neurons and is a W dimensional vector. The logarithm of the two distributions shown above corresponds to the first two terms in equation (15). The likelihood function or loss, that is, the last term in this equation, can be set to an appropriate function, which depends on the type of data in the application.

To approximate $F(D, \theta)$, we draw values for w from the variational distribution. Given that $q(w|\theta)$ is a Gaussian distribution, the sampled weights can be drawn from a standard normal distribution after which they are shifted and scaled with μ and σ, respectively. Sampling the weights in this manner might seem unnecessary, as we can as well directly sample from the Gaussian distribution. However, this so called reparameterisation trick is crucial in the proposed algorithm of Blundell et al. (2015) to get an unbiased estimate of the gradient of equation (15). In addition, to make sure that σ is always nonnegative, Blundell et al. (2015) use the softplus function to transform the parameter ρ to σ with $\sigma = log(1 + \exp(\rho))$. Therefore, in this example where we assume a Gaussian distribution for the variational distribution, the parameter that describes the distribution

of the weights is $\theta = [\mu, \rho]'$. To summarize, the Bayes by Backprop algorithm iterates through the following steps for the Gaussian example,

1. Draw $\epsilon \sim \mathcal{N}(0, I)$, where ϵ is a W-dimensional vector.
2. Create the weights by $w = \mu_k + log(1+\exp(\rho_k)) \odot \epsilon$, where $\odot$ is the element wise multiplication operator. μ_k and ρ_k are the estimated distribution parameters of the previous epoch and are both W-dimensional vectors.
3. Approximate $F(D, \theta)$ with equation (15) and optimize with respect to θ_k, that is, with respect to μ_k and ρ_k.
4. Retrieve the partial gradients $\frac{\delta F(D,\theta_k)}{\delta \mu_k}$ and $\frac{\delta F(D,\theta_k)}{\delta \rho_k}$, which can be used to update the distribution parameters to μ_{k+1} and ρ_{k+1}.

Blundell et al. (2015) use GD to update the distribution parameters in step 4. Nonetheless, as it is possible to apply other methods to update the parameters, this research will use the Adam algorithm in step 4, which is discussed in more detail in Appendix B.

After the BNN is trained, the estimate of the distribution parameter θ can be utilized to form predictions for unseen data. In the Gaussian example given above, the vector θ contains the estimates of μ_j and ρ_j for each $j \in W$. As can be seen in equation (17), the variational distribution consists of the probability distributions for each weight and bias neuron. We can thus sample values for the weights and bias neurons from these estimated distributions. This is accomplished by sampling a value for each weight and bias neuron $j \in W$, from the distribution $\mathcal{N}(\mu_j, \sigma_j^2)$, where $\sigma_j^2 = \big(log(1 + \exp(\rho_j))\big)^2$. In essence, this is similar to the creation of the weights in step 2 in the Bayes by Backprop algorithm, although we use the final estimates of μ and ρ instead of the estimates μ_k and ρ_k of epoch k. Afterwards, the sampled values can be regarded as point estimates in a regular NN. With the use of the activation function, it is then possible to use the regular theory from NNs to form predictions for y with equation (5) in section 4.1. This procedure results in one single prediction for each observation. Nonetheless, when we repeat the above procedure of sampling values for the weights and bias neurons, we can create an infinite number of NNs with the use of only one BNN. Each NN in the ensemble consists of different values for the weights and bias neurons and will result in different predictions of y. In this manner, the uncertainty in the distributions of the weights can demonstrate the uncertainty about which model in the ensemble is able to describe the values of y. This can expose uncertainty about a specific observation (Blundell et al., 2015). For example, if there is much uncertainty in the weights, it is possible that the ensemble of models results in many different predictions for y.

4.3 Architecture of the Bayesian Neural Networks

In this section we discuss the architecture of the BNNs implemented in this research. First of all, in section 4.3.1 we review the choice of distributions that are required for the Bayes by Backprop

algorithm discussed in section 4.2.2. Next to that, we discuss the manner in which the algorithm is implemented in section 4.3.2. Finally, we address the network structure of the two separate BNNs and the combined BNN, regarding the choice of the likelihood function and the activation functions of the models in section 4.3.3 and 4.3.4, respectively.

4.3.1 Choice of distributions

The Bayes by Backprop algorithm, which is introduced in section 4.2.2, aims to estimate the following function,

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathbb{E}_{q(w|\theta)} \left[log\left(q(w|\theta)\right) - log\left(p(w)\right) - log\left(p(D|w)\right) \right] \tag{18}$$

The functional choices for the three components of equation (18) can be considered separately. Firstly, during our analysis, a Gaussian distribution is used to estimate the variational distribution. The variational distribution is shown in equation (19). As the weights and bias neurons are assumed to be independent, the variational distribution is equal to the product of the Gaussian distributions for each weight and bias neuron $j \in W$. Secondly, the prior distribution is set to a Gaussian distribution with zero mean, as shown in equation (20). Different values of prior variances are employed during this research. From now one, we refer to this prior as the Gaussian prior. Next to that, we apply a mixture of two Gaussian distributions with zero mean as prior distribution (equation (21)), as proposed by Blundell et al. (2015). In this manner, the initial assumptions on the weights are more flexible, as the weights can originate from two different Gaussian distributions. We refer to this prior as the Mixture Gaussian prior. Finally, the last part of equation (18) is equal to the likelihood function of the model and can be seen as the loss C in a regular NN. Since an appropriate choice of the likelihood function depends on the characteristics of the data, the implementations in this research are further discussed in sections 4.3.3 and 4.3.4. To conclude, the following distributions are applied during the estimation of the BNNs, in which either equation (20) or (21) is used,

$$q(w|\theta) = \prod_{j=1}^{W} \mathcal{N}(w_j|\mu_j, \sigma_j^2) \tag{19}$$

$$p(w) = \prod_{j=1}^{W} \mathcal{N}(w_j|0, \sigma_{prior}^2) \tag{20}$$

$$p(w) = \prod_{j=1}^{W} \left(\pi \mathcal{N}(w_j|0, \sigma_{1,prior}^2) + (1-\pi)\mathcal{N}(w_j|0, \sigma_{2,prior}^2) \right) \tag{21}$$

$$-log\left(p(D|w)\right) = C \tag{22}$$

Note that the likelihood function in equation (22), depends on the actual data D. For example, if we use the SSE function, this will imply that $C = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$. On the contrary, the variational

distribution and prior distribution do not depend on the data and neither does their contribution to the approximation of the loss of the BNN in equation (15). This implies that their values in the approximation do not scale with the number of observations, while the value of the likelihood function does. This is of no importance when all observations are used in each epoch of the training. However, when BGD is used to optimize the weights, it is needed to scale the different parts in the approximation of the loss appropriately. This scaling is necessary to have a proper weighting of the variational and prior distribution against the likelihood function. For instance, given a number of weights, the contribution of the variational and prior distribution to the loss will be similar for 100 versus 100000 observations, contrary to the contribution of the likelihood function. As BGD implies that a fraction of the data is used, it is needed to scale the approximation of the loss proportionally. In conclusion, if BGD is used, step 3 of the Bayes by Backprop algorithm in section 4.2.2 is estimated with,

$$F(D,\theta) \approx \frac{1}{b}\big[log\left(q(w|\theta)\right) - log\left(p(w)\right)\big] + C \tag{23}$$

in which b represents the number of batches employed in BGD (Blundell et al., 2015).

4.3.2 Implementation of Bayes by Backprop

The training of the BNNs is accomplished in python, with the use of the libraries `tensorflow` and `tensorflow probability`. These libraries are able to estimate many types of NNs, among which BNNs. Although the most elemental functions of `tensorflow` cannot be used to train a BNN, there exist functions in the libraries that allow us to build different parts of the training process. Combined, they are able to construct a general BNN. However, not all distributions discussed in the previous section can be implemented with the existing functions of the libraries. Currently, `tensorflow probability` does not provide the application of a Mixture Gaussian prior, which is shown in equation (21). Nonetheless, it is possible to add manual written code in which we can define the 'layer' that is used for the weights and bias neurons. In this manual written layer, we specify the variational distribution, the prior distribution, the trainable parameters and their contribution to the approximated loss of equation (23). This layer is utilized in each epoch of the training. The code iterates through steps 1 till 3 of the Bayes by Backprop algorithm, specified in section 4.2.2. The approximation of the loss, $F(D,\theta)$, in step 3 of the algorithm is estimated with the use of the variational and prior distributions of equation (19) and (21). Hence, in this step, the Mixture Gaussian distribution is employed as the prior distribution of w. Before these three steps are employed in each epoch of the training of the BNN, the distribution parameters are initialized to specific values. If $j = 1, ..., W$ refers to all weights and bias neurons in the BNN, the algorithm of section 4.2.2 can be extended by the following initialization:

0. Initialize the trainable parameters of the variational distribution, $\theta_j = [\mu_j, \rho_j]'$. For each j,

   (a) initialize ρ_j to the constant value ρ

(b) initialize μ_j by sampling from the distribution $\mathcal{N}\Big(0, (\sigma^2_{1,prior} * \pi + \sigma^2_{2,prior} * (1-\pi))\Big)$, where the parameters $\sigma_{1,prior}$, $\sigma_{2,prior}$ and π are given in equation (21)

Combine the initialization of each $[\mu_j, \rho_j]$ to form the W-dimensional parameter vectors μ_0 and ρ_0

After the loss of the BNN (see equation is approximated with the use of the Mixture Gaussian prior in the manual written layer, the approximation is passed to the original code which uses the original functionalities of `tensorflow` and updates the trainable parameter $\theta = [\mu, \rho]$ with the Adam algorithm. By manually adding this layer in the code, we are able to include the Mixture Gaussian prior in the estimation of the BNN. Although there exists a layer in `tensorflow probability` in which the Gaussian prior of equation (20) is implemented, we as well construct a manual written layer for this prior. This layer allows for a more flexible initialization of the weights, which is more consistent with the layer for the Mixture Gaussian prior. This is preferable as the performance of the two different priors will be compared for varying values of initialization. In the layer with the Gaussian prior, the distribution of step 0.(b) is replaced by $\mathcal{N}(0, \sigma^2_{prior})$.

In addition to the manually written layers, an adjustment is made to the existing likelihood function of `tensorflow` which is employed in equation (22). The likelihood functions return an average loss over the number of observations. For example, the cross-entropy function of `tensorflow` calculates the average cross-entropy loss, instead of the sum over all observations. In a regular NN this will not influence the gradient of the loss, as the loss is simply scaled with a constant factor. However, in a BNN the loss does not only consists of the likelihood function. The first two terms of equation (23), the variational and prior distribution of the network, are of influence as well. As these functions do not scale with the number of observations, the use of the average likelihood functions of `tensorflow` would imply that the three components of the loss of the BNN are not weighted correctly. For that reason, the existing likelihood function of `tensorflow` is multiplied with the number of observations to regain the sum of the likelihood function instead of the average.

4.3.3 Two separate networks

In addition to choosing the prior and variational distribution of the BNN in section 4.3.1, several other modelling decisions need to be made. The BNNs in this analysis have multiple hyperparameters, which can take many different values and can influence the training of the BNN substantially. Some of the hyperparameters in this research are related to the network structure of the model, such as the number of hidden layers and the number of neurons per layer. The learning rate and batch size, which influence the updates of the parameters in each epoch, can differ as well. Next to that, there are several hyperparameters related to the prior and variational distribution. The hyperparameters are chosen by means of a grid search combined with k-fold cross validation. The hyperparameters and precise settings of the grid search are discussed in section 4.4.2.

Apart from the hyperparameters, other modelling decisions need to be made regarding the architecture of the models. We consider two separate BNNs to predict the two clinical markers in the data set of MS patients. The first model predicts if the level of disability of a MS patient increases or not, that is, if CDP is 1 or 0. The second model predicts the expected number of relapses. Since the number of relapses is a count variable and CDP is a binary variable, the two models will have different types of output neurons. However, the input neurons will be similar across the two models. The variables that are used for the input layer are discussed in section 3 and can be seen in table 1. Furthermore, both models will use the rectified linear unit (ReLu) activation function for the hidden layers, shown in equation (24). In this equation z refers to the pre-activation value, as introduced in section 4.1. The ReLu function is shown to perform well in empirical applications with deep NNs (Zeiler et al., 2013).

$$f(z)^{ReLu} = max(z, 0) \tag{24}$$

The activation function for the output layer will differ between the models. The activation function for the CDP model will be set to the softmax function. This function returns a value between 0 and 1 for each output neuron and forces the sum of these outputs to equal 1. In this manner it properly represents the chance an individual belongs to either of the two groups. Using this structure, the output layer of the CDP model has two output neurons, one for $CDP = 0$ and one for $CDP = 1$. The softmax output for neuron $i \in (1, 2)$ in the output layer L is equal to

$$f(z)_i^{CDP} = \frac{e^{z_i}}{\sum_{j=1}^{2} e^{z_j}} \tag{25}$$

Moreover, to estimate the number of relapses, the ReLu function is used as activation function in the output layer. This function returns a nonnegative value, which resembles the range of the count variable. The relapse model has one output neuron, for which the ReLu function returns a continuous prediction of y. To conclude, in the CDP model the output layer consists of two neurons which describe the chance of belonging to either of the two groups, while the relapse model has one output neuron which describes the expected number of relapses.

Since the two BNNs predict a different type of variable, the performance of the models will be evaluated with different likelihood functions. As the CDP is a binary variable, the cross-entropy function is used for the CDP model. Furthermore, since the output of the relapse model is continuous, the SSE is used to evaluate the distance to the actual number of relapses.

$$C^{CDP} = -\sum_{i=1}^{N} (y_{i,1} log(\hat{y_{i,1}}) + y_{i,2} log(\hat{y_{i,2}})) \tag{26}$$

$$C^{relapse} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{27}$$

In the above equations, $\hat{y}$ and y refer to the predicted output of the last layer of the BNN and the observed value, respectively. Since the CDP model is a binary classification problem with two

output neurons, the cross-entropy function in equation (26) consists of two parts. When CDP equals 0, $y_{i,1} = 1$ and $y_{i,2} = 0$, while $y_{i,1} = 0$ and $y_{i,2} = 1$ if CDP equals 1. Hereafter, when we refer to the single CDP model or the single relapse model, we refer to the BNN structures addressed in this section.

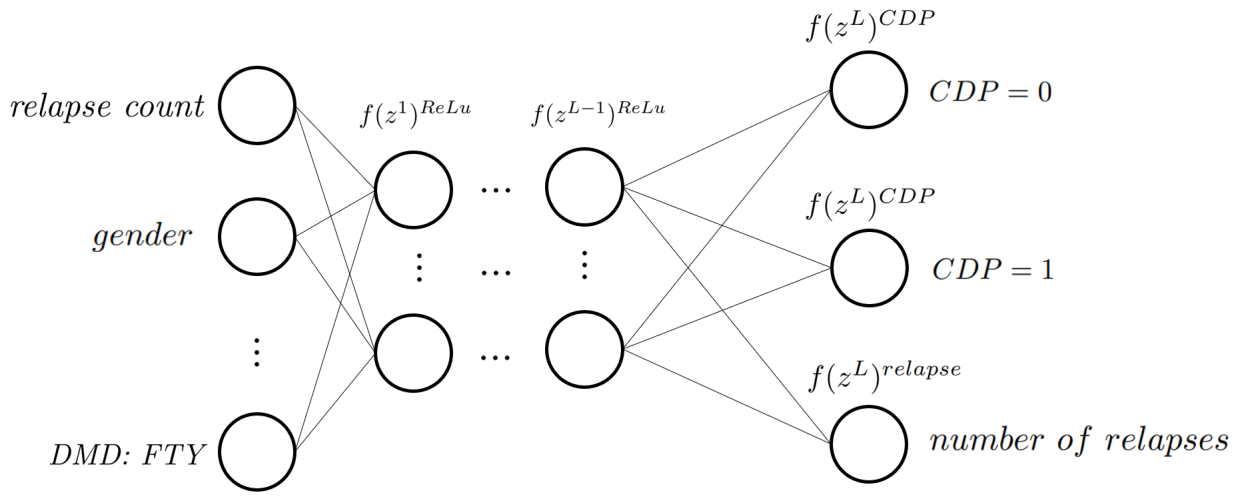
4.3.4 One combined network

In addition to the two separate models, we train a combined BNN which models the two clinical markers simultaneously. The combined BNN has the same set of input neurons as the separate models described in section 4.3.3. However, this model has three output neurons, that is, two for CDP and one for the number of relapses. As before, the ReLu activation function is used in the hidden layers and the output neurons for the CDP and the number of relapses are activated with the softmax and ReLu function, respectively. We use hard parameter passing in the multi-task learning network, which means that the output neurons share the same hidden layers (Caruana, 1997). The likelihood functions for CDP and the number of relapses are again set to the cross-entropy function and the SSE. The total likelihood function for the combined BNN is retrieved by adding the two different likelihood functions. This results in the following likelihood function, in which y_1, y_2 and y_3 correspond to the two variables for the CDP and the number of relapses, respectively.

$$C^{combined} = -\sum_{i=1}^{N} (y_{i,1} log(\hat{y_{i,1}}) + y_{i,2} log(\hat{y_{i,2}})) + \sum_{i=1}^{N} (y_{i,3} - \hat{y_{i,3}})^2 \tag{28}$$

The combined BNN structure is visualized in figure 5. Note that different activation functions for the two output variables are employed in the last layer of the network. As before, the hyperparameters of the model are chosen through a grid search and k-fold cross validation. From now one, we refer to this BNN structure as the combined model.

Figure 5: Visual representation of the structure and differing activation functions of the combined BNN.



4.4 Validation

In this section we discuss the methods that are needed to assess the performance of the BNNs. First of all, the performance measures that are used to evaluate the models are addressed in section 4.4.1. Secondly, we discuss how we examine the effects of the hyperparameters in the models, which allows us to select the best performing sets of hyperparameters in section 4.4.2. In section 4.4.3 we explain how these sets of hyperparameters are used to train the final models. Finally, we introduce the benchmark models in section 4.4.4, which are used to compare the performance of the BNN.

4.4.1 Performance measures

We assess the predictive performance of the models with different measures. The MSE is used to evaluate the predictions of the single relapse model, as the predicted output of this model is continuous. Moreover, the accuracy is commonly used to measure the performance of binary classification models. The accuracy divides the number of correct classifications by the total number of classifications. Thus, the measure reveals the percentage of correct classifications. However, the accuracy might not be an appropriate measure for the single CDP model. This finding is driven by two observations. First of all, the data set is imbalanced as the patients with a CDP value of 1 represent only 18.86 percent of the data set, which is shown in figure 1 in section 3.2. Next to that, from a medical point of view it is more crucial to correctly predict a CDP of 1, compared to correctly predicting a CDP of 0. In this situation, the accuracy can be an uninformative measure, as it does not distinct between the two different groups. For instance, an accuracy of 81.14 percent might seem high at first sight, yet it is possible that the model only predicts a CDP value of 0. This implies that the 18.86 percent of patients with a CDP of 1 are all classified incorrectly. For this reason, we use the area under the precision recall curve (AUPRC), which is a measure that stresses the importance of the correct classification of the less frequently observed group and is more informative for imbalanced data sets (Saito & Rehmsmeier, 2015). The AUPRC makes use of the confusion matrix, shown in figure 6. For a certain threshold, this matrix divides the precision

Figure 6: Confusion matrix

		Predicted: 1	Predicted: 0
Actual	1	True Positive (TP)	False Negative (FN)
	0	False Positive (FP)	True Negative (TN)

of the classification model into different areas of interest. For example, if the threshold is equal to 0.4, all predictions that are smaller than 0.4 are classified as 0, while the remaining predictions are classified as 1. After all predictions are classified, they can be divided into the four categories in the confusion matrix. The threshold of the model can be set to a value between 0 and 1, resulting in different confusion matrices. The AUPRC considers the precision and recall, shown in equation (29), of the model at many different thresholds between 0 and 1. Together, these values result in

a curve ranging from 0 till 1 on both the recall and precision axis.

$$recall = \frac{TP}{(TP + FN)} \qquad precision = \frac{TP}{(TP + FP)} \tag{29}$$

A threshold of 1 implies that all predictions are classified as 0. In this case, the recall is equal to zero. On the contrary, a threshold of 0 results in all predictions being classified as 1, which means that all predictions are spread over the true positives (TP) and false positives (FP). This results in a recall of 1 and a precision which equals the proportion of observations belonging to class 1 in the data set. The AUPRC summarizes this curve in one value by calculating the area under the curve. Note that the AUPRC does not include the True Negatives (TN), which is preferred in this application as the TP are of higher importance. In this research, the precision recall curve is estimated by using 200 different threshold values, evenly distributed between 0 and 1. Furthermore, a random classifier would achieve an AUPRC which equals the proportion of the smaller class (Saito & Rehmsmeier, 2015). Hence, in our analysis, the value of the AUPRC should be compared to the proportion of individuals with a CDP value of 1, which equals 18.86%.

Finally, to compare the performance of the two single models with the combined model, the performance measures for the combined model are similar to those of the single models. In other words, the predictions of the combined model for the variable CDP are evaluated with the AUPRC, while the predictions for the number of relapses are evaluated with the MSE.

4.4.2 Grid search

The performance of the BNN depends on the choice of hyperparameters. As discussed in section 4.1, hyperparameters are fixed during the training of the network and their value needs to be chosen with care. Since there are multiple hyperparameters in the proposed models of this research, which can all take values within a large range, there exist many different possible model specifications. Although exploring the endless possibilities of combinations can be tedious, it is important to consider many combinations of hyperparameters as we cannot argue with certainty how they will affect the performance of the model. For this reason, we compare the performance of the three BNNs in this research over a grid search of values for all hyperparameters. This as well allows us to examine the joint effect of several hyperparameters. Due to the relatively large computational time of the models and the large number of hyperparameters, the total computational time of the grid search increases quickly. In order to analyze a feasible grid search which considers enough different values per hyperparameter, we split the grid search into two parts. In the first grid search we examine the hyperparameters that are related to the basic structure and the training process of the network. Subsequently, the second grid search explores the effect of the hyperparameters that are related to the Bayesian nature of the network. Before we discuss the hyperparameters of the two grid searches in more detail, we explain how we evaluate the performance of each set of hyperparameter in the grid search in a stable manner that as well incorporates the uncertainty present in the BNN.

Evaluation of grid search

We evaluate the performance of the BNNs for each possible set of hyperparameters in the grid search as follows. First, the data is split into a training data set and testing data set. The training data is chosen by randomly selecting 80% of the data. To ensure that the test data is exclusively used to assess the performance of the final model, after all modeling decisions are made, we do not use the test data in the grid search. Instead, we use k-fold cross validation. For this, we further divide the training data into five folds, where each fold contains 20% of the original training data. Afterwards, we use 4 folds to train the BNN, after which the remaining fold can be used as validation data to evaluate the performance of the BNN. Hence, the validation data can be regarded as unseen data for the trained BNN. This procedure is repeated five times, where each fold operates as validation data once. Afterwards, the average performance over the five folds is computed. The exact split to create the five folds is kept equal for all models in the grid search. As discussed in section 4.2.2 the estimated variational distributions can be used to make predictions, by sampling values for the weights and bias neurons and using these values to construct a regular NN. This NN can then be employed to form predictions for the unseen validation data. However, using only one prediction does not give much information about the performance of the BNN, as the uncertainty in the weights can results in different predictions for one particular observation. For this reason, we form 100 different predictions after the BNN is trained. For each of these predictions, we evaluate the performance with the measures described in section 4.4.1. Afterwards, we compute the average performance of the 100 predictions and use this as the performance of the particular BNN in the k-fold cross validation.

To summarize, using 5-fold cross validation, we train five BNNs for each set of hyperparameters in the grid search. For each of these five BNNs, we create 100 different predictions for the observations in the validation data set. Afterwards, the average performance of the 100 predictions is computed. When this is done for each of the five BNNs, we compute the average performance over the five folds. Finally, this average performance is regarded as the performance belonging to the particular set of hyperparameters in the grid.

First grid search

The hyperparameters in the first grid search include the depth of the network, the number of neurons per hidden layer, the batch size and the learning rate. The values of the grid search for these hyperparameters are shown in table 3. As can be seen in this table, the hyperparameters that correspond to the basic structure of the network, that is, depth and number of neurons, are combined into one hyperparameter `network structure`. For instance, if a model has a `network structure` of $[2, 4]$, the model has 2 hidden layers. After the input layer, there are 2 neurons in the first hidden layer and 4 neurons in the second hidden layer, which finally connects to the last output layer. Moreover, a bias neuron is added for each neuron in the network, including the neurons in the output layer. Furthermore, the latter two hyperparameters `batch size` and `learning rate`

relate to the training process of the network. The hyperparameter `batch size` is equal to the size of B in equation (2), while the `learning rate` corresponds to λ in equation (32) of the Adam algorithm in Appendix B. In addition to testing several batch sizes for BGD, we apply regular GD by considering the full data set in each epoch.

Table 3: First grid search

`network structure`	{[2], [10], [25], [2,4], [4,2], [8,20], [20,8], [15,30], [30,15], [2,4,8], [8,4,2], [2,4,10,25], [25,10,4,2]}
`batch size`	{Full data set, 128, 64, 32}
`learning rate`	{0.4, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001}

This table shows the range of values that is used for the first set of hyperparameters. The values of the hyperparameters that are examined in the second grid search are set to `prior distribution` *= Gaussian,* `prior variance` *= 1 and* `variational variance rho` *= -3*

The first grid search is executed separately for all three models, that is, the single CDP model, the single relapse model and the combined model. Each model is trained for 200 epochs. For each model, we evaluate the performance of the models within the grid search and select the optimal set of hyperparameters. As the single CDP and single relapse model are two different models with different performance measures, it is possible that their optimal set of hyperparameters differs. Next to that, the best performing hyperparameters for the single models do not have to be similar to those of the combined model. For this reason, we may end up with four different optimal sets of hyperparameters after the first grid search, to which we refer as H^1_{CDP}, $H^1_{relapse}$, $H^1_{com:CDP}$ and $H^1_{com:rel}$. The former two hyperparameter sets are based on the results of the grid search for the single models. Moreover, the choice of set $H^1_{com:CDP}$ is based on the best performing combined model, regarding the predictions of the CDP. Hence, we examine the AUPRC for the CDP predictions of the combined model and disregard the predictions of the number of relapses. In addition, the set $H^1_{com:rel}$ is based on the MSE's of the predictions of the combined model regarding the number of relapses.

Second grid search

We continue the second grid search with the optimal set of hyperparameters for each model. For example, the hyperparameters `network structure`, `batch size` and `learning rate` are set to $H^1_{relapse}$ during the second grid search of the single relapse model. The second grid search explores the effect of the hyperparameters that are related to the Bayesian part of the loss of the BNN, that is, the variational and prior distribution.

As seen in table 4, we examine the effect of the Gaussian prior versus the Mixture Gaussian prior of equation (20) and (21) with the hyperparameter `prior distribution`. For both prior distributions we vary the `prior variance`. In the application of the Gaussian prior, this corresponds to σ^2_{prior} in equation (20). Moreover, the prior variance of the Mixture Gaussian prior relates to

$(\sigma^2_{1,prior}, \sigma^2_{2,prior})$ in equation (21). In addition, we vary the `mixture probability` π in equation (21). Finally, we consider different initialization for the mean and variance of the variational distribution in equation (19). Firstly, the initialization of μ_j depends on the prior distribution, as discussed in section 4.3.1. The mean is thus indirectly initialized to different values by varying the `prior variance`. Secondly, we vary the initialization of the variance σ_j^2 through ρ_j, that is, through the `variational variance rho`. The transformation of ρ_j is done with the softplus formula, $\sigma_j = log(1 + \exp(\rho_j))$. We can thus conclude that a value of $\rho_j = -5$ results in a variational variance of approximately zero. Note that the hyperparameters of the variational distribution are exclusively used to initialize the parameters of the distribution. Afterwards, the parameters are updated in each epoch. On the contrary, the hyperparameters of the prior distribution are constant through the whole training process of the network.

Based on the results of the second grid search, we can choose the final optimal set of hyperparameters for each model, that is, $H^2_{relapse}$, H^2_{CDP}, $H^2_{com:CDP}$ and $H^2_{com:rel}$. Similar as for the first grid search, these sets are chosen by selecting the best performing models within the grid search.

Table 4: Second grid search

`prior distribution`	{Gaussian, Mixture Gaussian}
Gaussian	
`prior variance`	{0.01, 0.25, 1, 3, 100 }
`variational variance rho`	{-5, -3, -1, 0.5, 2}
Mixture Gaussian	
`prior variance`	{(1,0.25), (3, 0.25), (1, 0.01), (3, 0.01), (1, 0.0001)}
`mixture probability`	{0.95, 0.8, 0.5, 0.2}
`variational variance rho`	{-5, -3, -1, 0.5, 2}

This table shows the range of values that is used for the second set of hyperparameters. The `network structure`, `batch size` and `learning rate` are set to the best performing models of the first grid search.

4.4.3 Final models

After we retrieve the optimal sets of hyperparameters, we can examine the final models. These models are trained by using the full training data set of 80% and the performance is evaluated on the test data set. For the final models, we investigate several matters. First of all, we assess the stability of the models by training the models 100 times. The 100 models employ the same set of hyperparameters yet differ in their initialization. As discussed in section 4.3.2, the parameters of the variational distribution are initialized by drawing from a previously specified distribution. Hence, by training the model 100 times, we can examine the stability of the model over different values of initialization. In every epoch, each of the 100 models forms a single prediction for the test data. Subsequently, we can use the performance measures described in section 4.4.1, to assess the performance of each model and compute the average performance over the 100 models. Afterwards, we visualize the variation of the performance of the 100 models by plotting the average performance

plus and minus both two and three standard deviations. For example, for the single CDP model we examine the average AUPRC, added and subtracted with two and three standard deviations. Afterwards, we can examine if the performance of the model is stable with regard to a different initialization, or if it results in varying performance.

After we plot the stability of the models, we train one BNN for each optimal set of hyperparameters. Similar as for the models in the grid search, we assess the performance of the BNN by examining 100 different predictions. This allows the uncertainty of the weights to affect the predictions of the BNN. We can then visualize the variation of the performance over the 100 different predictions in a similar fashion as for the variation over the 100 different values of initialization. The varying performance over the 100 different predictions can indicate the level of uncertainty in the BNN. The performance of the final models is computed as the average performance over the 100 predictions in the last epoch. As the combined model will be applied to predict both clinical markers, we assess the performance of the predictions for both variables in the combined model. We compare the performance of the single and combined model to examine if the theoretical advantages of multi-task learning improve the prediction of the two clinical markers. Finally, we compare the performance of each BNN with the benchmark models discussed in the succeeding section.

4.4.4 Benchmark models

In addition to comparing the performance of the single BNNs with the combined BNN, we compare the results with several benchmark models. First of all, we compare the BNNs with the results of the two GLMs of Stühler et al. (2020), estimated on the synthetic data. The authors examine three different models, which differ in their input variables. The two more elaborate models, the prognostic model and the predictive model, use the input variables shown in table 1 in section 3.1. The prognostic model exclusively uses these variables as input variables, while the predictive model uses some interaction effects as well. The results show that the difference in performance between the prognostic model and the predictive model is small, while the latter increases the model complexity. For this reason, we consider the prognostic model, estimated on the synthetic data, as benchmark in this research. Henceforward, we refer to the prognostic GLM, estimated on the synthetic data set as the synthetic GLMs. As discussed in section 3.2, we follow the modeling procedures from Stühler et al. (2020) and use the library `rstanarm` in R to estimate the synthetic GLMs.

Additionally, we compare the performance of the BNNs with two benchmark GLMs. As GLMs allow the variable of interest to originate from another family of distributions with the use of a link function (in addition to the Gaussian distribution), we need to specify these two components. We use a Poisson regression to compare the results with the predictions for the number of relapses of the BNNs. To estimate the Poisson, regression we use a Poisson distribution combined with a natural logarithm link function. Next to that, we estimate a logistic regression by setting the family distribution to a binomial distribution, combined with a logit link function. The logistic regression

is used for comparison with the predictions of the CDP. We employ the library `statsmodels.api` in python to estimate the two GLMs. To compare the performance of the models properly, we use the same split of training data and testing data for the estimation of all benchmark models. Finally, the performance measures as discussed in section 4.4.1 are obtained for all models.

4.5 A distribution of recommendations

The research of Stühler et al. (2020) applies the two GLMs to recommend which possible therapy switches are most suitable for a particular patient. The outcomes of the two models, for the CDP and the number of relapses, rank the possible therapy switches and are used in the discussion between the doctor and patient. As the choice of the recommended therapy switch can be based on both the CDP as the number of relapses, two possibly different recommendations can be given. The combined BNN in this research as well results in two predicted values for the two clinical markers. In other words, using the combined BNN to recommend therapy switches might still result in two different recommendations. However, the combined BNN has two features that are beneficial for this application. First of all, both clinical markers are predicted within one model. As Caruana (1997) explains, learning multiple variables simultaneously in one model has the potential to improve the performance of the model as each output variable can contain information that is valuable for the training of the other variable. Next to that, BNNs can easily form a distribution of predictions, by creating an ensemble of models. As discussed in section 4.2.2 the estimated variational distribution is used to form regular NNs that consist of sampled weights and bias neurons from these distributions. As we can draw an infinite number of samples from the variational distribution, we can create an infinite number of NNs, which can be employed to form an infinite number of predictions for each individual. In this manner, the uncertainty of the weights can be expressed in the uncertainty of the predictions for a particular observation (Blundell et al., 2015).

We can take an advantage of the expression of uncertainty in the BNN for the application of personalized medicine. This is achieved in the following manner. For one particular patient, we examine the predictions for each possible therapy switch. For instance, if the patient is currently taking the therapy *DMF*, we apply the combined BNN to form predictions for the CDP and the number of relapses with all possible values of index therapy, except *DMF*. Hence, the variable index therapy is adjusted to all possible values while the remaining input variables are kept the same. Afterwards, we can compare the predicted values for the two disease markers and choose the two recommended therapy switches. For example, if the predicted CDP is lowest when the index therapy equals *FTY* and the number of relapses is lowest for the therapy *NA*, both therapies are recommended. As we can easily form 100 different predictions for a particular patient by constructing an ensemble of 100 NNs, this procedure is repeated 100 times. Since two therapies are recommended in each round of predictions, the predictions of the combined BNN result in 200 recommendations. We can then examine the distribution of the recommendations. This allows us to investigate if one particular therapy is recommended more often or if all therapies

are recommended with roughly the same frequency. This expresses the uncertainty of the BNN, regarding the best possible therapy switch. Finally, we can apply the majority vote to choose the final recommendation. In this manner, the recommendation is not based on one single prediction for the patient, but on a distribution of recommendations, formed by a distribution of predictions.

Finally, note that the BNN can as well be applied to form a distribution of predictions. For instance, the ensemble of NNs can create 100 different predicted probabilities for the CDP, which can express the uncertainty regarding the predictions of a particular patient. However, the aim of this research is to examine if the combined BNN can be applied to recommend a therapy switch based on two clinical markers. For this reason, we do not further examine the ability of the BNN to form a distribution of predictions.

5 Results

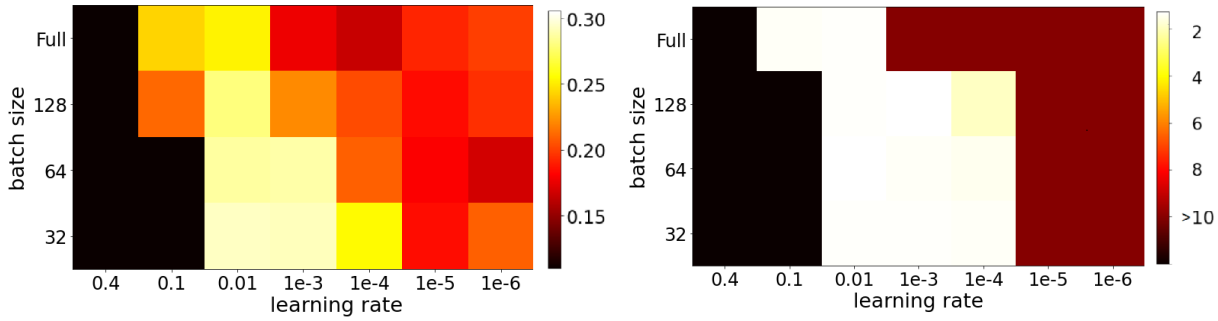
This section presents the results of the single BNNs and the combined BNN, for which the exact architecture is given in section 4.3.3 and 4.3.4, respectively. The models are trained on the synthetic data set, as described in section 3, and aim to predict the two clinical disease markers studied in this research, that is, the CDP and the number of relapses. First, we examine the outcomes of the two grid searches in section 5.1 and 5.2, for which the set-up is discussed in more detail in section 4.4.2. Next to that,we assess the stability of the final models which are trained with the optimal set of hyperparameters in section 5.3. Afterwards, in section 5.4 we compare the performance of the single BNNs with the combined BNN and compare the BNNs with the benchmark models, which are introduced in section 4.4.4. Finally, section 5.5 presents an example of how the combined BNN can be applied to form a distribution of recommended therapy switches for a particular patient.

5.1 First grid search

The results of the first grid search are shown figure 13 till 16 in Appendix D. The first two figures show the AUPRC of the predictions for the CDP of the single CDP model and the combined model. The latter two figures show the MSE of the predictions for the number of relapses of the single relapse model and the combined model. For some combinations of hyperparameters, the BNN is not able to optimize the loss (shown in equation (23)), as the loss becomes infinitely large. These models in the grid search are represented with a black box.

When we examine the results of the grid search regarding the AUPRC of the single CDP model and the combined model, it can be seen that there exist specific values for the `learning rate` that result in relatively low performing models. First of all, a `learning rate` of 0.4 or 0.1 is often too large since the training of the model fails in many of these cases. In addition, models with a `learning rate` of 0.0001 or smaller do in general lead to worse performing models. This can for example be seen in figure 7a, which displays the results of the grid search for the single CDP model in which the `network structure` is equal to [4,2]. Moreover, it is visible that the use of the

full data set in each epoch (that is, using GD instead of BGD) often leads to a low AUPRC. As BGD uses fractions of the data in each epoch, it often results in more generalized models which perform better on unseen data. This could possibly explain the better performance of BGD in the grid search. In addition, there seems to be an interaction effect between the `learning rate` and the `batch size` on the performance. However, although this effect is visible in figure 7a, it is not observed in each `network structure` of the grid search. Furthermore, when we compare the



(a) AUPRC single CDP model.

(b) MSE combined model.

Figure 7: This figure shows the performance of the first grid search. Figure 7a and 7b correspond to the model with a `network structure` of [4,2] and [2,4,10,25], respectively. Note that the brighter boxes in the grid search mark the best performing models, as the AUPRC should be maximized while the MSE should be minimized.

performance over the hyperparameter `network structure` in figure 13 and 14, it seems that the models with more hidden layers perform worse than the smaller models. This could indicate that the deeper BNNs are too complex for the purpose of predicting the CDP. However, it is interesting to note that the `network structure` of the best performing set of hyperparameters for the single CDP model, which equals [2], does not perform well for the combined model, as seen in figure 14a. It is possible that the flexibility of this `network structure` is too low for the combined model, which is trained to fit the two clinical markers simultaneously. The patterns of the AUPRC within the grid search, regarding the `learning rate`, `batch size` and `network structure`, are observed for both the single CDP as the combined model.

When we consider the results of the first grid search for the MSE of the single relapse model and the combined model in figure 15 and 16, we observe similar patterns regarding the `learning rate` and `batch size`. Nonetheless, the ranges for which the hyperparameters lead to relatively well performing models is larger. The grid search for the AUPRC results in a few sets of hyperparameters that perform considerably better than all other sets. On the contrary, the grid search for the MSE shows that many sets of hyperparameters return roughly the same performance. In addition, the effect of the `network structure` on the MSE differs. While the AUPRC seems to deteriorate for more complex BNNs, the MSE does not increase when the complexity of the BNN increases. Both differences can be observed in 7b, which presents the MSE for the combined BNN with a `network structure` of [2,4,10,25].

As described in section 4.4.2, we use 5-fold cross validation to evaluate the performance of the models in the grid search and choose the optimal set of hyperparameters. These sets are used to train the models in the second grid search. For each model, the best performing hyperparameter set is shown in table 5. As expected from the results of the grid search, each of the best performing models applies BGD and uses a `learning rate` of 0.01 or 0.001.

Table 5: Optimal hyperparameters first grid search

	Single CDP H^1_{CDP}	Single relapse $H^1_{relapse}$	Combined (CDP) $H^1_{Com:CDP}$	Combined (relapse) $H^1_{Com:rel}$
`network structure`	[2]	[8,4,2]	[25]	[2,4]
`batch size`	64	32	64	64
`learning rate`	0.001	0.01	0.001	0.01

This table shows the optimal set of hyperparameters for each model, constructed from the first grid search. The last two columns display the optimal set of hyperparameters for the combined model, decided on the AUPRC and MSE, respectively.

5.2 Second grid search

The results of the second grid search are shown in figures 17 till 20 in Appendix E. These figures demonstrate the effect of the Bayesian hyperparameters, for the best performing models of the first grid search. Within the models that apply a Gaussian prior, we can observe an interaction between the two hyperparameters `prior variance` and `variational variance rho`. Combinations in which both variances are close to zero result in better performing models. This is for example visible for the AUPRC of the combined model in figure 8. Note that a value of -5 for the `variational variance rho` corresponds to a variational variance close to zero. Low values for the `prior variance` and `variational variance rho` as well result in the best performing models for the number of relapses, although the difference between the MSE for other sets of hyperparameters is not that substantial as for the AUPRC.

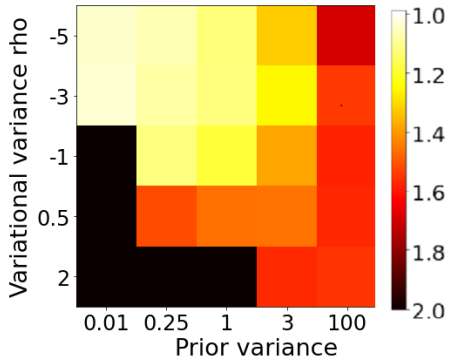
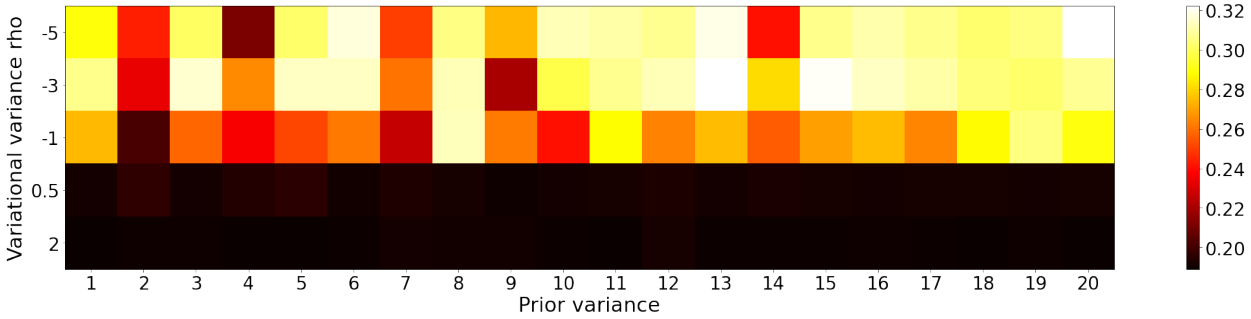


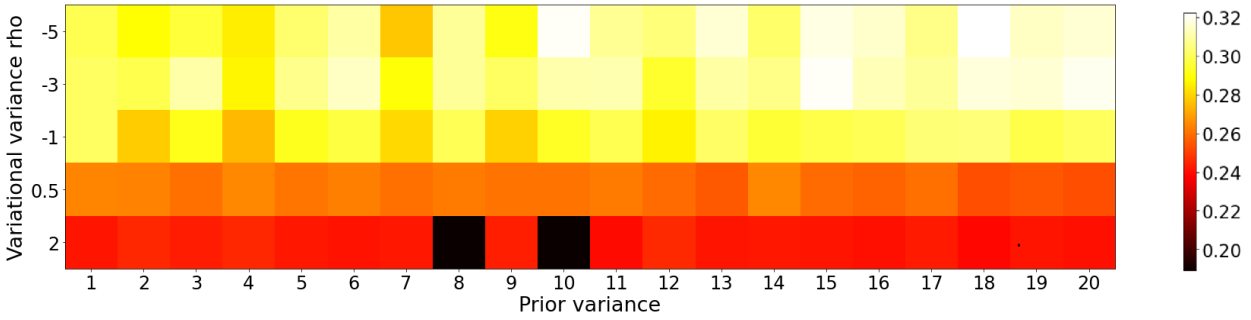
Figure 8: AUPRC second grid search for combined model with $H^2_{com:CDP}$ and the Gaussian prior.

Furthermore, the models that employ a Mixture Gaussian prior as well perform better for more negative values of the `variational variance rho` and low values of the `prior variances`.

Moreover, when we compare the second grid search of the single models (figure 17 and 19) with those of the combined models (figure 18 and 20), we observe that the Mixture Gaussian prior results in more well performing models for the combined model. This difference is well represented in figure 9, which shows the AUPRC for the single CDP model and the combined model. In general, the performance of the combined model with the Mixture Gaussian prior is similar or better compared to the single model. This observation can possibly be explained by the additional flexibility that the Mixture Gaussian grants to the weights in the BNN. In the Mixture Gaussian distribution, the weights can originate from two distributions, with different prior variances. This feature might be of higher importance for the combined model since it estimates two different output variables, connected to the hidden layers of the network with two different activation functions. It is possible that the additional flexibility of the Mixture Gaussian prior is beneficial for the combined model. The second grid search results in four different sets of hyperparameters, H^2_{CDP}. $H^2_{relapse}$,



(a) AUPRC single CDP model with H^1_{CDP}



(b) AUPRC combined model with $H^1_{Com:CDP}$

Figure 9: Second grid search for the Mixture Gaussian prior. The prior variances and mixtures probabilities are categorized with the values 1-20, for which the corresponding hyperparameter settings are shown in table 9 in Appendix E.

$H^2_{Com:CDP}$ and $H^2_{Com:rel}$. These sets of hyperparameter lead to the best performing models and are used during the training of the BNNs in the subsequent sections. The four sets are shown in table 6. It is interesting to note that three of the four hyperparameter sets apply the Mixture Gaussian prior.

Table 6: Optimal hyperparameters second grid search

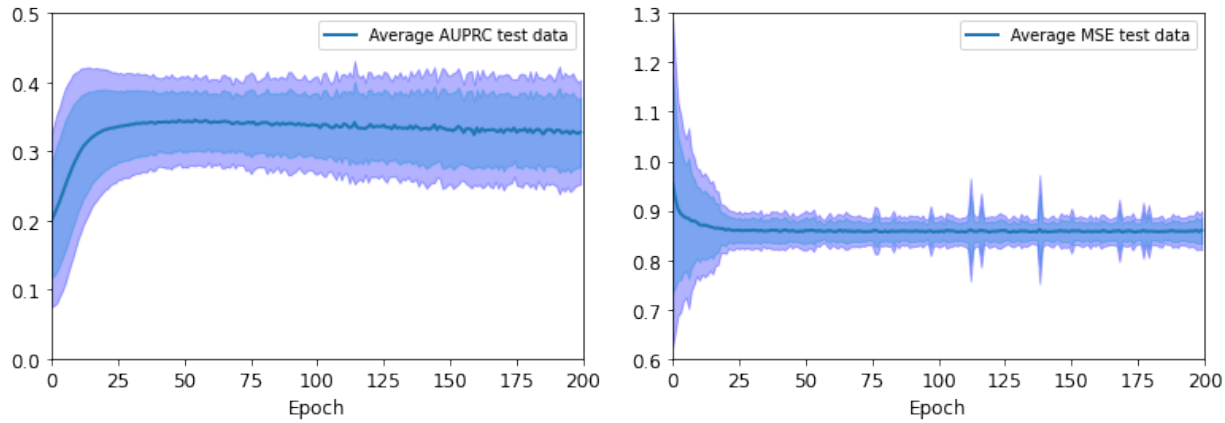
	Single CDP H^2_{CDP}	Single relapse $H^2_{relapse}$	Combined (CDP) $H^2_{Com:CDP}$	Combined (relapse) $H^2_{Com:rel}$
`network structure`	[2]	[8,4,2]	[25]	[2,4]
`batch size`	64	32	64	64
`learning rate`	0.001	0.01	0.001	0.01
`prior distribution`	Mixture Gaussian	Gaussian	Mixture Gaussian	Mixture Gaussian
`prior variance`	(1, 0.01)	0.01	(1, 0.01)	(1, 0.25)
`variational variance rho`	-3	-5	-5	-3
`mixture probability`	0.8	-	0.2	0.2

This table shows the optimal set of hyperparameters for each model, constructed from the second grid search. The last two columns display the optimal set of hyperparameters for the combined model, decided on the AUPRC and MSE, respectively.

5.3 Stability of final models

The four hyperparameter sets of the second grid search are used to form the final models, which are trained on the full training data and are evaluated on the testing data. The plots in the following sections all show the performance of the test data, which ensures that the performance of the models is judged on unseen data.

Figure 10 shows the performance for 100 different runs of the combined model. The figure on the left presents the AUPRC with the hyperparameter set $H^2_{com:CDP}$, while the figure on the right shows the MSE of the set $H^2_{com:rel}$. The difference between the 100 model is caused by



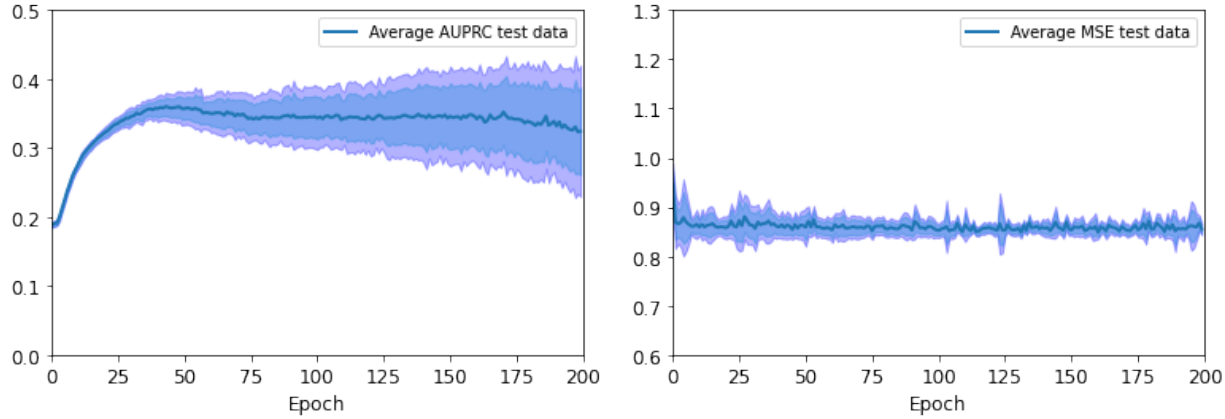
(a) AUPRC combined model with $H^2_{com:CDP}$.

(b) MSE combined model with $H^2_{com:rel}$.

Figure 10: For each epoch, this figure shows the average performance, plus and minus 2 and 3 standard deviations for 100 runs of the model.

their initialization. Since the parameters are initialized through the use of a random draw from a distribution, which is specified in section 4.3.2, the 100 models start at slightly different values. In each epoch, the average performance over the 100 models is plotted. Next to that, the variation

between the models is visualized by adding and subtracting both 2 and 3 standard deviations to the average. Examining figure 10, it appears that there exists quite some variation in the AUPRC between different initialization of the same model. On the contrary, the variation of the MSE in the first couple of epochs is relatively large, after which it reduces. This observation is similar for the results of the single models, which are shown in figure 21. However, a part of this variation can be explained by the Bayesian nature of the model. Since BNNs use random draws from the variational distribution to form the predictions, the same model with the same variational distributions can result in different predictions. In other words, the fluctuation in the performance might be due to the fluctuation in the predictions. To test this, we compare the results of the 100 different models with one model, which is used to form 100 different predictions. In other words, we train one BNN, which is employed to form a distribution of predictions. In this manner we can visualize if the variation is caused by a different initialization of the model or different predictions within the model. The results for the combined model are shown in figure 11.



(a) AUPRC combined model with $H^2_{com:CDP}$.

(b) MSE combined model with $H^2_{com:rel}$

Figure 11: For each epoch, this figure shows the average performance, plus and minus 2 and 3 standard deviations for 100 different predictions made by one model.

When we compare figure 10 and figure 11, it can be noted that the variation in the first couple of epochs is substantially larger in figure 10. Nonetheless, this can be explained by the fact that the initialization between these models differs. Throughout the epochs, the average performance of the models in the two figures seems to converge to the same value and the difference in the variation reduces. Hence, the initialization of the parameters in the BNN seems to have less influence on the performance after a couple of epochs. For this reason, we continue our analysis with one run of the BNN.

5.4 Performance of final models

The plots of the performance of all final models are shown in figure 22 in Appendix G. The single models are exclusively used to predict the CDP or the number of relapses. On the contrary, we evaluate the predictions of the combined model for both the CDP as the number of relapses. In this manner we can evaluate the total performance of the combined model, as it will be applied to model both clinical markers simultaneously.

When we examine the plots in figure 22, it seems that the average performance of all BNNs converges to a certain value and does not fluctuate much in the last epochs. In addition, it can be seen that the variation of the predictions within one model fluctuates throughout the epochs. For example, figure 11 shows that the variation of the AUPRC for the combined model increases with the number of epochs. On the contrary, the variation of the MSE is relatively smaller. This could mean that the predictions of the number of relapses are closer together than those of the CDP. Despite this difference, we cannot directly conclude that the varying performance of the 100 predictions is unfavorable, as it might be a better representation of the uncertainty of the BNN.

We compare the performance of the final models with the average performance over 100 different predictions in the last epoch. The results of the average performance for all four sets of hyperparameters are shown in table 7. This table as well shows the performance of the benchmark models. When we analyze the results, several outcomes stand out. First of all, comparing the AUPRC of the CDP predictions, we can conclude that the performance of the single CDP model is better compared to the combined model. Moreover, regarding the predictions of the number of relapses, the combined model outperforms the single relapse model. Secondly, the results show that

Table 7: Performance of BNNs and benchmark models

	CDP	*Relapse*
	AUPRC	MSE
Single BNN (H^2_{CDP})	0.349	-
Single BNN ($H^2_{relapse}$)	-	0.860
Combined BNN ($H^2_{com:CDP}$)	0.325	0.858
Combined BNN ($H^2_{com:rel}$)	0.302	0.855
Synthetic GLM	0.360	0.798
Logistic regression	0.366	-
Poisson regression	-	0.783

This table shows the performance measures of the single BNNs, the combined BNNs and the benchmark models. For the BNNs, the average performance of a distribution of 100 different predictions in the last epoch is taken.

the hyperparameter sets are of influence on the overall performance of the combined model. The combined model is trained on two different sets of hyperparameters, which are either optimal for the predictions of the CDP or the number of relapses. When we compare the performance of those models, it can be noticed that the AUPRC is better for the $H^2_{com:CDP}$ model, while the MSE is

better for the $H^2_{com:rel}$ model. This is as expected, as the former set of hyperparameters is chosen on the AUPRC and the latter on the MSE. However, it is interesting to note that the performance of the other clinical marker is in a similar range. This is especially true for the MSE of the number of relapses in the $H^2_{com:CDP}$ model, which still outperforms the MSE of the single relapse model. This indicates that the combined model can be used to construct predictions for both variables simultaneously. However, the combined model does not outperform both single models, which means that we cannot directly conclude that the benefits of multi-task learning improve the predictions of the two clinical markers.

When we compare the performance of the BNNs with the benchmark models, we can conclude that all benchmark models outperform the BNNs. This is observed for the synthetic GLMs and for the less complex logistic regression and Poisson regression. A possible explanation for this difference in performance could be the complexity of the BNNs. As each weight and bias neuron relates to two parameters in the variational distribution, there is a considerable difference in the number of parameters that needs to be estimated. This number increases quickly with the depth of the BNN and the number of neurons in the hidden layers. In addition, since NNs in general require large amounts of data, the size of the data set can be considered small as it consists of 3007 observations. A possible explanation for the lower performance might be that the benefits of the additional flexibility of a BNN are not completely employed, as the training data is too small to detect all complex relationships in the data. This makes the less complex models preferable with respect to the performance. While the performance of the benchmark models is superior to that of the BNNs, the BNNs have characteristics that can be valuable in some applications. In addition to the simultaneous predictions of both clinical markers, the BNNs can express the uncertainty over the predictions for a particular patient. As these characteristics of the BNN are useful in the context of personalized medicine, there exists a trade off between these characteristics and the performance of the predictions. It depends on the application if we are willing to lose some prediction accuracy, in exchange for these characteristics. The following section analyses if the combined BNN can be used to express uncertainty in the application of medical decision making.

We investigate the effects of three extensions, which can possibly improve the performance of the BNN. We examine the use of a larger synthetic data set, the sigmoid activation function and the early stopping algorithm. The results of these extensions are shown in Appendix H, I and J, respectively. Although the performance of some of the BNNs improves, none of the extensions consistently improves the performance of all BNNs.

5.5 Distribution of recommendations

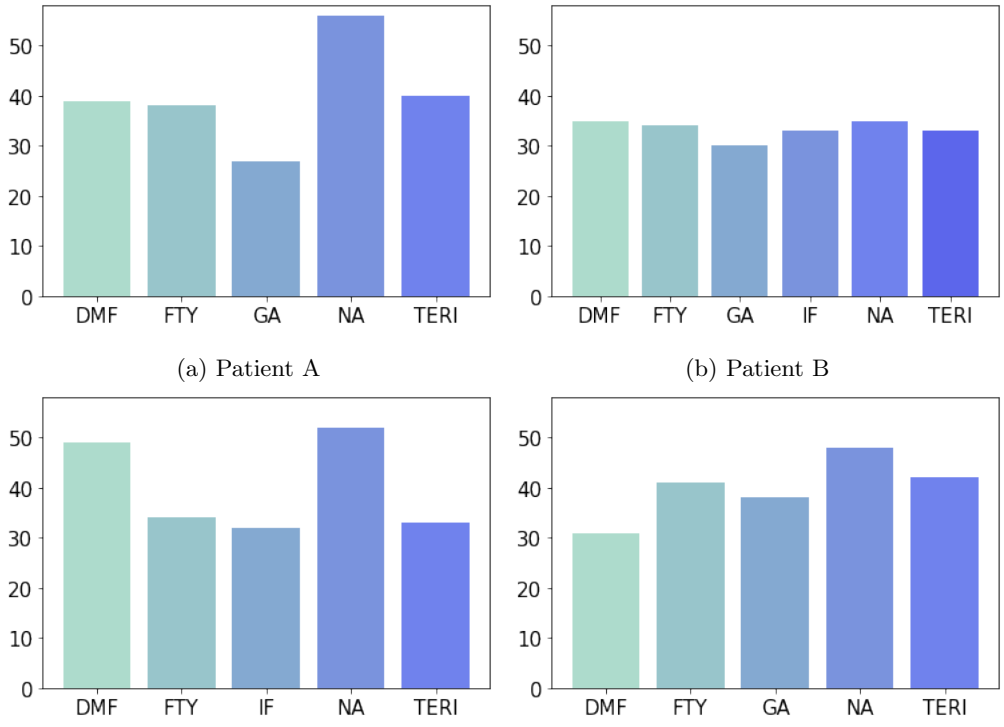
In this section we apply the combined BNN to express the uncertainty regarding the recommendation of the best possible therapy switch for four patients in the test data set. Table 8 shows the observed values of several variables for each patient. The DMD that is taken as current therapy is not used in the recommendation, as the BNN is trained to model therapy switches. We use the

combined BNN, which is trained with the hyperparameter set $H^2_{com:CDP}$ and apply the methodology discussed in section 4.5 to form a distribution of recommendations. Both the predictions of the CDP as the number of relapses are applied to recommend an optimal therapy switch in each of the 100 predictions, which implies that a total of 200 recommendations is made.

Table 8: Characteristics of the four patients

	CDP	number of relapses	current therapy	index therapy
Patient A	1	1	IF	FTY
Patient B	0	0	NoDMD	DMF
Patient C	0	1	GA	FTY
Patient D	1	3	IF	TERI

Figure 12 shows the frequency with which each possible index therapy is chosen. Based on these figures, we can make one final recommendation for each patient by means of the majority vote. For example, the index therapy *NA* is recommended for patient A, as it is recommended 56 out of 200 time. Although we can make one final recommendation for each patient, the figures show that the uncertainty of the model regarding the optimal therapy switch is quite large. None of



(a) Patient A

(b) Patient B

(c) Patient C

(d) Patient D

Figure 12: This figure shows how often each possible therapy switch is recommended based on 100 different predictions of the combined model. Each prediction returns two recommendations, based on the CDP and the number of relapses. The combined model is trained with the hyperparameter set $H^2_{com:CDP}$.

the recommendations for the four patients results in one recommended therapy switch, which is recommended much more often than the other therapies. It is probable that patients or doctors consider this uncertainty as too large when they prefer a different therapy switch, for example, because of certain side effects. In addition, the uncertainty within the recommendations of each patient does not seem to increase for more uncommon values of the variables CDP and the number of relapses. For instance, patient D has a CDP of 1 and experiences 3 relapses, which is less frequently observed in the data set, as seen in figure 1 in section 3.2. However, it is not clear that the combined BNN is more uncertain about this patient, than it is about the more frequently observed values of patient C.

Based on these figures we can conclude that the combined BNN is able to express its uncertainty about the recommended therapy switch. Nevertheless, it is not clear if the uncertainty in the recommendations differs between different type of patients. Finally, for the application of medical decision making, the uncertainty between the possible therapy switches is quite large.

6 Conclusion

MS is a destructive disease, that influences the lives of patients on a physical and mental level. Throughout the years, patients suffer from progressive irreversible disability and relapses during which the symptoms of MS worsen (Reich et al., 2018). Furthermore, patients often experience stress or anxiety. For that reason, it is relevant to find effective DMDs that slow down the progression of MS. However, as the disease courses of patients and their treatment responses to different DMDs are heterogeneous, it is challenging to identify the best possible DMD for a particular patient (Derfuss, 2012). This is further complicated by the fact that the exact cause of MS is not yet understood (Reich et al., 2018). In addition, the clinical markers that are commonly used to describe the disease course of a patient might not be a complete representation of the disease course. For this reason, it is valuable to consider multiple clinical markers to evaluate the effectiveness of different DMDs (Goodin et al., 2002).

The research of Stühler et al. (2020) uses two separate models to predict the number of relapses and the CDP, which represents the disability progression of a patient. For this, the authors use a data set which consists of demographic information about the patients and their previous disease course. Each patient in the data sets is switching to a new therapy cycle with another DMD. By predicting the CDP and the number of relapses for each possible therapy switch, they are able to show which therapy switches are suitable for a particular patient. This information can be used in the discussion between the patient and the doctor.

This research extends the research of Stühler et al. (2020) by using BNNs to recommend the best possible therapy switch for MS patients. The characteristics of BNNs result in several advantages in this application. First of all, we can apply the BNN to model the two clinical markers simultaneously, instead of using two separate models. This can potentially improve the perfor-

mance of the model, as it is possible that the output variables in multi-task learning models are able to learn from each other (Caruana, 1997). BNNs can model multiple variables simultaneously without making any assumptions about the relationships between the variables. This makes them suitable models to describe the two clinical markers simultaneously, as the relationships between the markers is shown to be complex (Confavreux et al., 2003; Scalfari et al., 2010). Next to that, BNNs are flexible models which can potentially described the complex relationships in the data of the heterogeneous disease. Finally, contrary to regular NNs, BNNs can express the uncertainty of the model regarding the predictions of particular observations (Blundell et al., 2015). BNNs aim to estimate the probability distribution of the weights in the network. Once the distributions are estimated, the BNN can be applied to form an ensemble of NNs. This ensemble can be used to create a distribution of recommendations, which means that the BNN is able to express the uncertainty about the possible therapy switches. Since the therapy switch can be influential on a patient's life, this expression of uncertainty is essential in the conversation between the doctor and patient.

To evaluate if the potential benefits from multi-task learning improve the performance of a combined BNN, we as well construct two single BNNs. These single BNNs are trained to predict either the number of relapses or the CDP. To train the BNNs, we use a synthetic data set, which is based on the original data set of Stühler et al. (2020). This synthetic data set is as well used to construct several benchmark models to compare the performance of the BNNs. We re-estimate the GLMs as described in the research of Stühler et al. (2020) and estimate two less complex GLMs.

The results of this research show that the combined BNN does not consistently outperform the two single BNNs. Although the predictions for the number of relapses improves for the combined model, the predictions for the CDP worsens. Moreover, when we compare the performance of the BNNs to that of the less complex benchmark models, the benchmark models consistently outperform the BNNs. This indicates that the complexity of the BNN is too high for the application in this research. This observation might be explained by the data set, which consists of 3007 patients. BNNs are highly flexible models, yet it is possible that this flexibility cannot be fully employed as the data set is too small for the BNN to learn the complex relationships.

The BNN still has the advantage that it can express the uncertainty of the model regarding the possible therapy switches. When we examine the distribution of recommendations for four different patients, we observe that the BNN is quite uncertain about the best possible therapy switch. Although it is possible to detect one therapy switch which is recommended most often, we observe that each other therapy switch is as well recommended quite frequently. In other words, it might be too uncertain that the recommended therapy switch is actually better than the other therapy switches. While it is favorable that the BNN can express its uncertainty, it is probable that the level of uncertainty is too high for doctors and patients.

To conclude, BNNs have characteristics that are valuable in the application of medical decision making. Besides their high flexibility, the models can express the uncertainty about specific pre-

dictions. In addition, it is a suitable model to utilize the benefits of multi-task learning and predict multiple clinical markers simultaneously, as BNNs do not require any assumptions about the relationships between the markers. Nonetheless, the results in this research show that the uncertainty of the BNN regarding the best possible therapy switch for MS patients is quite large and that less complex models result in more accurate predictions on the available data set. This leads to the conclusion that the recommendations of the combined BNN are too uncertain for this particular application.

In further research it remains of interest to investigate if the performance of BNNs improves for larger data sets with more MS patient or if they are applicable for other complex diseases. In the current setting we do not recommend using this specific model in the decision making process for MS patients. Nevertheless, the combined BNN does show certain features that can be valuable in personalized medicine. If the performance of the BNN can meet the performance of existing models in different medical applications, the benefits of BNNs are an interesting addition to the conversation between the patient and doctor. Note that a recommendation made by the BNN does not have the be leading in the medical decision, as the uncertainty regarding different types of drugs can as well be considered as additional information for the patient or doctor before they make the final choice.

There are several limitations in this research that can possibly be improved. First of all, it is possible that the potential of the BNN is not fully utilized in this research. The data set is quite small and it is possible that some of the complex relationships are lost during the creation of the synthetic data set. Next to that, the imbalanced nature of the data set might complicate the training of the BNN. It is possible that the performance of the BNN improves once a larger data set is available or once we account for the imbalance in the data. However, further research is needed to examine these possibilities. Secondly, BNNs can be constructed with many different network architectures. For example, the number of relapses can as well be considered as a categorical variable, for which we create separate output neurons for each possible value. Next to that, the combined model can take different structures in the hidden layer. It is possible to separate the neurons in the hidden layers from a certain layer onward, after which the neurons are exclusively trained to fit either the CDP or the number of relapses. As there are many possible extensions to the BNNs, there is room for many possible improvements.

References

Bejarano, B., Bianco, M., Gonzalez-Moron, D., Sepulcre, J., Goñi, J., Arcocha, J., ... Villoslada, P. (2011). Computational classifiers for predicting the short-term course of multiple sclerosis. *BMC neurology*, *11*(1), 67.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Caruana, R. (1997). Multitask learning. *Machine learning*, *28*(1), 41-75.

Confavreux, C., Vukusic, S., & Adeleine, P. (2003). Early clinical predictors and progression of irreversible disability in multiple sclerosis: an amnesic process. *Brain*, *126*(4), 770-782.

Derfuss, T. (2012). Personalized medicine in multiple sclerosis: hope or reality? *BMC medicine*, *10*(1), 116.

Dilsizian, S. E., & Siegel, E. L. (2014). Artificial intelligence in medicine and cardiac imaging: harnessing big data and advanced computing to provide personalized medical diagnosis and treatment. *Current cardiology reports*, *16*(441).

Goodin, D. S., Frohman, E. M., Garmany, G. P., Halper, J., Likosky, W. H., Lublin, F. D., ... van den Noort, S. (2002). Disease modifying therapies in multiple sclerosis. *Neurology*, *58*(2), 169-178.

Graves, A. (2011). Practical variational inference for neural networks. *Advances in neural information processing systems*, *24*, 2348-2356.

Greenberg, E. (2012). *Introduction to bayesian econometrics.* Cambridge University Press.

Hastie, T., Tibshirani, R., & Friedman. (2009). *The elements of statistical learning: data mining, inference, and prediction.* Springer Science Business Media.

Hinton, G. E., & Van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. *Proceedings of the 16th Annual Conference On Learning Theory (COLT)*, 5–13.

Itchhaporia, D., Snow, P. B., Almassy, R. J., & Oetgen, W. J. (1996). Artificial neural networks: current status in cardiovascular medicine. *Journal of the American College of Cardiology*, *28*(2), 515-521.

Jain, B. A., & Nag, B. N. (1997). Performance evaluation of neural network decision models. *Journal of Management Information Systems*, *14*(2), 201-216.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*.

Kobelt, G., Berg, J., Lindgren, P., Fredrikson, S., & Jönsson, B. (2006). Costs and quality of life of patients with multiple sclerosis in europe. *Journal of Neurology, Neurosurgery Psychiatry*, *77*(8), 918-926.

Miller, A., Avidan, N., Tzunz-Henig, N., Glass-Marmor, L., Lejbkowicz, I., Pinter, R. Y., & Paperna, T. (2008). Translation towards personalized medicine in multiple sclerosis. *Journal of the neurological sciences*, *274*(1-2), 68-75.

Neal, R. M. (2012). Bayesian learning for neural networks. *Springer Science Business Media.*, *118*.

Nielsen, M. A. (2015). *Neural networks and deep learning.* Determination press.

Reich, D. S., Lucchinetti, C. F., & Calabresi, P. A. (2018). Multiple sclerosis. *The New England Journal of Medicine*, *378*(2), 169–180.

Rovaris, M., Confavreux, C., Furlan, R., Kappos, L., Comi, G., & Filippi, M. (2006). Secondary progressive multiple sclerosis: current knowledge and future challenges. *The Lancet Neurology*, *5*(4), 343-354.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, *10*(3), e0118432.

Scalfari, A., Neuhaus, A., Degenhardt, A., Rice, G. P., Muraro, P. A., Daumer, M., & Ebers, G. C. (2010). The natural history of multiple sclerosis, a geographically based study 10: relapses and long-term disability. *Brain*, *133*(7), 1914-1929.

Schork, N. J. (2015). Personalized medicine: time for one-person trials. *Nature*, *520*, 609-611.

Stühler, E., Braune, S., Lionetto, F., Heer, Y., Jules, E., Westermann, C., ... Group., N. S. (2020). Framework for personalized prediction of treatment response in relapsing remitting multiple sclerosis. *BMC medical research methodology*, *20*(1), 20-24.

Trapp, B. D., Peterson, J., Ransohoff, R. M., Rudick, R., Mörk, S., & Bö, L. (1998). Axonal transection in the lesions of multiple sclerosis. *New England Journal of Medicine*, *338*(5), 278-285.

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q. V., ... Hinton, G. E. (2013). *On rectified linear units for speech processing.* IEEE.

Zhou, Z. H., Jiang, Y., Yang, Y. B., & Chen, S. F. (2002). Lung cancer cell identification based on artificial neural network ensembles. *Artificial Intelligence in Medicine*, *24*(1), 25-36.

Appendices

A Abbreviations

ADAM - adaptive moment estimation
AUPRC - area under the precision recall curve
BGD - batch gradient descent
BNN - Bayesian Neural Network
CDP - confirmed disease progression

CNS - Central nervous system
DMDs - Disease modifying drugs
DMF - Dimethylfumarat
EDSS - expanded disability status scale
FN - False negative
FP - False positive
FTY - Fingolimod
GA - Glatirameracetat
GD - gradient descent
GLM - generalized linear model
IF - Interferon-ß1
KL - Kullback- Leibler
MS - Multiple sclerosis
MSE - Mean squared error
NA - Natalizumab
NN - Neural network
NoDMD - no previous Disease modifying drug
NTD - NeuroTransData
ReLu - rectified linear unit
RRMS - Relapsing remitting multiple sclerosis
SGD - stochastic gradient descent
SPMS - Secondary progressive Multiple Sclerosis
SSE - sum of squared error
TERI - Teriflunomide
TN - True negative
TP - True positive

B Adam

The Adam algorithm of Kingma & Ba (2014) makes use of an adaptive learning rate by updating two moment estimates in each epoch. In each epoch k, the algorithm derives the gradient g_k. To match the notation of equation (2) in this research, note that $g_k = \frac{\delta C_{s \in B}}{\delta W_k}$. Next, the gradient is used to estimate the two moments with the following equations,

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k \qquad\qquad v_k = \beta_2 v_{k-1} + (1 - \beta_2) g_k^2 \tag{30}$$

The values β_1 and β_2 in these equations are fixed throughout the algorithm and can take values in the range $[0, 1)$. As the two moment estimates are biased, the next step corrects the biases:

$$\hat{m_k} = \frac{m_k}{(1 - \beta_1^k)} \qquad\qquad \hat{v_k} = \frac{v_k}{(1 - \beta_2^k)} \tag{31}$$

Finally, these bias corrected terms are used to update the parameters of the model.

$$W_{k+1} = W_k - \frac{\lambda}{\sqrt{\hat{v_k}} + \epsilon} \hat{m_k} \tag{32}$$

Based on empirical testing, the authors propose default values of $\lambda = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1^{-7}$. During this research, the value of λ is set to several different values, while the values of β_1, β_2 and ϵ are set to the proposed default values.

C Backpropagation

Assume that the activation functions of all layers are set to $f()$ and that there are no bias neurons in the network. Then the gradients with respect to the weight matrices W^l and W^{l-1} equal

$$\frac{\delta C}{\delta W^l} = \frac{\delta C}{\delta a^L} \frac{\delta a^L}{\delta z^L} \frac{\delta z^L}{\delta a^{L-1}} ... \frac{\delta a^l}{\delta z^l} \frac{\delta z^l}{\delta W^l} \tag{33}$$

$$\frac{\delta C}{\delta W^{l-1}} = \frac{\delta C}{\delta a^L} \frac{\delta a^L}{\delta z^L} \frac{\delta z^L}{\delta a^{L-1}} ... \frac{\delta a^l}{\delta z^l} \frac{\delta z^l}{\delta a^{l-1}} \frac{\delta a^{l-1}}{\delta z^{l-1}} \frac{\delta z^{l-1}}{\delta W^{l-1}} \tag{34}$$

If we ignore the last term in equation (33) and (34), the two gradients only differ by the extra multiplication of $\frac{\delta z^l}{\delta a^{l-1}} \frac{\delta a^{l-1}}{\delta z^{l-1}}$. Backpropagation uses the chain rule to recursively retrieve the gradients of the weight matrices in all layers, without making duplicate computations. Let

$$d^L = \frac{\delta C}{\delta a^L} \frac{\delta a^L}{\delta z^L}$$

Starting from d^L, each layer can be retrieved in the following manner,

$$\begin{aligned} d^{L-1} &= d^L \frac{\delta z^L}{\delta a^{L-1}} \frac{\delta a^{L-1}}{\delta z^{L-1}} \\ d^{L-2} &= d^{L-1} \frac{\delta z^{L-1}}{\delta a^{L-2}} \frac{\delta a^{L-2}}{\delta z^{L-2}} \\ &... \\ d^1 &= d^2 \frac{\delta z^2}{\delta a^1} \frac{\delta a^1}{\delta z^1} \end{aligned}$$

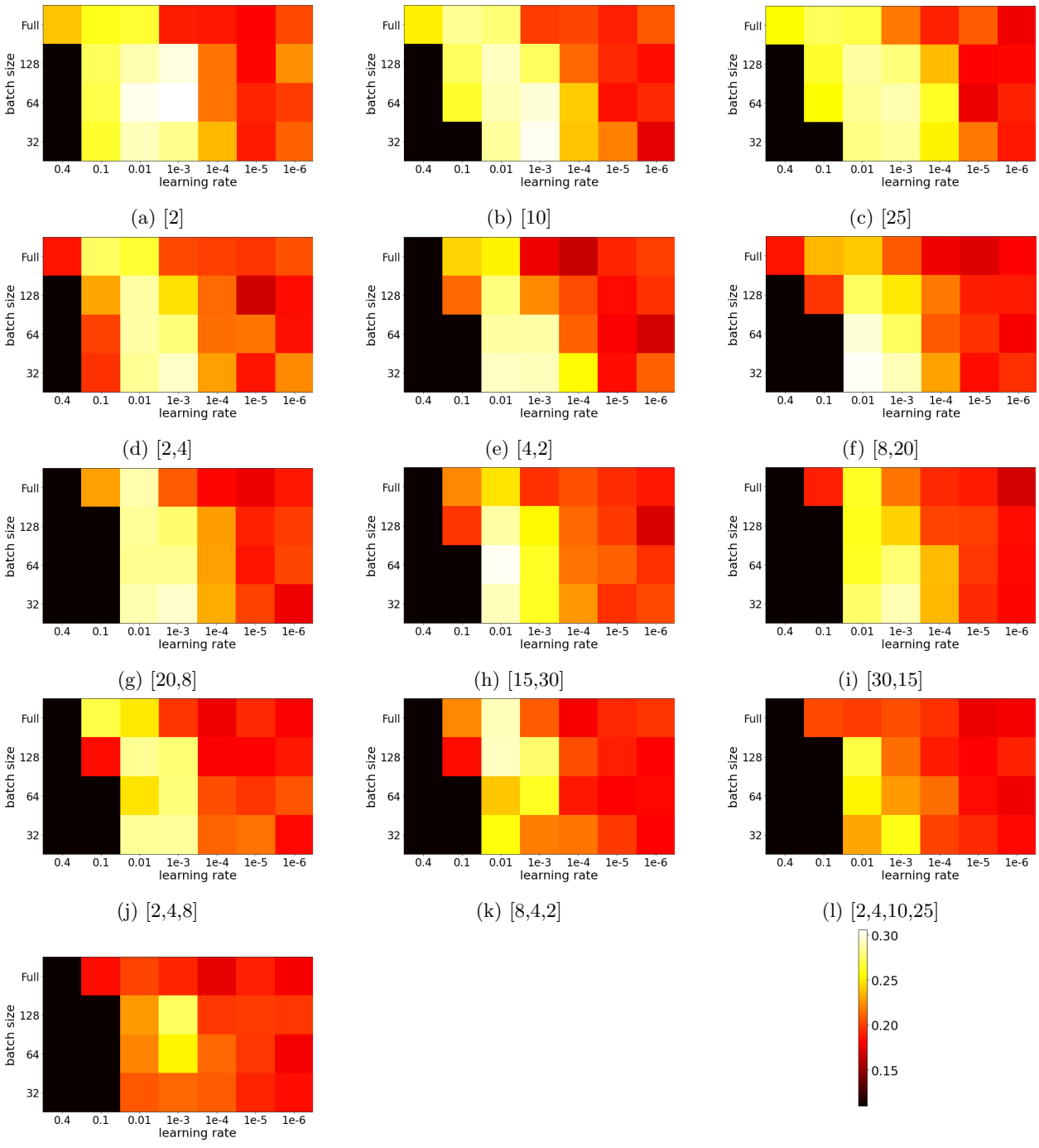
This results in the recursive relation,

$$d^l = d^{l+1} \frac{\delta z^{l+1}}{\delta a^l} \frac{\delta a^l}{\delta z^l} \tag{35}$$

Finally, adding back the last term of (33) which we previously ignored, we find

$$\frac{\delta C}{\delta W^l} = d^l \frac{\delta z^l}{\delta W^l} \tag{36}$$

D Results of first grid search



(a) [2] (b) [10] (c) [25]

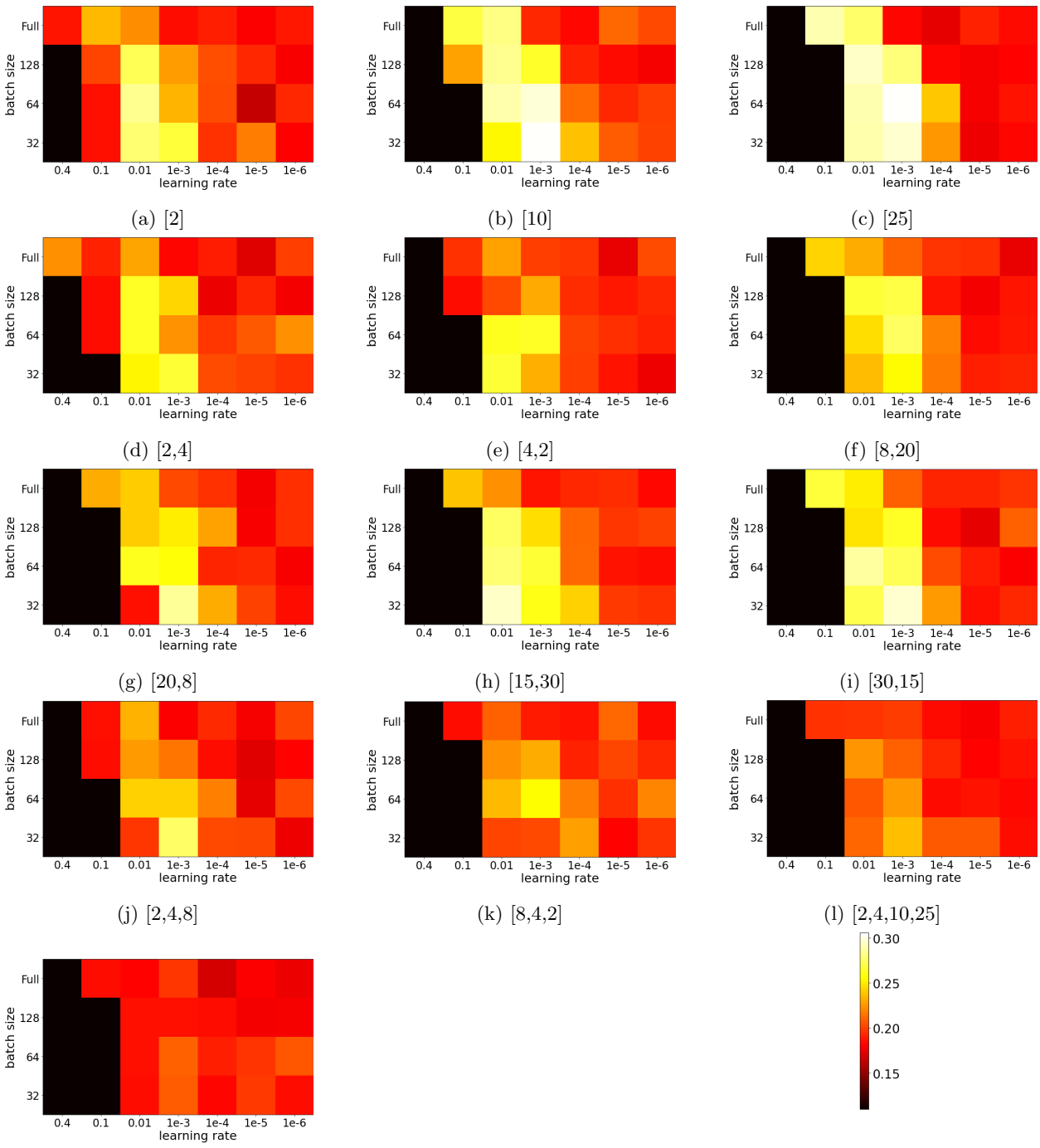
(d) [2,4] (e) [4,2] (f) [8,20]

(g) [20,8] (h) [15,30] (i) [30,15]

(j) [2,4,8] (k) [8,4,2] (l) [2,4,10,25]

(m) [25,10,4,2] (n) AUPRC legend

Figure 13: First grid search: AUPRC for the single CDP model. The results are given at different values for the `batch size`, `learning rate` and `network structure`. A black output means the BNN was not able to optimize the loss.



(a) [2] (b) [10] (c) [25]

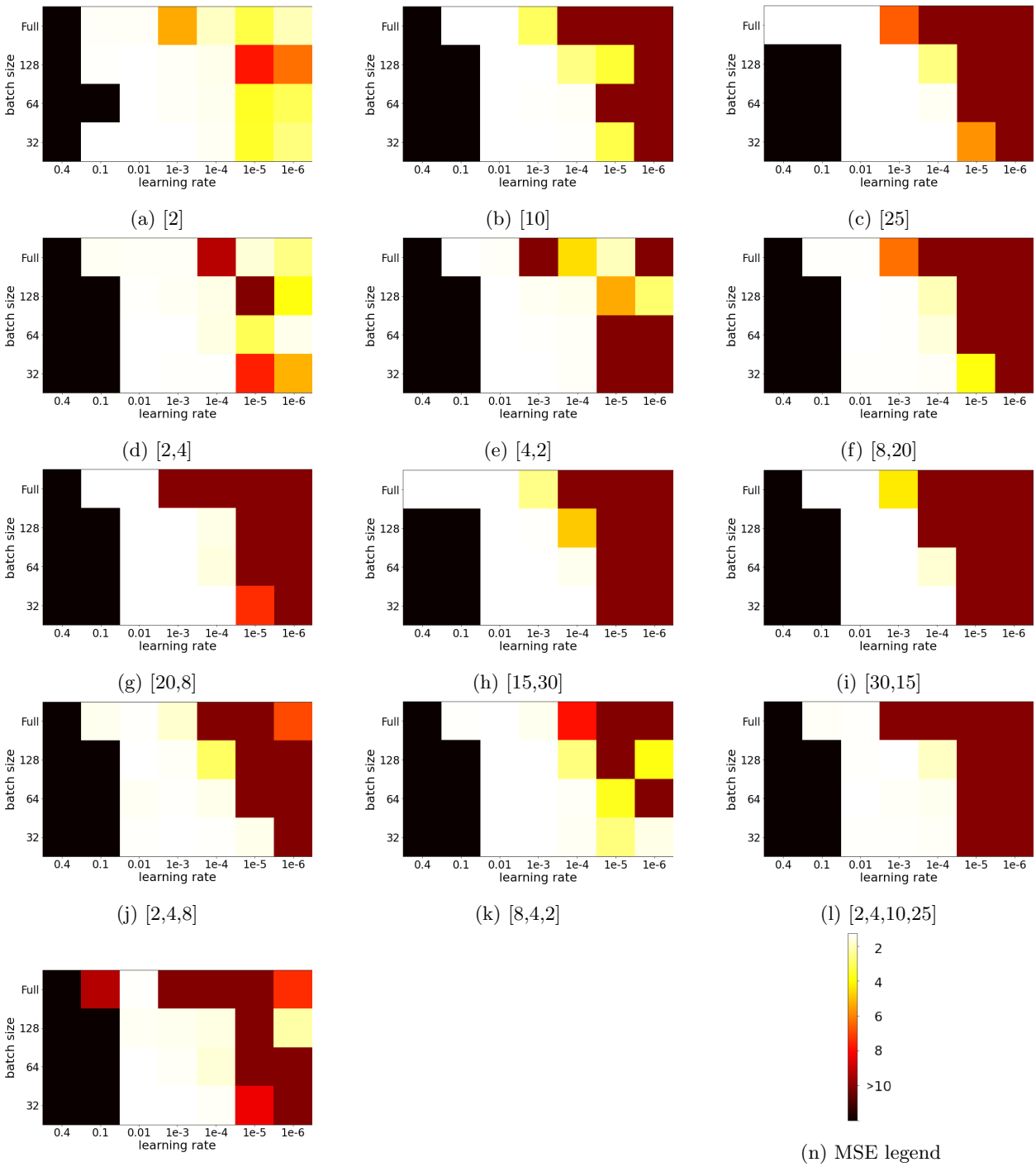
(d) [2,4] (e) [4,2] (f) [8,20]

(g) [20,8] (h) [15,30] (i) [30,15]

(j) [2,4,8] (k) [8,4,2] (l) [2,4,10,25]

(m) [25,10,4,2] (n) AUPRC legend

Figure 14: First grid search: AUPRC for the combined model. The results are given at different values for the `batch size`, `learning rate` and `network structure`. A black output means the BNN was not able to optimize the loss.



(a) [2] (b) [10] (c) [25]

(d) [2,4] (e) [4,2] (f) [8,20]

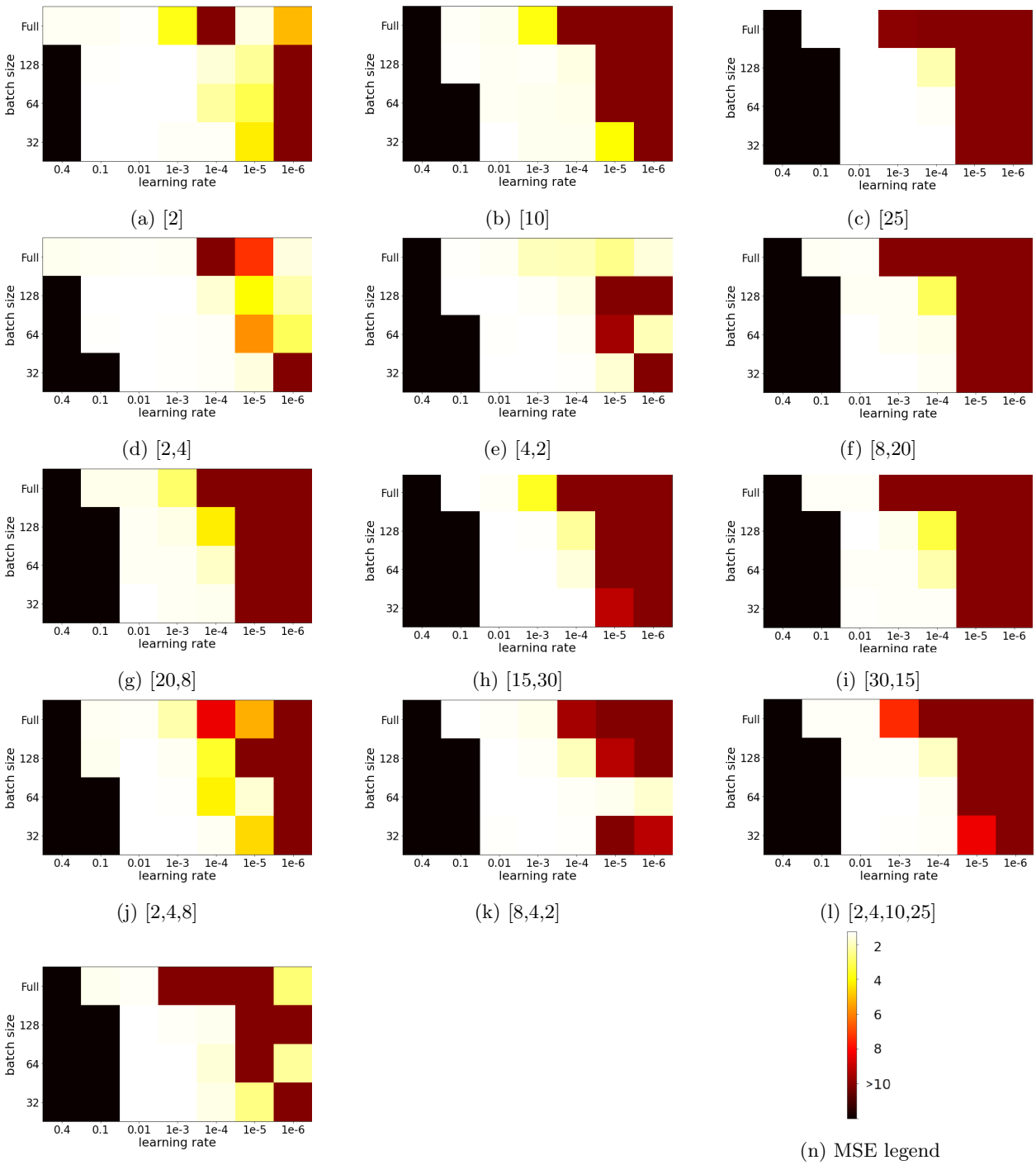
(g) [20,8] (h) [15,30] (i) [30,15]

(j) [2,4,8] (k) [8,4,2] (l) [2,4,10,25]

(m) [25,10,4,2] (n) MSE legend

.

Figure 15: First grid search: MSE for the single relapse model. The results are given at different values for the `batch size`, `learning rate` and `network structure`. A black output means the BNN was not able to optimize the loss.



(a) [2]

(b) [10]

(c) [25]

(d) [2,4]

(e) [4,2]

(f) [8,20]

(g) [20,8]

(h) [15,30]

(i) [30,15]

(j) [2,4,8]

(k) [8,4,2]

(l) [2,4,10,25]

(m) [25,10,4,2]

(n) MSE legend

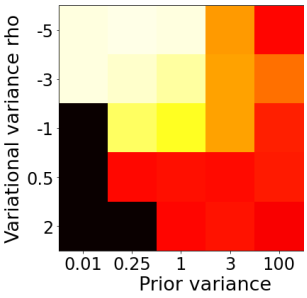
.

Figure 16: First grid search: MSE for the combined model. The results are given at different values for the `batch size`, `learning rate` and `network structure`. A black output means the BNN was not able to optimize the loss.

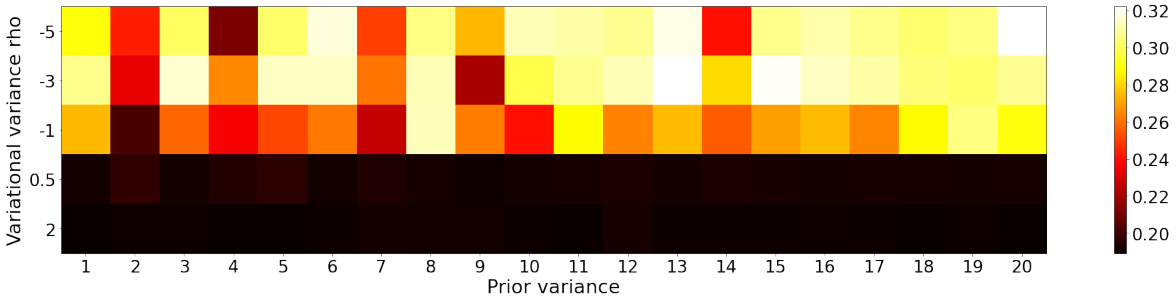
E Results of second grid search

Table 9: Corresponding hyperparameter setting for the categories in the Gaussian Mixture Prior.

Category	`prior variance`	`mixture probability`
1	(1, 0.25)	0.95
2	(3, 0.25)	0.95
3	(1, 0.01)	0.95
4	(3, 0.01)	0.95
5	(1, 0.0001)	0.95
6	(1, 0.25)	0.8
7	(3, 0.25)	0.8
8	(1, 0.01)	0.8
9	(3, 0.01)	0.8
10	(1, 0.0001)	0.8
11	(1, 0.25)	0.5
12	(3, 0.25)	0.5
13	(1, 0.01)	0.5
14	(3, 0.01)	0.5
15	(1, 0.0001)	0.5
16	(1, 0.25)	0.2
17	(3, 0.25)	0.2
18	(1, 0.01)	0.2
19	(3, 0.01)	0.2
20	(1, 0.0001)	0.2

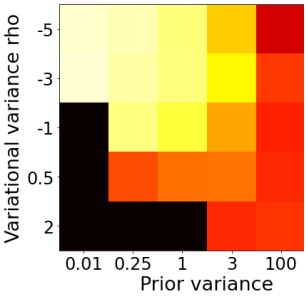


(a) Gaussian prior

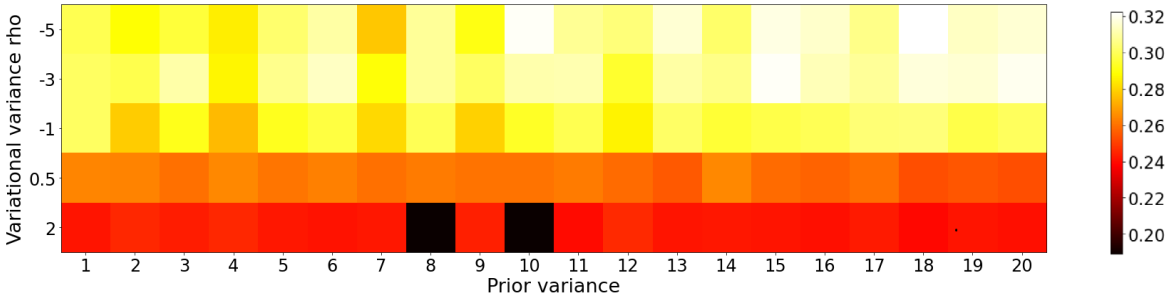


(b) Mixture Gaussian prior

Figure 17: Second grid search: AUPRC for the single CDP model with H^1_{CDP}. The categories on the x-axis in figure 17b are explained in table 9

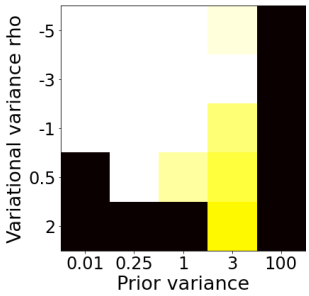


(a) Gaussian prior

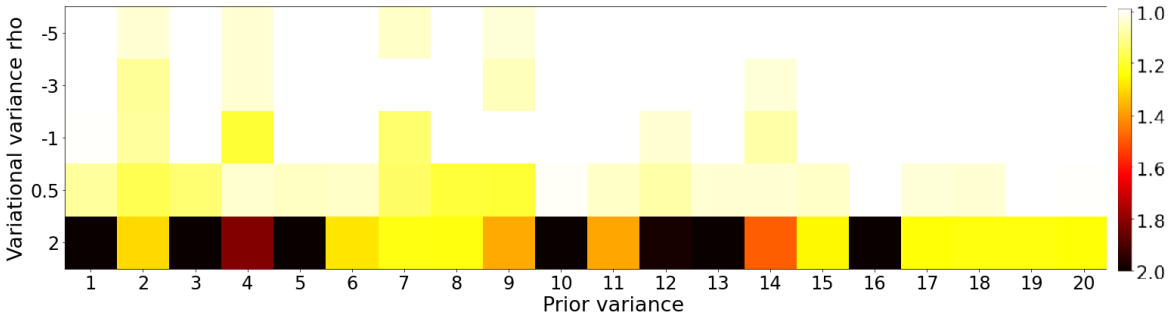


(b) Mixture Gaussian prior

Figure 18: Second grid search: AUPRC for the combined model with $H^1_{Com:CDP}$. The categories on the x-axis in figure 18b are explained in table 9

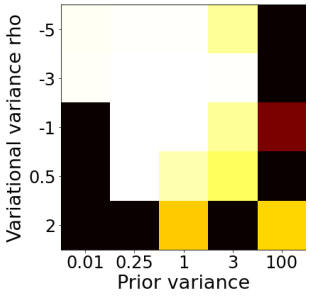


(a) Gaussian prior

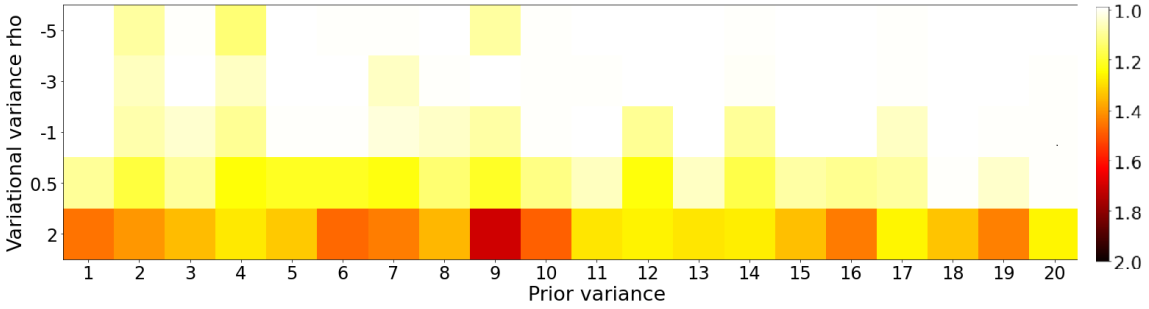


(b) Mixture Gaussian prior

Figure 19: Second grid search: MSE for the single relapse model with $H^1_{relapse}$. The categories on the x-axis in figure 19b are explained in table 9



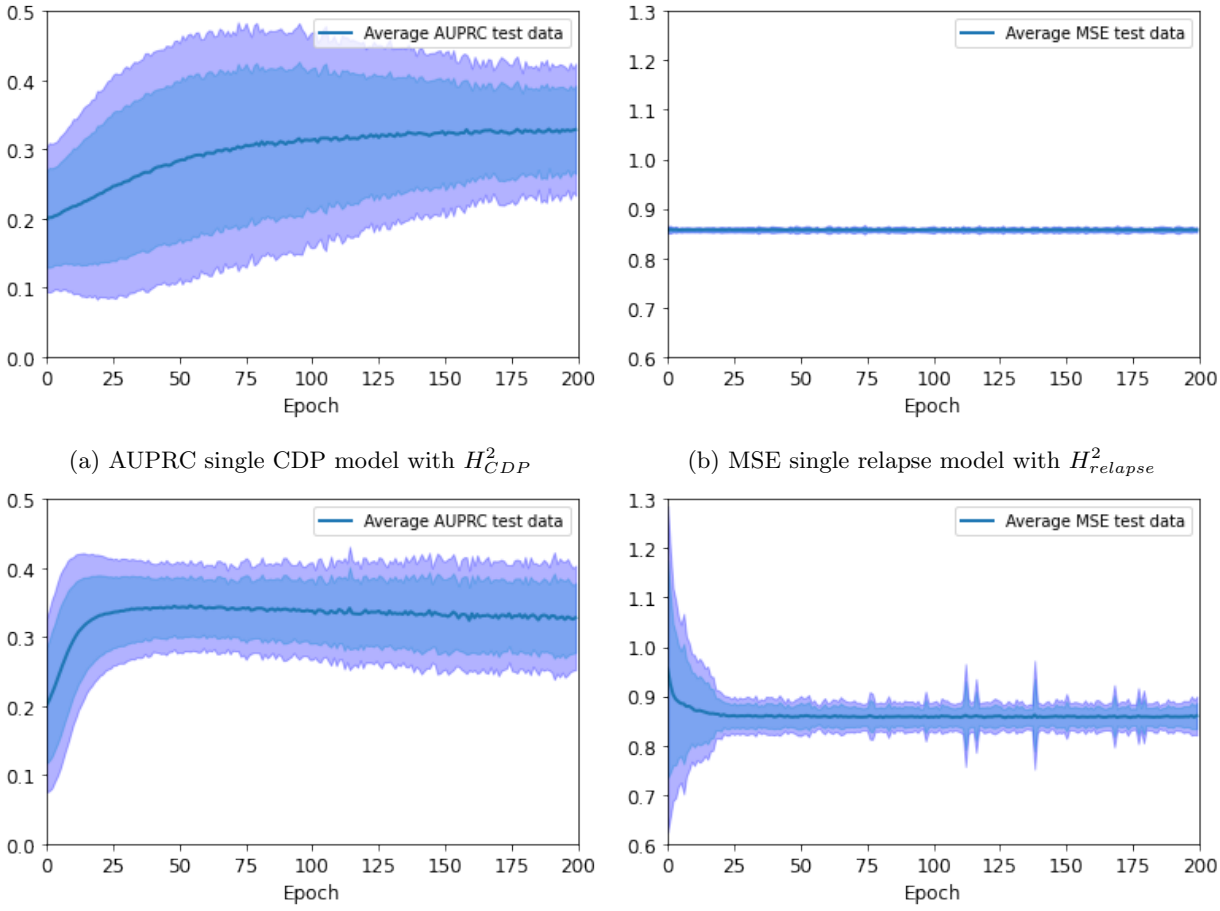
(a) Gaussian prior



(b) Mixture Gaussian prior

Figure 20: Second grid search: MSE for the combined model with $H^1_{Com:rel}$. The categories on the x-axis in figure 20b are explained in table 9

F Results of final model: stability



(a) AUPRC single CDP model with H^2_{CDP}

(b) MSE single relapse model with $H^2_{relapse}$

(c) AUPRC combined model with $H^2_{com:CDP}$

(d) MSE combined model with $H^2_{com:rel}$

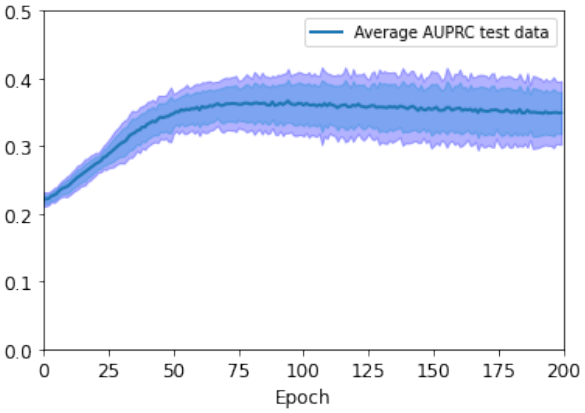
Figure 21: This figure shows the stability of the final models. For each epoch, the graph shows the average performance, plus and minus 2 and 3 standard deviations for 100 runs of the model.

Table 10: Performance of 100 BNNs with different values of initialization

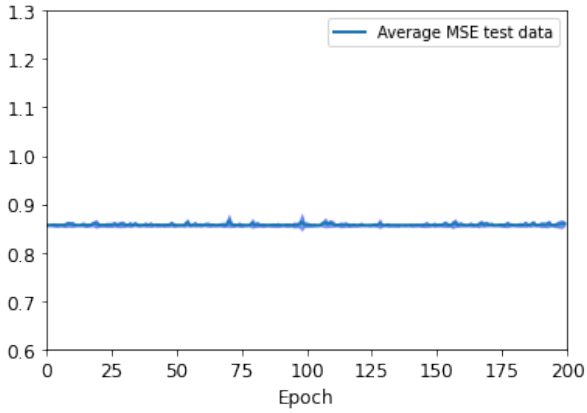
	AUPRC	MSE
Single BNN (H^2_{CDP})	0.329	-
Single BNN ($H^2_{relapse}$)	-	0.857
Combined BNN ($H^2_{com:CDP}$)	0.328	-
Combined BNN ($H^2_{com:rel}$)	-	0.860

This table shows the average performance of the stability plots in figure 21. The performance is taken as the average performance of the predictions of the 100 different BNNs in the last epoch.

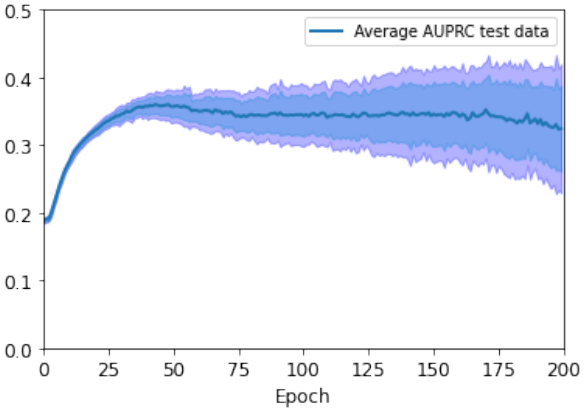
G Results of final model: performance



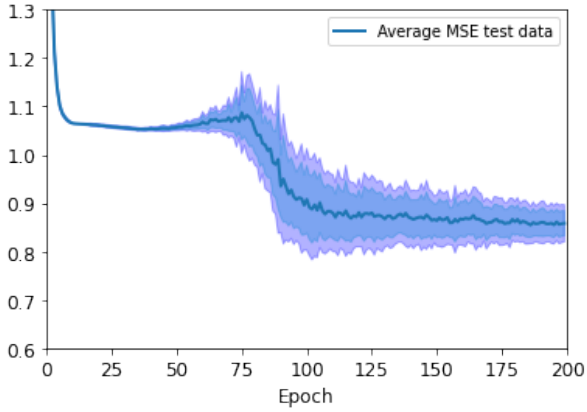
(a) AUPRC single CDP model with H^2_{CDP}



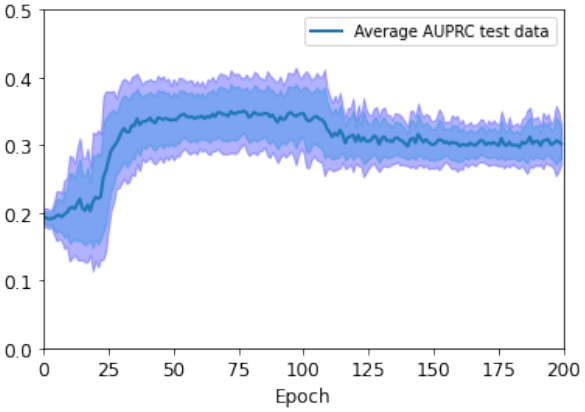
(b) MSE single relapse model with $H^2_{relapse}$



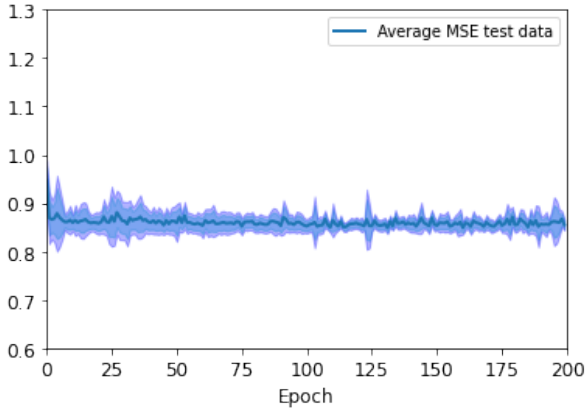
(c) AUPRC combined model with $H^2_{com:CDP}$



(d) MSE combined model with $H^2_{com:CDP}$



(e) AUPRC combined model with $H^2_{com:rel}$



(f) MSE combined model with $H^2_{com:rel}$

Figure 22: This figure shows the final performance of the single model on the first row and the combined model on the second and third row. In each epoch, 100 different predictions are made. The graph shows the average performance, plus and minus 2 and 3 standard deviations.

H Large data set

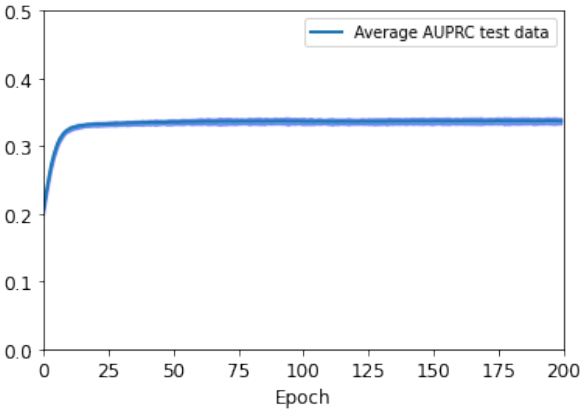
As Blundell et al. (2015) address, the uncertainty in a BNN can decrease when the data set increases. In this case, the BNN has more training data which can be used to learn the relationships in the data. To examine this possibility, we create a synthetic data set of 30,000 observations to train the BNNs. This is accomplished by modifying step 2 in the creation of the synthetic data set, as explained in section 3.2. Instead of setting the sample size equal to the number of individuals that have combination c in the original data set, we set the sample size equal to that number multiplied by 10. Hence, for each combination of the variables number of relapses, CDP and index therapy we use step 3 to create a data set which is 10 times as large. Since the data set is synthetic, there are no new relationships in the larger data set. For this reason, we do not expect that the performance of the BNNs improves because of the additional data. However, it is possible that the BNNs are capable to learn from the additional data as the existing relationships are still present in the observations and more examples are given during the training. This could reduce the uncertainty within the model.

The average performance over 100 predictions of the BNNs trained on 30,000 observations are shown in table 11. When we examine the results, we observe that the performance of some of the models improves, while it deteriorates for others. Thus, the overall performance of the BNNs does not improve when we train the models on a larger data set. In addition, the plots in figure 23 show the variation of the performance of the BNNs. When we compare the variation of these models with the variation of the BNNs trained on the smaller data set, shown in figure 22, we notice a large difference. The variation of the BNNs trained on the larger data set is much smaller. This could imply that the uncertainty in the model reduces when more data is available. The comparable performance combined with the reduced variation indicates that BNNs can be a suitable solution in a scenario where flexible models are required but there is only a small data set available. In this scenario, the models can express the additional uncertainty.

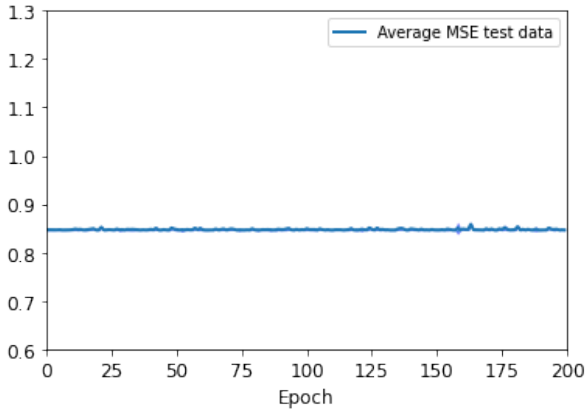
Table 11: Performance of BNNs with 3000 and 30,000 observations

	$N = 3000$		$N = 30,000$	
	AUPRC	MSE	AUPRC	MSE
Single BNN (H^2_{CDP})	0.349	-	0.337	-
Single BNN ($H^2_{relapse}$)	-	0.860	-	0.847
Combined BNN ($H^2_{com:CDP}$)	0.325	0.858	0.341	0.912
Combined BNN ($H^2_{com:rel}$)	0.302	0.855	0.325	0.914

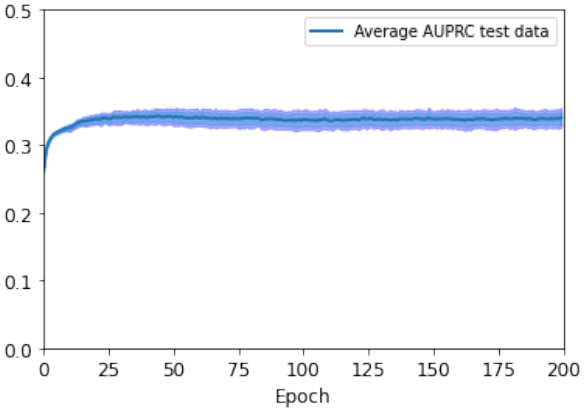
This table shows the performance of the single BNN and the combined BNN with the small and large data set. The performance is taken as the average performance of the 100 predictions in the last epoch.



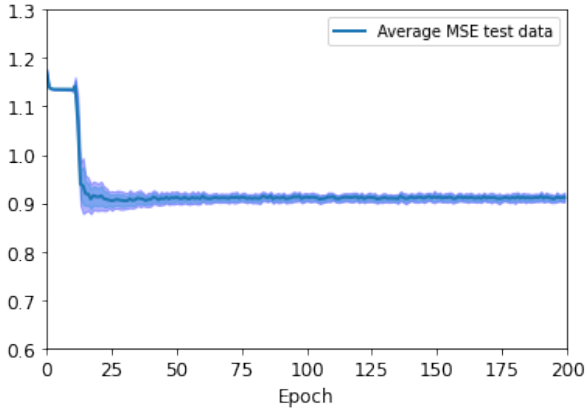
(a) AUPRC single CDP model with H^2_{CDP}



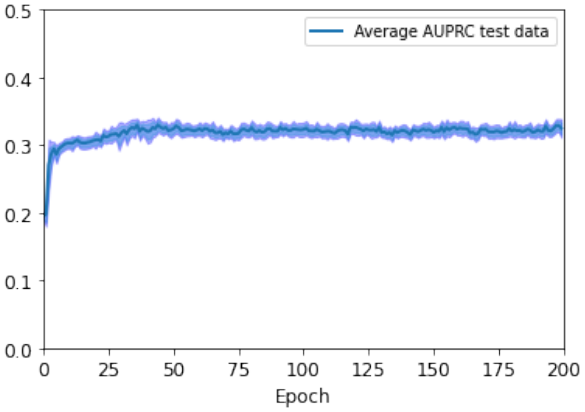
(b) MSE single relapse model with $H^2_{relapse}$



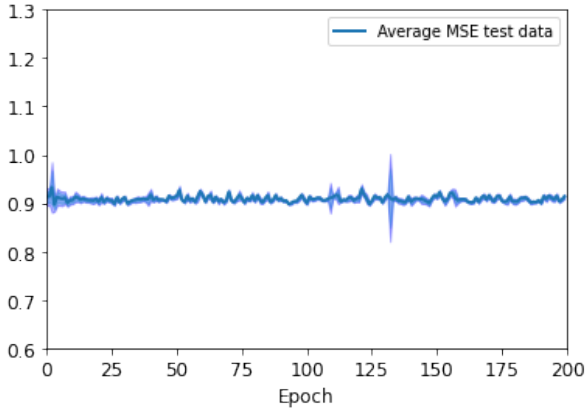
(c) AUPRC combined model with $H^2_{com:CDP}$



(d) MSE combined model with $H^2_{com:CDP}$



(e) AUPRC combined model with $H^2_{com:rel}$



(f) MSE combined model with $H^2_{com:rel}$

Figure 23: This figure shows the performance of the single model on the first row and the combined model on the second and third row for the data set with 30,000 observations. In each epoch, 100 different predictions are made. The graph shows the average performance, plus and minus 2 and 3 standard deviations.

I Sigmoid activation function

The predictions of the variable CDP are constructed by using two output neurons in the last layer of the network. The values in the output neurons are activated with the softmax function. This function is equal to,

$$f(z)_i^{softmax} = \frac{e^{z_i}}{\sum_{j=1}^{n^L} e^{z_j}} \tag{37}$$

where n^L is the number of output neurons and z_j is the pre-activation value. Note that for $n^L = 2$, the probabilities of belonging to the two groups in the above formula are similar to applying the sigmoid activation function to estimate the probability of belonging to one of the groups.

$$f(z)^{sigmoid} = \frac{e^z}{e^z + 1} \tag{38}$$

When the sigmoid activation function is used there is only one output neuron in the last layer, instead of two for the application with the softmax activation function. Hence, the BNN with the sigmoid activation consists of less parameters, as the second output neuron in the softmax activation is connected to each neuron in the previous layer and is as well connected to an extra bias neuron. For this reason, the complexity of the BNN increases in the application with two output neurons. We examine the performance of the BNN when the sigmoid activation function is used and the CDP is predicted with the use of one output neuron. The average performance over 100 predictions for the CDP are shown in table 12. Although the formulas to compute the

Table 12: Performance of BNNs with 2 versus 1 output neurons

	Softmax	Sigmoid
Single BNN (H^2_{CDP})	0.349	0.178
Combined BNN ($H^2_{com:CDP}$)	0.325	0.175

This table shows the AUPRC of the single BNN and the combined BNN for two different output layers. When the softmax activation function is employed in the last layer, the BNN has two output neurons. On the contrary, when the sigmoid activation function is used, the BNN has one output neurons. The performance is taken as the average performance of the 100 predictions in the last epoch.

probability of belonging to each of the CDP classes are similar, the difference in performance is large. It is possible that the additional complexity caused by the softmax function actually makes the model more flexible regarding the prediction of the less observed value for CDP, that is, a CDP of 1. Since the network has two output neurons that each have weights connecting them to the hidden layers, this might introduce more flexibility in the last layer.

J Early stopping

When we examine the plots in figure 22, it can be noticed that the AUPRC reaches a maximum in a particular epoch after which it slightly decreases. For this reason, it might improve the performance of the BNN if we use early stopping, which aims to decide the optimal number of epochs based

on the change in the performance (Nielsen, 2015). As the performance of a BNN can fluctuate over the epochs and it is possible that the performance deteriorates before it finally improves, it is general practice to stop the training of the network if the performance did not improve after a couple of epochs. We implement the early stopping algorithm by stopping the training of the network if the performance did not improve for 10 epochs, which we refer to as $C = 10$. Since the combined models form predictions for two different variables, we use the early stopping criteria on both performance measures. In other words, if the early stopping criteria stops the training of the model after 100 epochs based on the AUPRC and it stops the training after 120 epochs based on the MSE, we stop the training after 120 epochs. Note that this solution is not optimal, as it is possible that the AUPRC deteriorates after the 100^{th} epoch. On the other hand, stopping the training at the 100^{th} epoch can imply that the MSE is not optimal yet. This means that although early stopping can improve the prediction performance for one of the variables, it can deteriorate the performance for the other variable. We use 80% of the original training data to train the BNN and use 20% of the original training data as validation data set. In this manner, the test data is not used to decide the number of epochs. After we find the number of epochs after which we should stop training, we train the BNN on the full training data with that particular number of epochs and retrieve the final performance. Table 13 shows the performance of the BNNs without early

Table 13: Performance of BNNs with and without early stopping

	No early stopping		$C = 10$	
	AUPRC	MSE	AUPRC	MSE
Single BNN (H^2_{CDP})	0.349	-	0.356	-
Single BNN ($H^2_{relapse}$)	-	0.860	-	0.858
Combined BNN ($H^2_{com:CDP}$)	0.325	0.858	0.345	1.053
Combined BNN ($H^2_{com:rel}$)	0.302	0.855	0.325	0.857

This table shows the performance of the single BNN and the combined BNN with and without early stopping. The performance is taken as the average performance of the 100 predictions in the last epoch.

stopping, that is, the results of section 5.4, and the performance with early stopping. The use of early stopping improves the performance of the single models. On the contrary, the performance of the combined models improves with respect to the AUPRC, yet it worsens with respect to the MSE. To conclude, early stopping can improve the performance of one of the variables in the combined model, while it can damage the other. For this reason, we cannot conclude that the performance of the combined model improves due to the implementation of early stopping.