Deep Neural Networks (DNNs) are nowadays the most popular approach used in most applications of AI. They are structures composed of many complex processing layers (hence "deep"), each of which is composed of multiple processing units, so-called neurons (hence "neural"). DNNs are used for making predictions, classifying objects, controlling robots, and many more tasks. They can be viewed as processing machines that when served an input produce relevant outputs. These inputs can be some features, for example, an image of a cat, and outputs can be labels, for example, "cat". The predictions standard DNNs produce are, however, far from perfect and they can be catastrophically incorrect. Being incorrect itself is not such a major issue, but what is important, DNNs do not know when they are incorrect. This is because DNNs do not manage uncertainty well - they are overly confident about their predictions, for example, in the case when they are shown an input that was not previously (during training of such a network) seen.

A more reliable counterpart of DNNs is Bayesian Neural Networks (BNNs). BNNs are DNNs in the sense that they follow the same design pattern: consist of multiple complex processing layers. On the other hand, they learn and perform predictions differently, and what is the most important they know when they do not know. This is thanks to the Bayesian inference framework. All information (including predictions) in the Bayesian framework is inherently encoded in a way including information about uncertainty, by representing it with distributions over possible values. It applies also to BNN's parameters: when training a BNN we learn their values only up to some level of certainty.

In Bayesian learning, model before seeing any data is already assumed to encode certain *a priori* knowledge. In particular, by *priors* we mean distributions over a model's (e.g. BNN's) parameters (sometimes also over model structure) before actual training or inference is performed. Then, when data is presented to the model it can update these distributions accordingly to the so-called *posteriors*. For example, before seeing ever a cat we can know it has four legs and a tail, but after seeing some pictures a cat detection model can update itself in a way that it would also check for fur and pointy ears. Nevertheless, if the original beliefs were completely wrong, it would take a long time to adjust them appropriately. Additionally, if they were also too strong it may be entirely impossible.

As illustrated above, setting the priors right is an important part of the model building process. However, for complex models such as BNNs, it is also a nontrivial task, since we lack the intuition about how a particular parameter's value translates to beliefs expressed by model outputs. Furthermore, the previous evidence shows that setting them naively may have significant consequences on performance of the model. Another problem with BNNs is that finding the posteriors is also a challenging task. This is due to the large number of parameters these models have as well as due to the large size of the data used for training them (=finding posteriors). In practice, posteriors are found only in an approximate way, which often results in suboptimal performance and additionally complicates the understanding of priors' impact. These limitations taken altogether cause BNNs to perform far below the expectations and the standard DNNs are still used more often in practice.

Our goal is to address some of the above challenges and make BNNs more competitive and release some of their potential. We hypothesize that to achieve better performance for various tasks, network structure, learning approach and priors need to be decided jointly. In particular, priors need to be given more attention and we plan to investigate how better priors can improve the performance of BNNs. Our goal is to make these priors smarter so they would adapt themselves to a given task. This implies two questions: First, how to create such smart and flexible enough priors. Second, how to make them learn with the approximate methods, in a way that BNNs would be competitive in terms of both training time and effectiveness.

To sum up our objectives in this project, we will start by investigating how priors learning can improve the performance of BNNs and when it is optimal to learn priors. We will begin with the identification of selection and learning methods for priors in Bayesian neural networks optimal for various objectives. Furthermore, we will search for optimal architectures of flexible priors and posteriors for parameters of BNNs. We plan to look into structured, hierarchical, and heterogeneous priors. Finally, from a theoretical point of view, we will study the correspondence of the BNNs and other Bayesian models and in particular, how BNNs relate to so-called Gaussian Processes.

We will evaluate our approaches on the standard benchmarks for several interesting and important settings. We aim at improving effectiveness against state-of-the-art approaches for uncertainty quantification.