Bayesian Neural Networks

Blair, Taylor Sorgman, Ava Bekaert, Conor

May 6, 2024

Abstract

Bayesian Neural Networks are...

Contents

1 Machine Learning

Machine learning includes a variety of topics of which artifical intelligence (AI) and neural networks are a subset. Neural networks in particular involve an abstract imitation of the human brain using simulated neurons which is trained on data using particular algorithms.

The term supervised learning refers to a specific common method of machine learning that uses a labeled dataset to train a model to correctly predict the labels via some training algorithm. A classification problem in supervised learning involves predictions where all labels are grouped into a set of categories. The training process can be broken down into three major steps, which include a decision process, an error function, and an optimization process [7]. The decision process is the set of steps the algorithm takes after receiving the data based on the goal of the model. The second step in the process is the error function, which is the method of measuring chosen to see if the algorithm gave a “good" or “bad" input. The two most common choices of the error function are a simple yes or no on whether a data point was classified correctly in the case of classification, or the difference in value between the predicted outcome and the actual observed outcome for continuous values. Finally, the third and last step is the part of the process that implements learning. This step, the updating or optimization process, requires the algorithm to review the past data and outputs of the error function in order to better correct its decision-making process in the future.

Throughout this report, we will often use the terms machine learning, deep learning, and neural networks. It is important to note that although these are fundamentally related fields, deep learning is a subfield of neural networks that in particular focuses on more complicated neural networks, while neural networks are a subfield in machine learning.

We will be training a classifier on labeled images in a supervised learning process to predict what is depicted in the image from a range of possibilities such as dog, cat, and plane. These images come from the CIFAR-10 dataset, which will be discussed more in detail later in this paper.

1.1 Neural Network

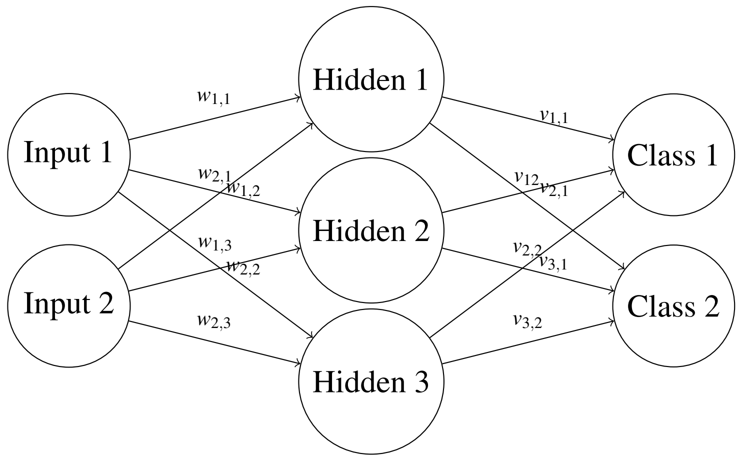


Figure 1: Example neural network

There are many effective introductions to neural networks to any level of detail available elsewhere, so we will present a straightforward summary of the concepts required for understanding the work of this paper.

Many models for analyzing data involve making an assumption about the functional form of the outputs given the inputs or predictors. Neural networks are no exception - at the most basic conceptual level, a neural network is just a really complicated function that has the potential to represent many patterns and dimensions through its complexity. Their name comes from the inspiration they take from the human brain, the idea that enough neurons with enough connections between them can encode everything needed for effective predictions. Indeed, the fundamental unit of a neural network is a neuron with its associated connections.

Neural networks are typically organized in a series of layers of many neurons, where each individual neuron of one layer is connected to every neuron of the previous layer. Data flows from an initial input layer, through hidden layers, then into the outputs as visualized in Figure [**fig:example-nn**]. The connections are assigned certain weights, which measure the influence of a neuron on the next layer. This structure ultimately yields a complicated yet well-defined function because every step is a clearly-defined arithmetic operation, and they are all composed into a single whole. Neural networks can have different architectures depending on the organization and types of layers involved.

1.2 Neural Network Neurons

As shown in Figure 2, any single neuron is connected to the neurons $x_1, x_2, ..., x_n$ from the previous layer. The outputs of the previous neurons become the inputs to the new neuron by means of connections with weights $w_1, w_2, ..., w_n$ associated with them (each previous neuron has a certain weight). The neuron itself first takes a value calculated as a weighted sum $\sum_i w_i x_i$ of the inputs x_i to it, then typically an additive bias is included. Lastly, the value is passed through an activation function which restricts the value to a certain range, typically $[0, 1]$.

The weights and the biases of a neural network are the parameters, and they are the values that are tuned in the training process. In a sense, they represent the "memory" of the network, whatever it has learned from training. For any particular network architecture (meaning the choice of types of layers and their order), once the model has been trained with a particular architecture, the parameters are all you need to run that network on new data to make predictions.

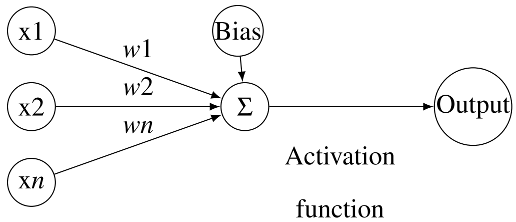


Figure 2: Neural network neuron

1.3 Convolutional Neural Networks

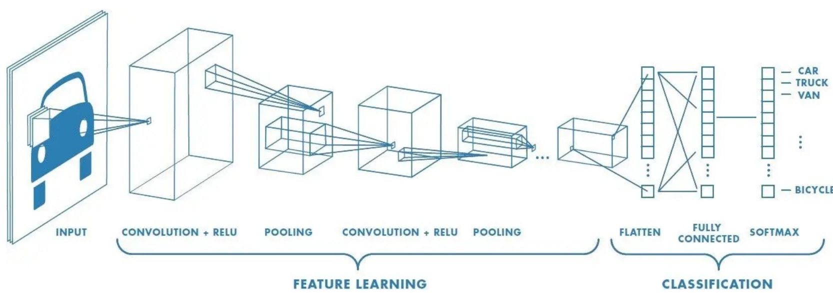


Figure 3: CNN pipeline [4]

Convolutional neural networks (CNN) are a type of neural network that is better suited for image recognition. While this might sound like a separate model structure, CNNs are largely the same. The primary difference from a traditional neural network is the convolutional layer, as highlighted in Figure [**fig:cnn-pipeline**]. Instead of reading the entire image at once, a convolutional layer slides a small window over the image, shrinking that window even further by the application of a convolution operation. Once the window has slid over the entire image, a downsized image is formed that may encode useful information more densely. Because of this downsizing action, CNNs typically require fewer parameters relative to the traditional neural networks described previously, which improves training costs and the capabilities of the model.

2 Bayesian Neural Networks

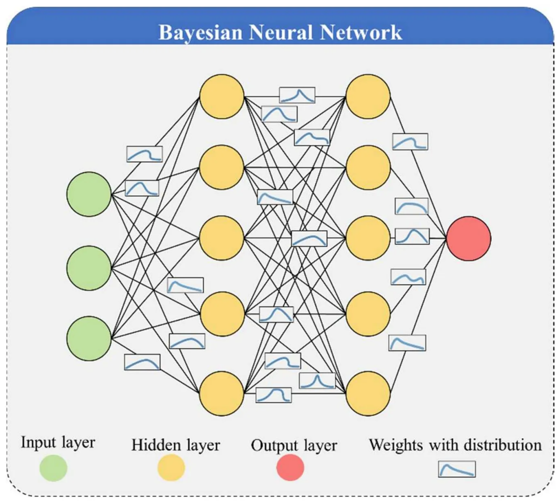


Figure 4: Example BNN [1]

Bayesian neural networks (BNN) take the same principle as your typical neural network, but factor in a measure of uncertainty into the prediction of the network. This uncertainty, of course, comes from the Bayesian theory of statistics, which believes that all events have some level of randomness, and thus must have uncertainty in their prediction.

One way this uncertainty can be built into the model is through a prior probability distribution. Uncertainty estimates of the model can be obtained from the parameters of the BNN in their probability distributions [6]. The average estimate of the parameters is computed through multiple different models during the training process, which can regularize the network. However, it can be very difficult to calculate the model posterior of a BNN. Approximations of the posterior of the model are most often used, but those are often still very computationally intensive. Different methods of approximating the model's posterior is a current research topic within the field.

The two main types of uncertainty within a BNN are aleatoric uncertainty and epistemic uncertainty. Aleatoric uncertainty is a measure of the uncertainty in the observations, which is uncertainty that is inherent in any form of data. This type of uncertainty is modeled by placing a prior distribution over the output of the BNN model. In contrast, epistemic uncertainty is a measure of the uncertainty caused by the model and its predictions. It is modeled by a prior distribution over the model's weights, and an analysis of how much the weights of the model change when the data given to the model changes [6].

But what is so great about calculating uncertainty in our model? Well, it's that our predictions become more uncertain. One of the major issues with neural networks as they are is they are highly prone to overfitting the data. Although they are great at predictions on the training data they are given, they often struggle with predictions on the testing data they are given due to the issue of overfitting. Many methods have been developed in order to regularize neural networks to work to prevent this from occurring, such as weight control measures. Using the Bayesian framework of uncertainty and building it into a model serves as a built-in prevention of overfitting.

A Bayesian neural network introduces priors on the parameters θ of a neural network and uses the data to adjust these priors to reach posterior distributions for these parameters. Recall that a (trained) neural network is really just determined by its parameters - the weights and biases. This gives rise to the conceptual interpretation that a Bayesian neural network is an infinite ensemble of possible neural networks which can be sampled to obtain any number of individual networks.

Constructing a Bayesian neural network involves selecting a model architecture and choosing the priors, which are also known as the "stochastic model" in this context. The definition of Bayesian neural networks provides flexibility, since the primary difference is that neuron weights get distributions instead of individual values. As a result, most model architectures that work for a typical neural network also work for a Bayesian version of that network. Lastly, the training of a Bayesian neural network uses techniques such as MCMC for inference.

2.1 Bayesian Neural Network Neuron

A visualization of a Bayesian neuron appears in Figure 5. It is in fact quite similar to Figure 2, with the essential difference that the individual values from before have been replaced with distributions. Within a neuron, the distributions from input neurons are weighted by the weight distributions and combined together along with the bias distribution. After passing through the activation function, the neuron output is produced.

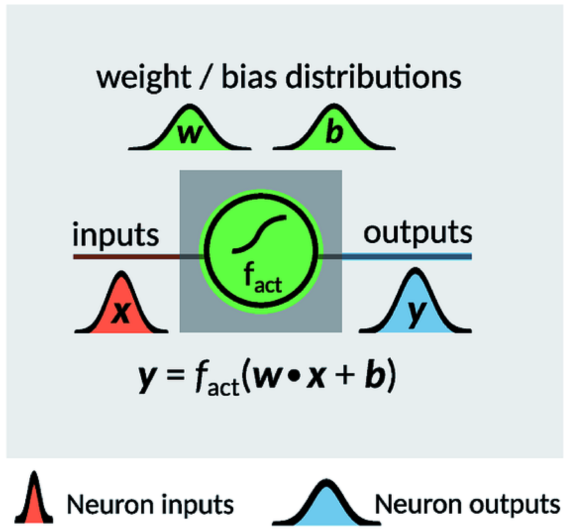


Figure 5: Example BNN Neuron [2]

2.1.1 Priors

The Bayesian paradigm requires a prior on the parameters of a neural network to facilitate training. However, making choice of a prior on all the parameters of a network, following anything other than a simple concept, is difficult, especially given the limited understanding of how priors at one layer affects the next in general. For the purposes of this work, only one prior-selection scheme will be considered, since considering many methods would require introducing many further new concepts.

A choice of prior that is both commonly used and intuitive is an independent normal distribution on each weight. In particular, a $\mathcal{N}(0,\sigma^2)$ distribution with zero mean and some standard deviation σ for each particular weight typically works. This can be thought of as a reasonable choice of vague prior. Since further interpretation of how these priors influence the model becomes even more complicated for other priors, selecting a more informative prior would be tricky. For an example of this complexity, Vladimirova et al. contend that for a network with these priors whose activation functions satisfying a certain "extended envelope property", the priors on the k^{th} hidden layer become sub-Weibull-distributed with parameter $k/2$ conditional on the data passing through the network. Such interpretation is valuable, but beyond the scope of this paper.

However, it turns out there is suggestive parallel between a prior in a Bayesian neural network and regularization in a typical neural network. Regularization is an additional term in the loss that penalizes large weights. Intuitively, this helps resist overfitting by preventing a single neuron from overly influencing the model, since such a neuron may be overemphasizing a fluke of the training set rather than a general pattern. The regularization term typically takes the form $\alpha||w||_p$ where $||\cdot||_p$ is some choice of metric on R^n, typically $p = 1$ (Manhattan/taxicab distance) or $p = 2$ (Euclidean distance), and α is the regularization hyperparameter that controls how strongly large weights are penalized.

As in Jospin et al., consider network parameters θ with a loss function L assumed to be the negative log likelihood loss, where D_y represents the true labels for the training data D_x. Suppose the network does not use regularization. Maximum likelihood estimation aims to find the choice of θ which minimizes the loss on the given data:

$$\hat{\theta} = \arg\min_{\theta} \left\{ L_{D_x,D_y}(\theta) \right\}.$$

From the Bayesian perspective, since the negative log likelihood loss is used, this means that

$$\hat{\theta} = \arg\min_{\theta} \left\{ -\log P(D_y|D_x, \theta) \right\} = \arg\max_{\theta} \left\{ P(D_y|D_x, \theta) \right\}.$$

This formula indicates to find the choice of θ that maximizes the probability of the true labels D_y for the data given the observed D_x values for parameters θ. Now, suppose a prior $p(\theta)$ is introduced:

$$\hat{\theta} = \arg\max_{\theta} P(D_y|D_x, \theta) p(\theta).$$

When converted back to the frequentist paradigm under log-likelihood loss, the multiplicative factor for the prior becomes an additive regularization term:

$$\hat{\theta} = \arg\min_{\theta} L_{D_x,D_y}(\theta) + reg(\theta), \quad reg(\theta) = -\log(p(\theta)).$$

Jospin et al. state that normal prior selected above is "equivalent to a weighted ℓ_2 regularization (with weights $1/\sigma$) when training a point estimate network." In this sense, the Bayesian model which utilizes priors specified in advance helps prevent overfitting simply by including priors in the first place.

2.2 Bayesian Convolutional Neural Networks

As discussed previously, Bayesian neural networks work with most functional models that work for typical neural networks. Convolutional neural networks are no exception - by simply replacing typical CNNs with Bayesian counterparts containing Bayesian neurons and the appropriate operations, a Bayesian convolutional neural network (BCNN) is created.

3 Simulation

We use a BCNN implementation from Github based on work from ... [6] [5]

3.1 CIFAR-10

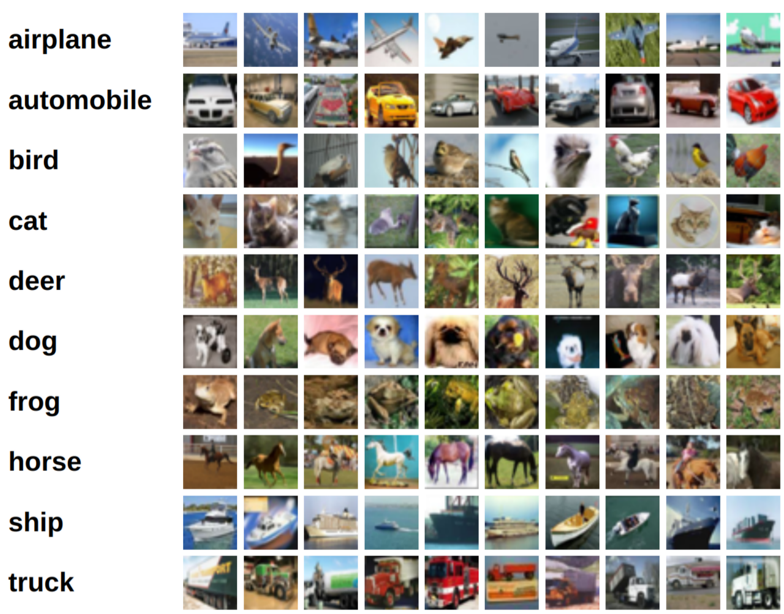


Figure 6: Example CIFAR-10 images [3]

The CIFAR-10 (Canadian Institute For Advanced Research 10) dataset is a machine learning benchmarking set. It contains $60,000$ 32×32 RGB pictures of airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks [3]. The ten classes within the dataset are mutually exclusive, meaning there is no image containing multiple of the objects mentioned above. The number of images is split evenly between each class, with $6,000$ images per each type of object. It is a very popular data set (in fact the most common) to use in machine learning and training neural networks, due to the extensive size, as well as the decent number of categories. The purpose of the images within the dataset being of a lower resolution is to allow researchers to see more quickly (and with less computation effort) how well an algorithm works on classification with the dataset. Convolutional neural networks are often seen to be the “best" at classifying CIFAR-10.

3.2 Hyperparamaters

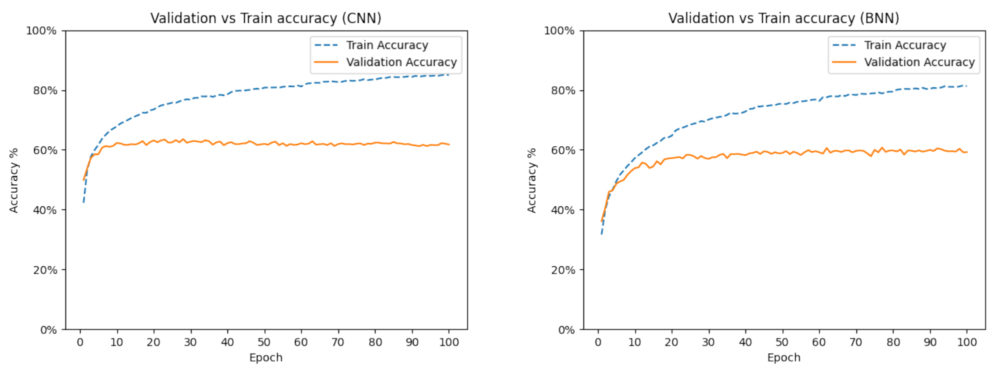
We choose to keep the number of layers and levels in the model the same to keep the models roughly equal, but tuned the hyperparamaters seperately. We did not adjust the priors on the BCNN, as those were set by the BCNN layer code from [5]. We found the following hyperparemeters to created the best model:

Hyperparameter	**CNN**	**BCNN**
Epochs	100	100
Learning Rate	0.001	0.003
Regularization Rate	0.001	0.001
Optimizer	Adamw	Adamw

3.3 Results

Metric	CNN	BCNN
Train Accuracy	84.96%	81.27%
Validation Accuracy	61.76%	59.21%
Time to Train	16 min 11 sec	22 min 11 sec

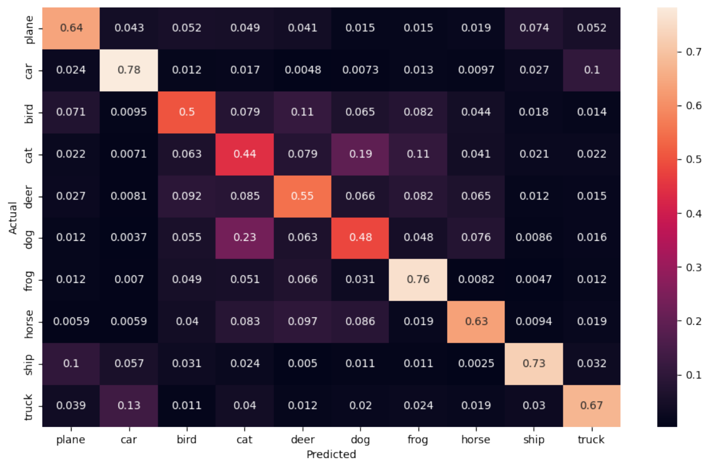
The results we achieved with training were about the same between the BCNN and the CNN. However, the bayesian model took longer to train.



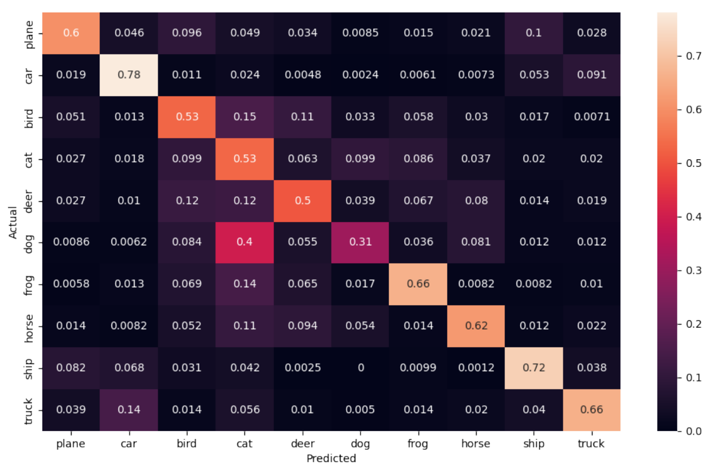
(a) CNN (b) BCNN

Figure 7: Train vs validation accuracy over time

Based on the divergent lines in figure 7, the model needs tuning. Specifically, the regularization rate needs to be raised. One interesting thing that appears in the figures, is that the BCNN took longer to diverge from the train accuracy compared to CNN.



(a) CNN



(b) BCNN

Figure 8: Confusion matrices

Based on the confusion matrix from two models in 8, the two models confuse similar sets of classes. Both models struggle to differentiate animals, but the BCNN has poorer accuracy on the classes that are easily confused.

References

Fleszar, J. (2023). Bayesian neural networks - capturing the uncertainty of the real world !!

Häse, F., Galván, I. F., Aspuru-Guzik, A., Lindh, R., & Vacher, M. (2019). How machine learning can assist the interpretation of ab initio molecular dynamics simulations and conceptual understanding of chemistry. *Chemical science*, *10*(8), 2298–2307.

Krizhevsky, A., Nair, V., & Hinton, G. (n.d.). Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/~kriz/cifar.html

Saha, S. (2018). A guide to convolutional neural networks — the eli5 way.

Shridhar, K., Laumann, F., & Liwicki, M. (2018). Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. *arXiv preprint arXiv:1806.05978*.

Shridhar, K., Laumann, F., & Liwicki, M. (2019). A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*.

What is machine learning (ml)? (2020).