Using Artificial Neural Networks to Approximate Bayesian Inference

Heejae Jang [1] **Aaron Lanz** [1] **Xinlei Lin** [1]

Abstract

Despite noisy and/or sparse evidence, humans are still able to generate reasonable inferences to solve inductive problems. In this paper, we train artificial neural networks (ANNs) on a task that requires inductive reasoning. We find that ANNs can perform these tasks using Bayesian-like strategies without the need for an explicit computation of the log likelihood. Our results provide a computational approach for approximating Bayesian inference that is inspired and consistent with the function of neural circuits in the brain.

1. Introduction

Humans and other animals often behave near optimally when faced with uncertainty (Battaglia et al., 2003; Ernst & Banks, 2002; Hillis et al., 2004; Körding et al., 2007; Merfeld et al., 1999; Wolpert et al., 1995). This uncertainty can come in the form of sensory or internal noise that blurs our interpretation of information (low level uncertainty) (Ghahramani, 2015; Ma et al., 2006; Orhan & Ma, 2017) or can emerge in the form of incomplete sensory input that ambiguates our models of the world (high level uncertainty) (Ghahramani, 2015). Although we have a solid understanding of the mathematical and computational strategies required to solve or approximate Bayesian inference tasks, we know much less about the strategies used by neural circuits to perform optimal inference.

ANNs are an ideal tool to study biologically plausible mechanisms of human learning and behavior. ANNs are a subset of machine learning tools that are inspired by the computational principles used by the nervous system (Zador, 2019). In particular, ANNs are composed of layers of interconnected nodes (synaptically coupled neurons) that propagate non-linear activity (action potentials and synaptic transmission) throughout the network to accomplish specific goals. Even the way ANNs are trained have potential neurological correlates (Payeur et al., 2020). ANNs have been successfully used to model numerous types of human behaviour, including semantic categorization (Rogers, 2003), syntax recognition (McClelland et al., 2013; Elman, 1990), motor control (McClelland et al., 2013), and perception (McClelland et al., 2013; Lecun et al., 2015). When trained on psychophysical tasks using noisy sensory data, ANNs do indeed perform optimally (Orhan & Ma, 2017), suggesting that neurally-inspired computational models can approximate Bayesian inference when faced with low level uncertainty. Whether the same types of models can perform Bayesian inference on tasks with high level uncertainty remains unclear.

Here, we train fully-connected ANNs on a version of the number game – a task involving decision making using incomplete data (Tenenbaum, 2000) – and then analyze representations in the output layer to determine if Bayesian-like strategies emerge in the network during training. We show that networks trained on the number game indeed display strategies consistent with Bayesian inference. These results emerge in relatively simple neural networks (2 layers, 16 nodes each) and improve as the networks increase in complexity. Furthermore, we show that the inference performed by neural networks shifts if the priors are modified, suggesting ANNs are a flexible strategy to approximate Bayesian inference in tasks with ambiguous and incomplete information.

2. Methods

2.1. Task: number game

The number game is a type of inductive problem first introduced in (Tenenbaum, 2000). In this game, a computer program generates positive numbers ranging from 1 to 100. The program presents the player with a small subset of numbers and the player's goal is to to evaluate which numbers will also be accepted by this program. Essentially, the task requires the player to induce what rule the program is using to generate the numbers based on sparse evidence.

2.2. ANN architecture

General structure Our ANN consists of an input layer fully connected to a number of hidden layers via a rectified linear unit (ReLu) activation function, which is then connected to the ouptut layer via a sigmoid activation function. The input layer is a one-hot encoding representation of two sets of inputs, one containing a set of observations, and the other containing the number to be evaluated. The

output layer is a single number; 0 if the number to be evaluated is not accepted by the program and 1 otherwise.

Training dataset For all training and testing, the numbers are positive numbers ranging from 1 to 100. The set of observations in the input layer is generated by sampling a hypothesis (uniform random sampling from the cumulative distribution function of prior probabilities) and randomly selecting 5 numbers at most from that hypothesis. Our hypothesis space consists of two types, mathematical and interval. Examples of mathematical hypotheses are odd/even numbers, square numbers, multiples of *n*, primes, numbers ending with a particular digit, and so forth. Interval hypotheses are consecutive numbers such as $X = [12, 13, 14, 15, 16, 17]$. These are manually defined, resulting in the entire hypothesis space to consist of 34 mathematical hypotheses and 5050 interval hypotheses. Each hypothesis is weighted with different prior probabilities through a free parameter λ. λ is attributed to mathematical hypotheses and each mathematical hypothesis is weighted with equal prior probabilities. 1-λ is attributed to interval hypotheses each of which is weighted differently using an Erlang distribution with σ = 10. The second pool in the input layer, the number to be evaluated, is sampled from the same hypothesis with 50% chance. The target output label is 1 if the number is sampled from the same hypothesis and 0 otherwise.

Optimization The ANN is optimized through a stochastic gradient descent of either mean-squared-error (MSE) or binary cross-entropy (BCE) as loss functions which we will compare in the results section.

Parameters The parameters to be explored are the prior probabilities λ, number of hidden layers, number of nodes in each hidden layer, shape of the network (e.g., constant or variable nodes in each layer), and type of loss function used.

2.3. Bayesian inferences

Probability theory, and in particular Bayes' rule, is thought to provide a "logic of uncertainty". Specifically, we may wish to learn a model of the relationship between pairs of variables X and Y. Mathematically, this is expressed as a conditional probability $P(Y|X)$.

A full Bayesian inference makes predictions by averaging over all likely hypotheses under a posterior distribution. For a discrete variable $X = \{x_1, ..., x_n\}$ belonging to a category C, the predictive distribution of new observations $Y = \{y_1, ..., y_n\}$ will be the product of posterior and likelihood, marginalized over all possible hypotheses:

$$P(y \in C \mid X) = \sum_{h \in H} P(y \in C \mid h) P(h|X) \qquad (1)$$

Here the posterior in (1) is computed as

$$P(h|X) = \frac{P(X|h)P(h)}{\sum_{h' \in H} P(X|h')P(h')} \qquad (2)$$

where we assume that all hypotheses are equally likely and we sample uniformly from the hypothesis space. Under the assumption that the samples from X are drawn i.i.d., the likelihood decomposes as a product of individual probabilities:

$$P(X|h) = \prod_{i=1}^{n} P(x^{(i)}|h). \qquad (3)$$

In this project, instead of making Bayesian computations stated above explicitly, the goal is to see whether fully connected neural networks can take uncertainty into account and achieve similar Bayesian-like predictions implicitly. Note that the neural networks still optimize weights as point estimations rather than distributions.

3. Result

3.1. Number game Bayesian predictions

To determine if ANNs can perform Bayesian inference on tasks with incomplete or ambiguous information, we simulated the number game - a task where players must evaluate whether a number belongs to a set of observed numbers, which are itself drawn from a mathematical hypothesis (e.g. all numbers end with 7) or interval hypothesis (e.g. all numbers are between 20 and 30). Since a set of observed numbers can often belong to numerous interval or mathematical hypotheses, this task contains ambiguous information that can be solved optimally using Bayesian inference. We began by calculating the theoretical probability of membership (according to Bayesian theory) for numbers 1-100 for several sets of observed numbers. As shown in Figure 1, when the observed numbers unambiguously point towards a certain hypothesis (e.g. $X = [16, 8, 2, 64]$ = mathematical; or $X = [16, 23, 19, 20]$ = interval), the Bayesian predictions reflect the appropriate hypothesis with high certainty. However, when the observed numbers are ambiguous (e.g. $X = [16]$), the Bayesian predictions reflect a mixture of mathematical and interval hypotheses. Thus, the number game is an excellent task to study Bayesian inference when faced with incomplete or ambiguous data.

The goals of the next sections are to train ANNs on the number game and see if the Bayesian predictions observed in Figure 1 emerge within the networks, and if they do, determine what types of architectures lead to optimal approximations. To train ANNs on the number game, we presented ANNs with pairs of observed numbers and unknown numbers (numbers to be evaluated), and trained the networks to predict whether the unknown number belonged with the

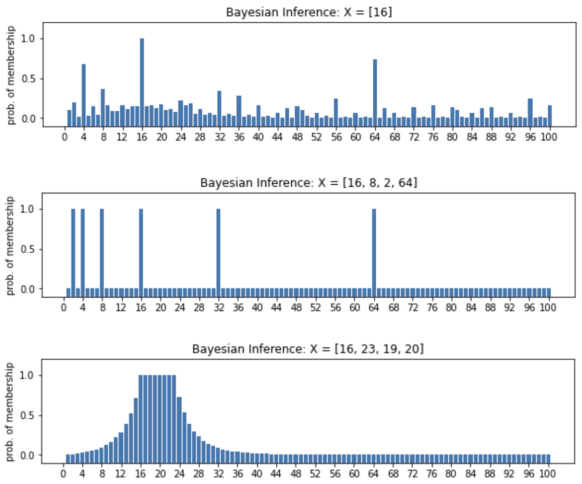

Figure 1. Probability of membership for numbers 1-100 as the observed number sets are varied between ambiguous numbers, mathematical numbers, and interval numbers. All calculations are performed using $\lambda = 2/3$.

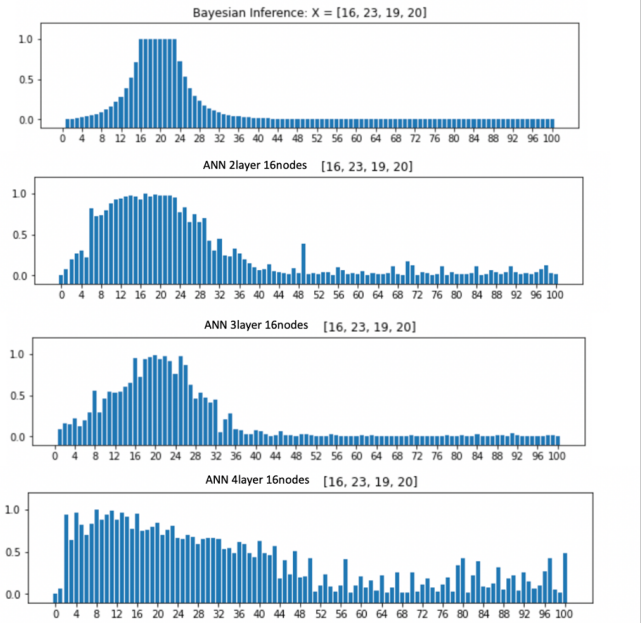

Figure 2. Performance comparison for numbers generated from an interval hypothesis. Panel 1: predictions made by Bayesian inference. Panel 2-4: predictions made by ANNs with increasing number of layers from 2 to 4 and fix number of nodes (16 nodes).

observed numbers. We constructed ANNs with two sets of inputs, one set containing the observed numbers and the other set containing the number to be evaluated; a variable number of hidden layers / nodes; and a single output node. After 1,000,000 training pairs, we froze the network weights and asked the network to predict whether the numbers 1-100 belonged with the observed number sets shown in Figure 1.

3.2. Vertical complexity

We compared ANN predictions with different number of layers. Figure 2 shows examples of ANNs with increasing number of layers and fixed node number (16). ANN with layer=2 and node=16 per layer achieves reasonable predictions compared to Bayesian inference (the goal), and ANN with layer=3 and node=16 per layer achieves better performance with its predictions more similar to the ones made by Bayesian inference. However, when we increase layer number to 4, the predictions become worse. The network fails to converge with layer number beyond 4. This result holds with similar number of nodes tested, suggesting that the vertical complexity of ANNs can affect the performance. Specifically, higher layer number can improve performance, but the ANN performance no longer benefits from more layers once the optimal number of layers is passed.

3.3. Horizontal complexity

We next compared the performance of three-layer ANNs with varying number of nodes within the hidden layers. For this analysis, we focused on the predictions for the observed numbers X = [16, 23, 19, 20]. As shown in Figure 3, all ANNs, regardless of the number of nodes, correctly classify the observed numbers as an interval hypothesis. However, as the number of hidden nodes increases from 8 to 64, the width of the 'prediction' narrows. Thus, the output of ANNs converges towards the Bayesian predictions as the hidden layers increase in width.

3.4. Network shape

We compared the ANN predictions with constant (64-64-64), decreasing (100-60-32), and increasing (32-60-100) number of nodes with successive hidden layers with the total number of nodes across all hidden layers kept constant. As shown in Figure 4, when the observation is a single number X = [16], all three models predict even numbers with high probabilities. However, while the constant and the increasing models seem to exhibit interval predictions around X = [16] and high prediction overall for all even numbers, the decreasing model more closely resembles Bayesian predictions in that it has a higher selectivity and specificity for multiples of 4. The decreasing model also outperforms the other two when powers of 2 are given as an observation (Figure 5). The constant model, in particular, predicts other even numbers with non-zero probabilities but the decreasing model exclusively selects power of 2 only.

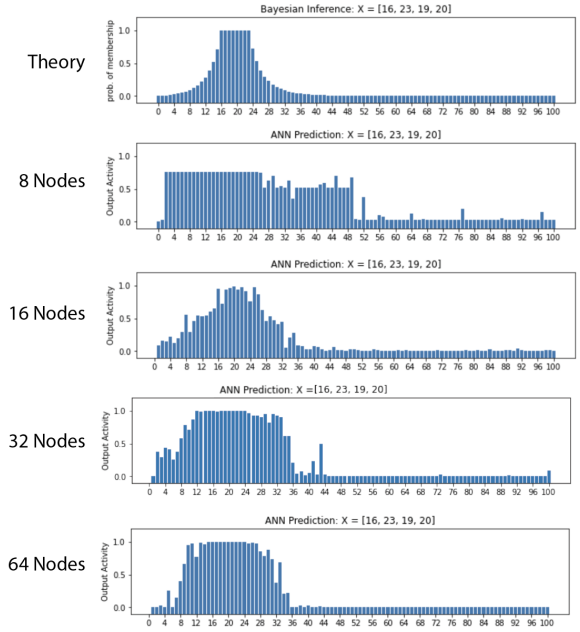

Figure 3. The ANN output converges towards the Bayesian prediction for X = [16, 23, 19, 20] as we increase the number of nodes in the hidden layers.

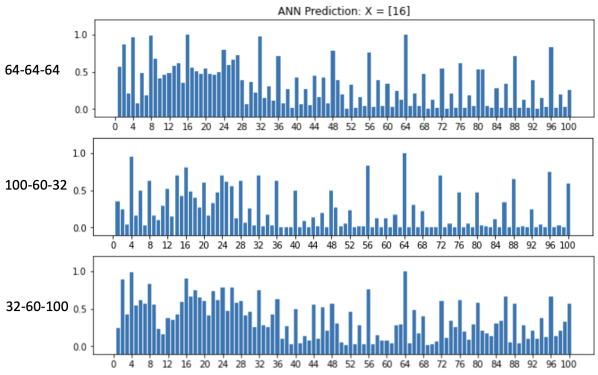

Figure 4. Despite sparse evidence consisting of a single number [16], the decreasing model closely resembles Bayesian predictions with higher selectivity and specificity for multiples of 4.

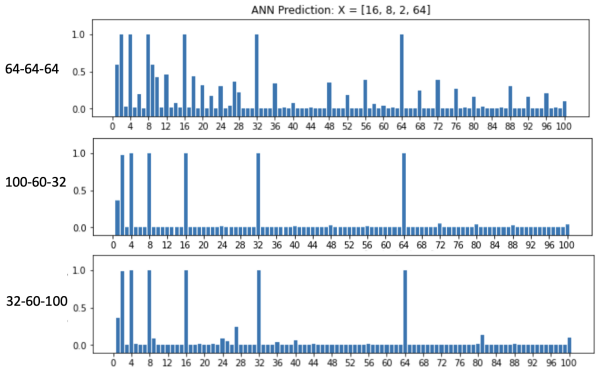

Figure 5. The decreasing model outperforms the other two models by exclusively selecting powers of 2.

3.5. Prior probability

For our best network, i.e., 3 hidden layers with decreasing number of nodes 100-60-32, we tested how predictions depend on the prior probabilities dictated by the free parameter λ. We predicted that a small λ value would yield interval predictions since interval hypotheses are more heavily weighted than mathematical hypotheses and the ANNs are trained more with examples drawn from interval hypotheses. Consistent with our prediction, as shown in Figure 6, whether the evidence is sparse (X = [16]) or consists of multiples of 2 (X = [16, 8, 2, 64]), λ=0.01 outputs interval predictions.

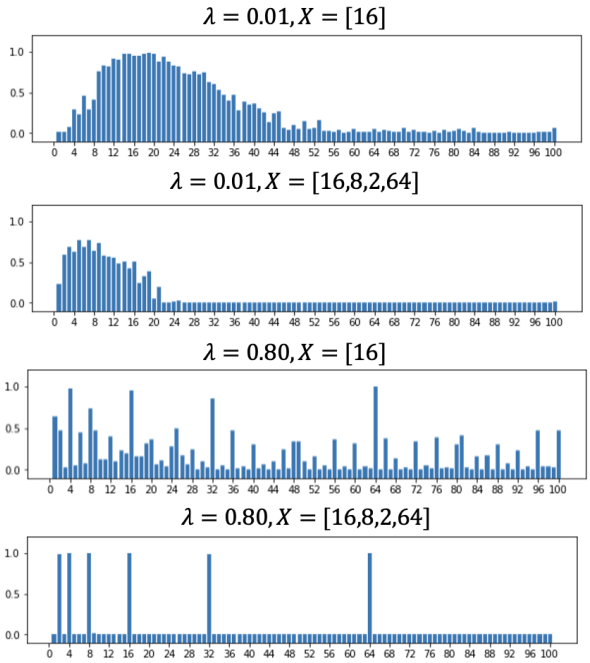

Figure 6. λ=0.01 yields interval predictions whereas λ=0.8 yields mathematical predictions.

3.6. Loss function

Loss functions define an objective against which the performance of the model is measured, and the parameters learned by the model is determined by minimizing the chosen loss function.

Binary Cross-Entropy Loss In the context of classification, the model's prediction $h(x_i)$ will be a value between 0 and 1 that can be interpreted as a probability of a new observation belonging to a class C after observing training examples x_i. If the probability were less than 0.5 we'd classify it as "not in class C". With the assumption that the data i.i.d, we have the following expression to obtain the cross entropy loss:

$$L(x, y) = \sum_{i=1}^{N} y_i \log(h(x_i)) + (1 - y_i) \log(1 - h(x_i)) \tag{4}$$

where y is the true label, which is either 1 or 0.

Mean-Squared Error Loss Given the assumption that the data follows normal distribution, the mean-squared error is given by calculating the squared difference from the true label and the output and summing across all training examples. MSE loss is often used in regression problems.

$$L(x, y) = \frac{1}{N} \sum_{i=1}^{N} (y_i - h(x_i))^2 \tag{5}$$

where y is the true label, which is either 1 or 0.

We found that the ANNs trained with MSE loss function converges to lower training error than those trained with BCE loss. As shown in Figure 7, in networks with 2 layers and 16 nodes per layer, the MSE case converges around 0.1 whereas the BCE case converges around 0.45. In networks with 4 layers and 64 nodes per layer, the MSE case achieves lower error (converging aorund 0.08), but the training error in the BCE case fails to decrease.

Besides the training error, the performance of networks trained with MSE loss is significantly better than those trained with BCE loss. Figure 8 shows performance of networks trained with different loss functions for interval hypotheses. Networks trained with MSE loss makes better predictions as the probabilities for numbers in the corresponding intervals are higher.

Our rationale for using a BCE loss function is that it is commonly used in classification tasks, which is consistent with our task - classifying whether or not the numbers to be evaluated belong to the same category as the observed numbers. However, a key assumption of BCE loss is that the data are generated from Bernoulli distributions and therefore can be classified into two categories. In Bayesian terms, this means we assume a binomial prior if we choose a BCE loss function, and a Gaussian prior if we choose a MSE loss function. In reality, the numbers are not binomially distributed. With an extensive training dataset and hypothesis space, it is likely that the true distribution of the data is closer to Gaussian, according to the central limit theorem. Therefore, ANNs trained with a MSE loss function achieves better results in the number game.

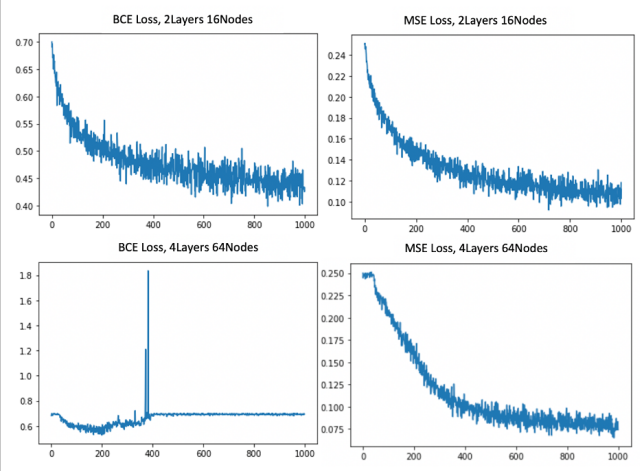

Figure 7. Training error comparison between BCE loss and MSE loss functions in networks with 2 layers, 16 nodes per layer and networks with 4 layers, 64 nodes per layer (all other parameters fixed). Errors are binned every 1000 epochs

3.7. Optimized ANN

While not exhaustive, the above analysis revealed that the optimal ANN architecture within our parameter space was the following: 3 hidden layers with decreasing nodes from 100N to 60N to 32N. Comparing the output of the optimized ANN to the Bayesian predictions made in Figure 1 show that this network can approximate Bayesian inference for ambiguous numbers, mathematical numbers, and interval numbers (Figure 9).

4. Conclusion

Based on the numerical results, in most cases, trained ANNs can make reasonable predictions for the number game and achieve similar performance to predictions made by explicit Bayesian inference, suggesting that ANNs seem to be able to perform Bayesian inference implicitly to some extent.

The predictions made by ANNs are affected by the architecture including the vertical complexity, horizontal complexity and network shape. Prior probability of the hypotheses space and loss functions also affect the quality of

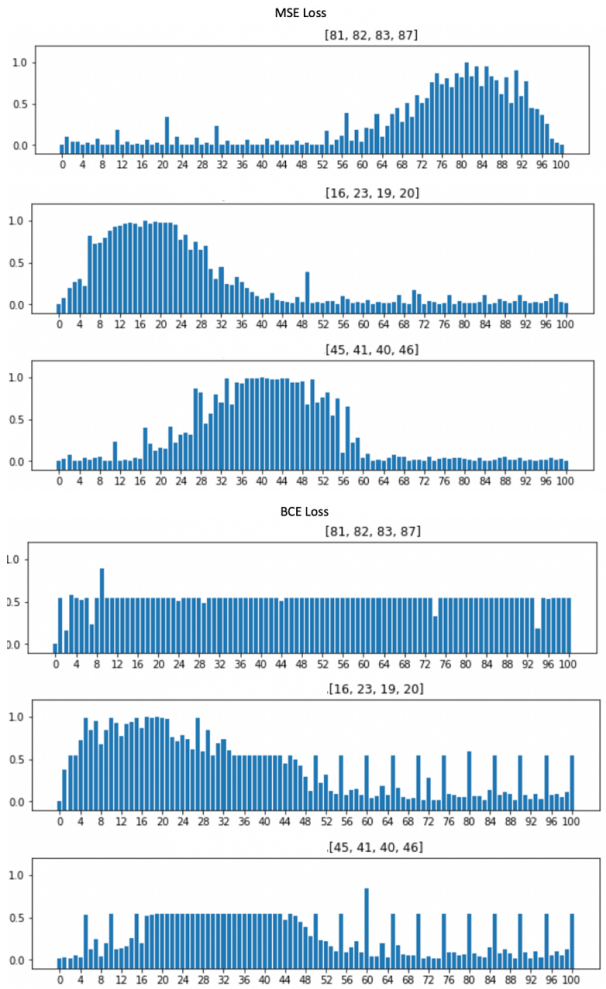

Figure 8. Performance for interval examples between BCE loss and MSE loss functions in networks with 2 layers, 16 nodes per layer.

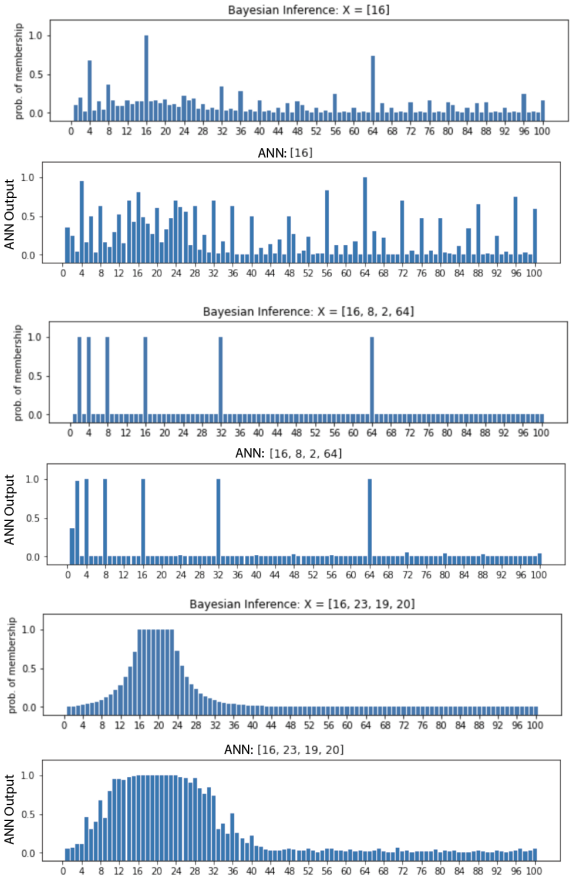

Figure 9. Comparison of theoretical Bayesian predictions with the output of the optimized ANN (3 layers decreasing 100N-60N-32N) for ambiguous numbers (X = [16]), mathematical numbers (X = [16, 8, 2, 64]), and interval numbers (X = [16, 23, 19, 20]).

predictions.

However, one limitation of our strategy is that our trained ANNs do not make optimal Bayesian inference. For example, in the case of example numbers $X = [16, 8, 2, 64]$, which are clearly not drawn from interval hypotheses, if we force the network to have a strong bias towards interval hypotheses ($\lambda = 0.01$), it makes predictions based on interval hypotheses while the optimal Bayesian inference still makes predictions based on mathematical hypotheses. It is unclear what kinds of predictions humans will make and whether humans still make optimal Bayesian inference when they have strong biases. It is likely that humans also perform sub-optimally when biases are enforced. One future experiment would be to compare human predictions with ANN predictions when there is a strong bias towards a certain type of hypotheses.

References

Battaglia, P. W., Jacobs, R. A., and Aslin, R. N. Bayesian integration of visual and auditory signals for spatial localization. 20:1391–1397, 2003.

Elman, J. L. Finding structure in time. 14:179–211, 1990.

Ernst, M. O. and Banks, M. S. Humans integrate visual and haptic information in a statistically optimal fashion. 415: 429–433, 2002.

Ghahramani, Z. Probabilistic machine learning and artificial intelligence. 521:452–459, 2015.

Hillis, J. M., Watt, S. J., Landy, M. S., and Banks, M. S. Slant from texture and disparity cues: Optimal cue combination. 4:967–992, 2004.

Körding, K. P., Beierholm, U., Ma, W. J., Quartz, S., Tenenbaum, J. B., and Shams, L. Causal inference in multisensory perception. 2:1–10, 2007.

Lecun, Y., Bengio, Y., and Hinton, G. Deep learning. 521: 436–444, 2015.

Ma, W. J., Beck, J. M., Latham, P. E., and Pouget, A. Bayesian inference with probabilistic population codes. 9:1432–1438, 2006.

McClelland, J. L., Rumelhart, D. E., and Hinton, G. E. The appeal of parallel distributed processing. pp. 52–57, 2013.

Merfeld, D. M., Zupan, L., and Peterka, R. J. Humans use internal models to estimate gravity and linear acceleration. 398:615–618, 1999.

Orhan, A. E. and Ma, W. J. Efficient probabilistic inference in generic neural networks trained with non-probabilistic feedback. 8:1–14, 2017.

Payeur, A., Guerguiev, J., Zenke, F., Richards, B. A., and Naud, R. Burst-dependent synaptic plasticity can coordinate learning in hierarchical circuits. 2020.

Rogers, J. L. M. T. T. The parallel distributed processing approach to semantic cognition. 4:310–322, 2003.

Tenenbaum, J. B. Rules and similarity in concept learning. 12:59–65, 2000.

Wolpert, D. M., Ghahramani, Z., and Jordan, M. I. An internal model for sensorimotor integration. 269:1880–1882, 1995.

Zador, A. M. A critique of pure learning and what artificial neural networks can learn from animal brains. 10:1–7, 2019.