Can we replace Hierarchical Bayesian Neural Networks with Neural Processes?

Chirag Sarda IIT Gandhinagar
Bhavesh Jain IIT Gandhinagar
Zeel B Patel IIT Gandhinagar
Nipun Batra IIT Gandhinagar

Abstract

We explore the performance of Hierarchical Bayesian Neural Networks (HBNN) compared to Conditional Neural Processes (CNPs) on datasets with limited data points to assess the extent of shared statistical information. CNPs employ an Encoder-Decoder architecture to facilitate information sharing, while HBNN leverages hierarchical model parameters to borrow statistical insights. Our study involves experimentation on both regression and classification tasks, specifically using the MakeMoons dataset while varying contextual conditions. The code is available at `https://github.com/chirag-25/CNP`

1 Introduction and Background

In our report, we experimented with a neural process family called Conditional Neural Processes (CNPs). These CNPs combine the strengths of two approaches: Deep neural networks and Non-parametric Bayesian methods like Gaussian Processes (GPs).

Deep neural networks are good at approximating functions but usually need a large amount of data to get good results. Moreover, it is also trained from scratch for every new task and works well for them. On the other hand, Bayesian methods like GPs can learn the meta behaviour of functions across multiple tasks with the help of right prior knowledge. However, GPs can be computationally expensive, and choosing the right prior information is not always easy.

The CNPs combine the best of both worlds. The main motivation of CNPs is to learn a single model for multiple tasks, i.e. meta modelling. They are inspired by the flexibility of GPs but structured as neural networks, which are trained using gradient descent. This means CNPs can make very accurate predictions even when only seeing a small amount of data for the task at training. They work well with a few shot examples as they use meta-learning to learn from structurally similar data (context). It makes the model solve for new tasks by solving similar tasks before. When faced with a new, similar problem, the model adapts its learned strategies to find solutions, making it efficient in few-shot scenarios.

We've tried using CNPs (Conditional Neural Processes) on different machine learning tasks, such as predicting the pattern of a sine wave, classifying data points in the Make-Moons dataset for various angles, and image creation of the MNIST dataset.

2 Neural Processes

Conditional Neural Processes (CNPs) are a type of machine learning model used for tasks that involve making predictions while considering context information. CNPs are designed to handle both regression (predicting continuous values) and classification (categorizing data points) tasks.

3 Problem Statement

We aim to address the challenge of enabling machine learning models to quickly learn and adapt to new tasks when presented with limited data. Our problem statement revolves around developing and improving techniques that empower neural networks, specifically Conditional Neural Processes, to become proficient in making accurate predictions or decisions when faced with unfamiliar tasks by leveraging insights gained from prior experiences and similar tasks (meta-learning). We also explore the performance of Hierarchical Bayesian Neural Networks (HBNN) compared to Conditional Neural Processes (CNPs) on datasets with limited data points to assess the extent of shared statistical information.

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151.

4 Architecture of the CNP model

A Conditional Neural Process (CNP) architecture consists of three main components: the encoder, the aggregator, and the decoder. These components work together to allow the CNP to process context data and make predictions for target data.

4.1 Encoder

- The encoder takes in the context data as input. Context data typically includes pairs of input points and their corresponding output values.
- The encoder's role is to capture the important information from the context data.
- We implement it using an Artificial neural network. It consists of three fully connected layers with ReLU activation functions.
- The final fully connected layer outputs a 128-dimensional latent representation. It does not have an activation function

4.2 Aggregator

- The aggregator takes the context representation generated by the encoder and processes it to summarize the context information in a useful way for making predictions.
- We used a mean aggregator for our experiments

4.3 Decoder

- The decoder takes the global representation of the context information, along with the target data (input points where predictions are desired), as input.
- We implement it using an Artificial neural network. It consists of three fully connected layers with ReLU activation functions.
- Depending on the task, the decoder may output continuous values (regression) or probability distributions over classes (classification).

5 Experiments using CNP

For this, we experimented with multiple datasets till now, we have worked on sine wave predictions, the Make-Moons dataset and the MNIST dataset.

5.1 Image completion for MNIST dataset

This is a very common experiment which is done using CNPs. We did this experiment to test whether the model that we have created from scratch is actually working or not. We trained on the first 100 images of the dataset and passed 200 points as context (out of 784). Each image in the dataset acts as a context set. All the context points while training and testing are selected randomly.

This was a simple experiment of image completion using CNPs. We trained the model on the MNIST dataset and then gave random context points in the forward pass to predict the target points (all 784 points). We were able to get good results.

We predicted on an image of digit two with different number of context points. From the Figure 1, Figure 2 and Figure 3 we see that as the context points decrease, the quality of prediction decreases.

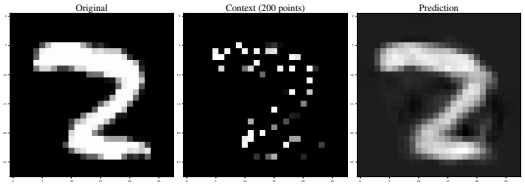

Figure 1: Completion of Image when 200 context points given at the time of prediction

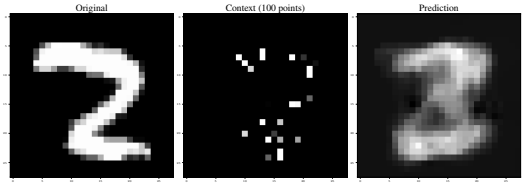

Figure 2: Completion of image when 100 context points were given at the time of prediction

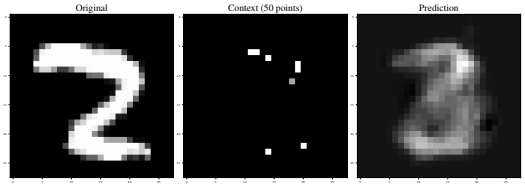

Figure 3: Completion of image when 50 context points were given at the time of prediction

5.2 Image completion for Celab Dataset

We experimented with CelebA image regression with varying numbers of context points. We provide the

model with 1, 10, 100 and 1000 context points and query the entire image. We then constructed the images using the resultant mean.

We first took the example from the seen data, i.e. the image we are completing is in the context set when the model was trained. The Figure 4 shows the mean and standard deviation error in varying the context of images.

We also took the example from the unseen data, i.e. the image we are completing is not in the context set when the model was trained. It is an entirely new image. The Figure 5 shows the mean and standard deviation error in varying the context of images. The CNPs When faced with a new celeb image, the model adapts the learned strategies that it learns while trainng to find solutions, enabling it to make efficient predictions.

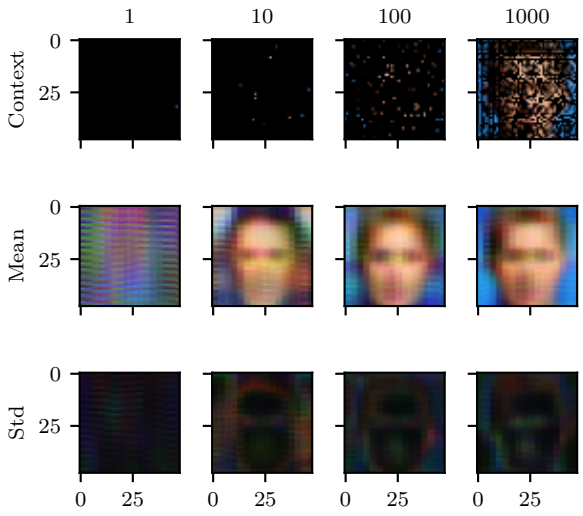

Figure 4: Pixel-wise image completion on CelebA with different contexts when presented with an seen image.

5.3 Sine wave prediction

The idea was to see whether the model could learn the sine wave function from a few context points after training the CNP on a variety of different amplitudes and phases. We did multiple experiments varying both the amplitude and phase of the data. For each experiment, we trained the data on around 100 context sets.

The main goal was to train the model on multiple sine functions having amplitude, frequency and phase varying in some particular ranges and build a model which can take a few points (context points) in the forward pass to determine the exact sine wave function and then predict on the target points accordingly.

While training, each context set had 42 points, out of which we passed 10 points as context and 32 as the target. During testing, we followed the same 10-32 split.

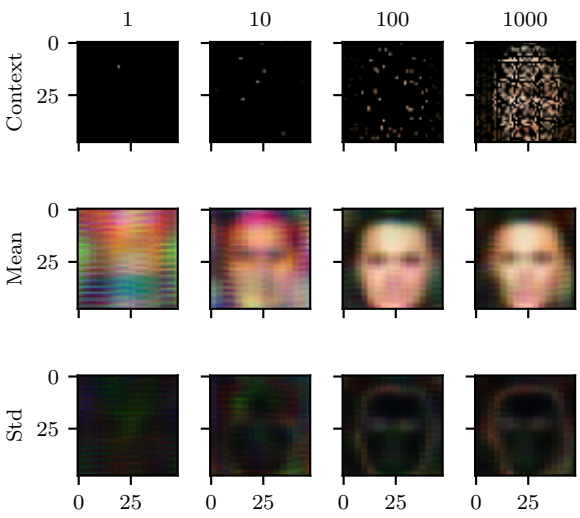

Figure 5: Pixel-wise image completion on CelebA with different contexts when presented with an unseen image.

5.3.1 Training the model on a single sine wave function

In this experiment, we trained the model on a single sine wave function having fixed amplitude, frequency and phase. In this, there was no work for the encoder and it was a simple regression problem. The model was working nicely.

In this, there was no role of context size as seen in the Figure 6 as the encoder does not play any role in this prediction. There is no information like amplitude and phase, which we want to "encode" for each dataset. While training, the decoder would ignore the values received by the encoder.

5.3.2 Training the model on multiple sine wave functions having different amplitudes

In this experiment, we trained the model on multiple sine wave functions having different amplitudes. We selected random 100 amplitudes between 1 and 5 to train the model. While testing, we again gave any amplitude between 1 and 5. The model could predict the sine wave function nicely (Figure 7. This new amplitude in most cases not seen by the model during training.

In the Figure 8 and Figure 9, we can see that the predictions improve as we increase the context size.

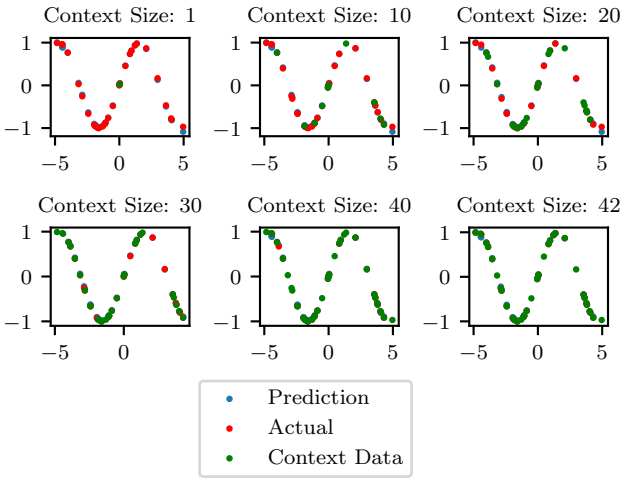

Figure 6: Predictions of sin wave (1 D regression task) with different context sizes using CNP

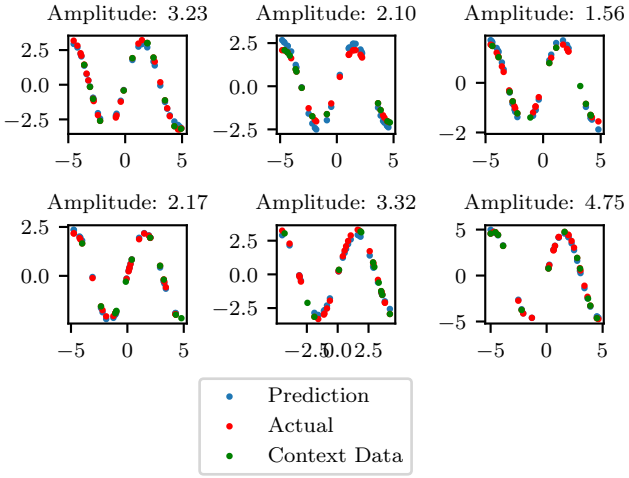

Figure 7: Selected random 100 amplitudes between 1 and 5 to train the model. While testing amplitudes between 1 and 5 were given. The model could predict the sine wave corresponding to that amplitude.

We see that the model does not perform well on extrapolation (Figure 10). The model was trained on data having amplitude between 1 and 5. While testing, as the amplitude of the data exceeded 5, our predictions became poor.

5.3.3 Training the model on multiple sine wave functions having different amplitudes and phases

In this experiment, we trained the model on multiple sine wave functions having different amplitudes and phases. We selected random 100 amplitudes between 1 and 5 and random 100 phases between 0 and pi to train

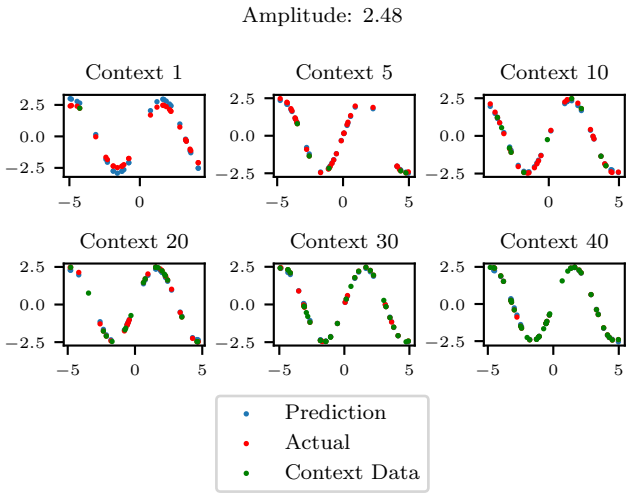

Figure 8: Fixing the amplitude and predictions with different context size.

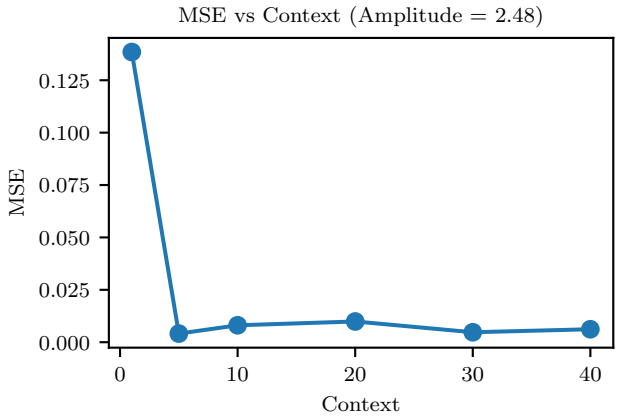

Figure 9: By fixing the amplitude and increasing the context size, the prediction becomes better as error start reducing.

the model. While testing we again gave any amplitude between 1 and 5 and any phase between 0 and π. The model was able to predict the sine wave function nicely as shown in Figure 11. This new amplitude and phase in most of the cases not seen by the model during training.

In Figure 12 and Figure 13 we see that the predictions improve with an increase in context size. This effect is more prominent here than the amplitude experiment as more information about the test set is required.

In Figure 14, the model performs poorly on extrapolation. While testing, as the amplitude and phases of the data went out of the training range, the predictions became poor.

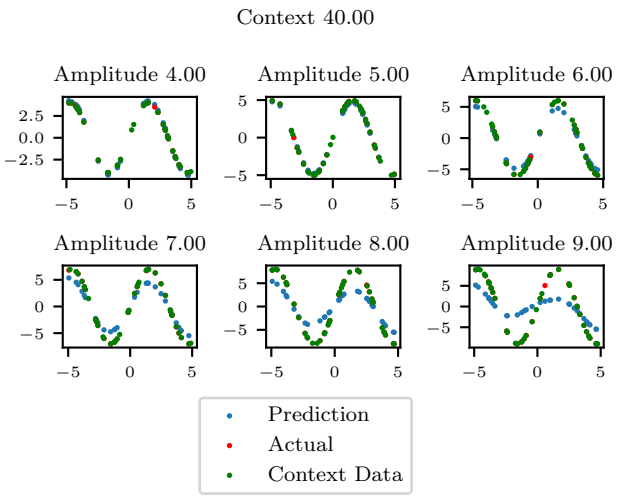

Figure 10: Fixing the context size and then extrapolating the amplitude. Model was trained on the amplitudes between 1 to 5, but as the amplitude exceeds the 5 predictions get poorer.

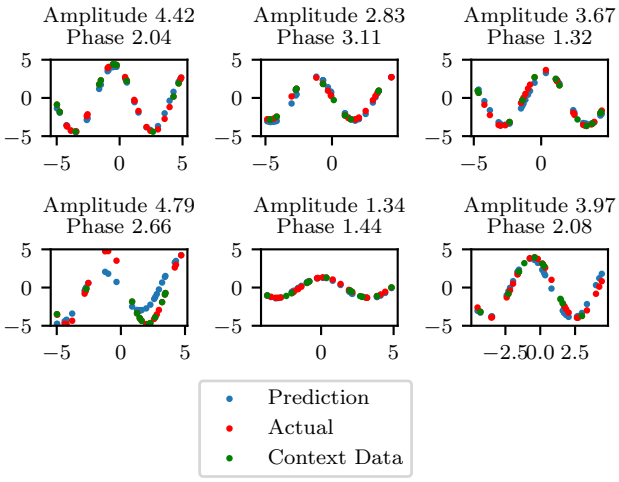

Figure 11: Training on Amplitudes from 1 to 5 and Phase Ranging from 0 to π. Subsequent Testing Involves Predictions on Random Phase and Amplitude Values within the Specified Range

5.4 Rotated Make-Moons Dataset

In this experiment, we trained the CNP on the rotated Make-Moons Dataset. The idea was to send a few context points (similar to a few shots learning) in the forward pass which will help the model figure out the angle of rotation of the moons and then predict the target points accordingly.

During training each context set had 50 random context points and 100 target points. We took random

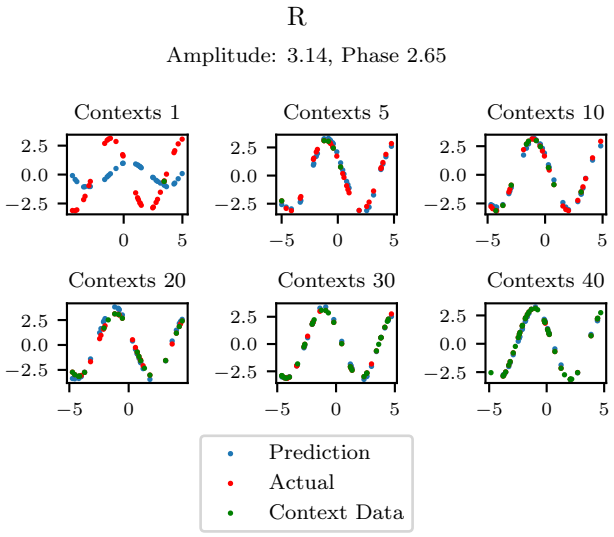

Figure 12: Fixing the amplitude and phase and predictions with different context sizes using CNP

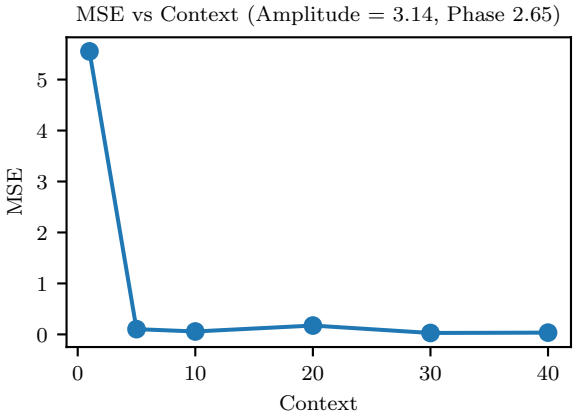

Figure 13: Fixing the Amplitude and Phase Evaluating the Performance CNP on Sin wave prediction task Across Different Contexts

100 angles between 0 and $2 * \pi$ to train the model. While testing we again gave any angle between 0 and $2 * \pi$. The model was able to predict the target points nicely(Figure 15). This new angle in most of the cases not seen by the model during training.

Now we changed the angles while training. Instead of giving angles between 0 and $2 * \pi$, we gave angles between 0 and π. This was done to see how the model performs on extrapolation.

The model was performing well near an angle close to 0 and π. On the rest of the angles, the prediction was poor (Figure 16)

We can also see in Figure 17 how the decision surface

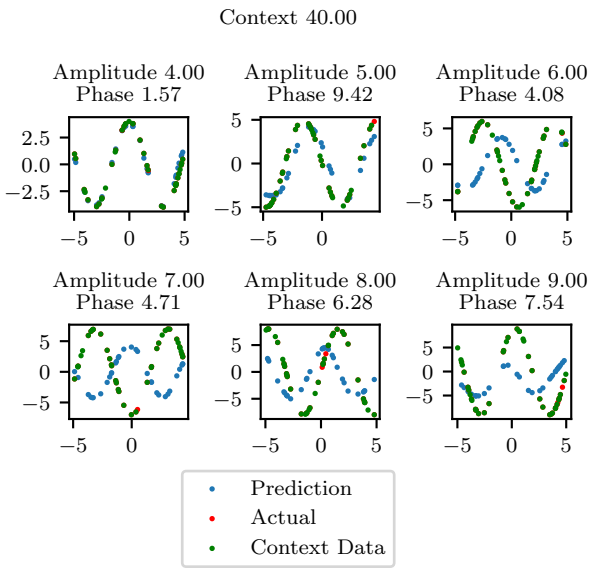

Figure 14: Extrapolation of Phase and Amplitude in Sin Wave Predictions Using CNP

changes with an increase in context points. As we increase the amount of context provided, the decision boundary becomes more accurate.

6 Hierarchical Bayesian Neural Networks

Hierarchical Bayesian modelling is a powerful statistical approach that aids in capturing complex relationships within data while incorporating prior information effectively. In the context of datasets with limited data points, hierarchical Bayesian models become particularly valuable.

The concept of the "pooled model" plays a significant role in sharing statistical strength within hierarchical Bayesian frameworks. In this approach, data from multiple sources or groups are combined into a single model, allowing information to be shared across these groups. This sharing of information helps overcome the challenges posed by small datasets.

When dealing with limited data, the pooled model benefits from a more robust estimation of parameters by borrowing strength from other data points or groups. Essentially, it helps regularize parameter estimates and reduces the risk of over fitting, as information is pooled together to create a more stable and generalized model.

A Hierarchical Bayesian Neural Network (HBNN) is a sophisticated machine learning model that combines neural networks with Bayesian principles, offering a unique tool for handling datasets with limited data points.

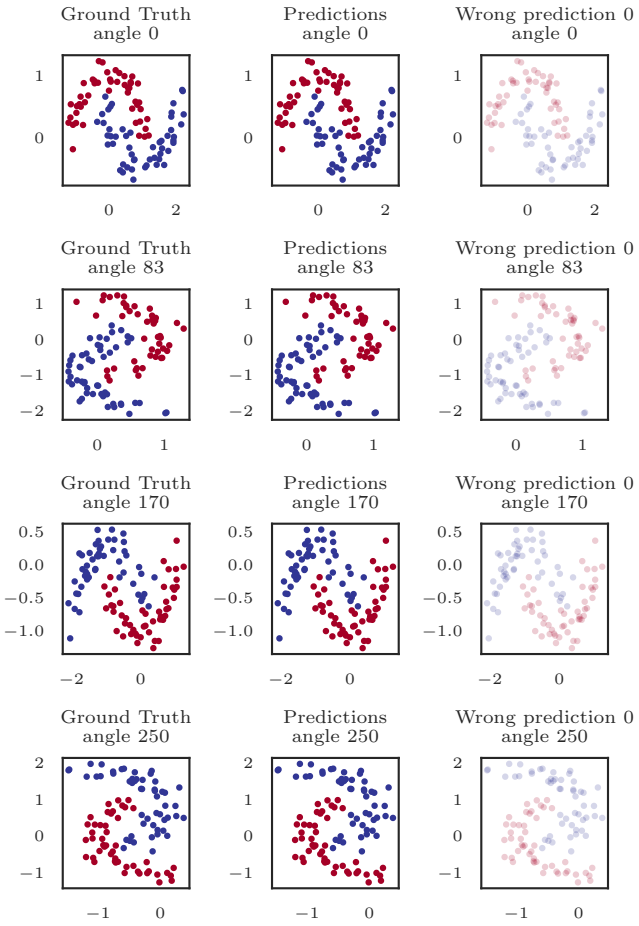

Figure 15: Interpolative Predictions for the Make-Moons Dataset at Four Different Angles Utilizing CNP.

6.1 Classification using HBNN

We used the rotated Make-Moons Dataset as the classification task. We considered four angles (0, 90, 180, 270). The first three angles have strong datasets with 100 points each. For the fourth angle, we vary the points from 10 to 100 to see the performance of both models. During training, we include all points for the first three angles while using a smaller context size for the last angle. We then employ MCMC sampling to generate MCMC chains.

In Figure Figure 19, we see as the number of context points increases: HBNN becomes adept at classifying the dataset into two distinct categories. This is also presented in Figure 20, where we observe an improvement in accuracy with an expanding context.

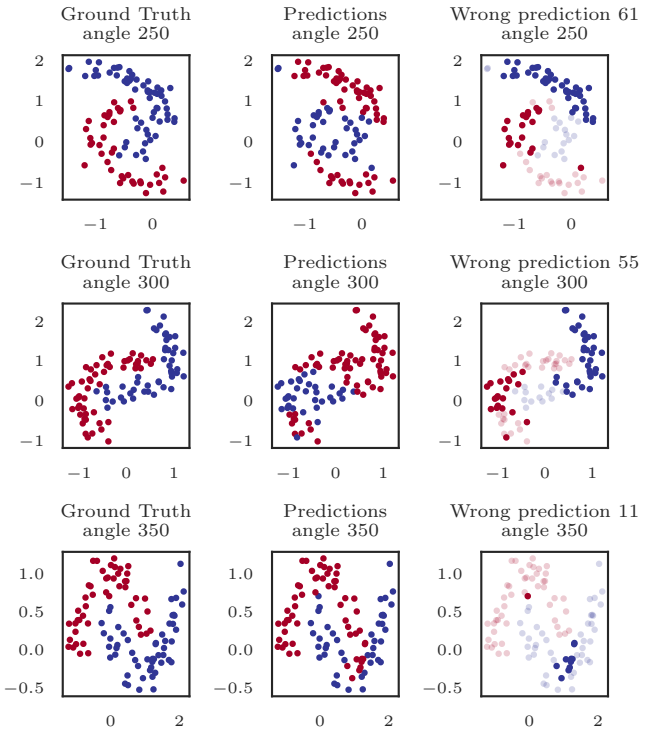

Figure 16: Training on Angles up to 180 Degrees and Extrapolating Predictions for Angles Beyond 180 Degrees.

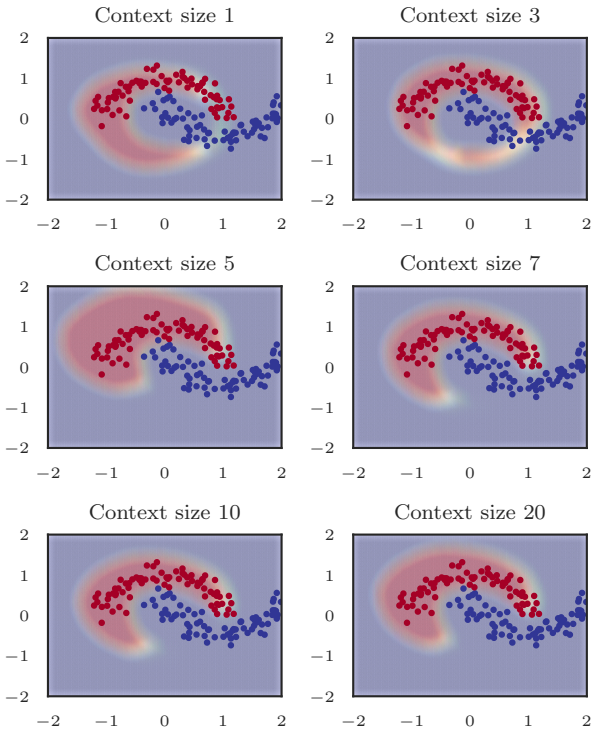

Figure 17: Evoulution of Decision surfaces of Make-Moons with different Context Size, Utilizing CNP

HBNN achieves this feat by utilizing group distributions to estimate weights from the data, effectively capturing variations in weights across different categories. The model's complexity is elegantly managed through Bayesian techniques, allowing it to proficiently model uncertainty. Moreover, it facilitates knowledge sharing between categories, all facilitated by its well-structured hierarchical framework.

6.2 Regression using HBNN

In our experiment, we aimed to predict a sine wave in a regression task, using varying numbers of context points. We considered four different amplitudes (1, 2, 3, and 5). The first three amplitudes were associated with robust datasets of 50 data points each. However, for the fourth amplitude, we deliberately manipulated the number of context points, ranging from 1 to 45, to assess the performance of the HBNN model under different context conditions.

During training, we included all data points for the first three amplitudes. For the fourth amplitude, we employed a smaller context size. To enhance our analysis, we utilized MCMC sampling to generate MCMC chains.

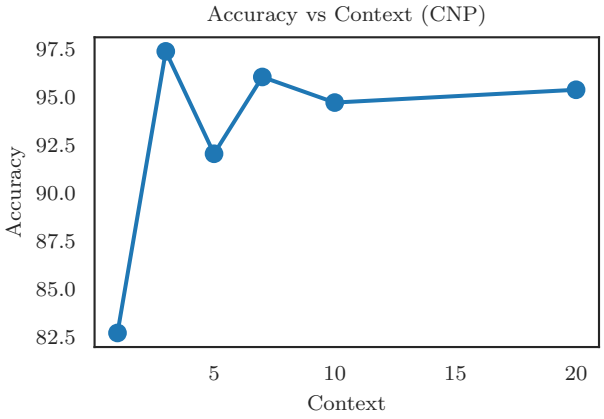

Figure 18: Evaluating the Performance CNP on Make-moon Classification task Across Different Contexts

In Figure 21, we observe that as the number of context points increases, HBNN exhibits a noteworthy ability to predict the sine wave, capturing its amplitude. Additionally, we notice a reduction in uncertainty, indicated by the standard deviation, as the context grows, signifying that the model gains greater confidence in its predictions.

Furthermore, as shown in Figure 22, the mean square error consistently decreases with an expanding con-

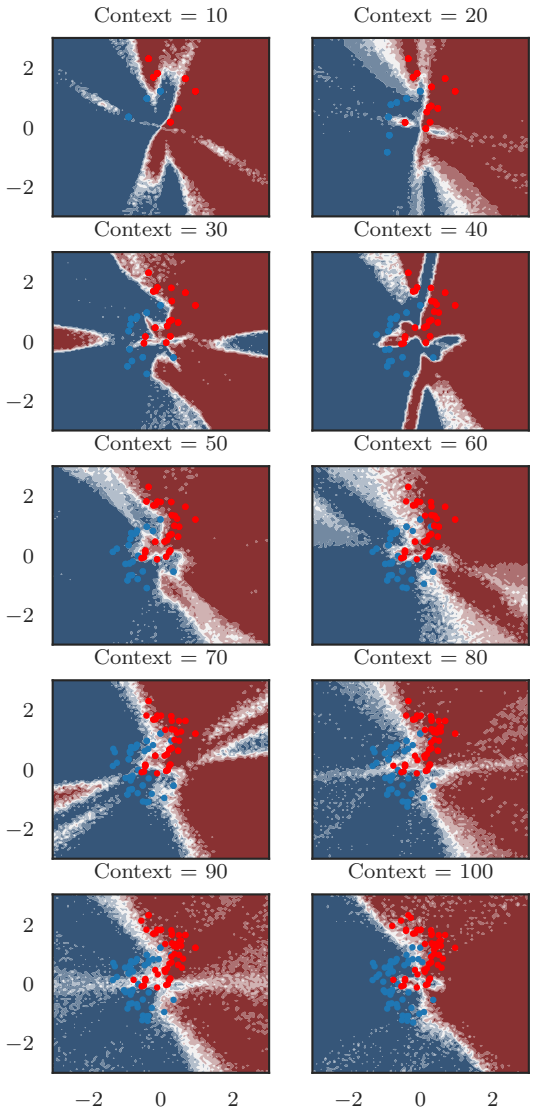

Figure 19: The Evolution of Contour Surfaces for the Rotated Makemoon Dataset at 270 Degrees, as Context Size Varies, Utilizing HBNN

text. This phenomenon is indicative of the model's reduced need to predict new information as context points accumulate, highlighting the model's improved predictive accuracy with a richer contextual foundation.

7 Comparing CNPs and HBNNs

CNP support Meta-Learning using few-shot examples. It help in predicting results for a new dataset with a limited number of points (context points) using the encoder-decoder architecture.

HBNN also enables us to predict results for a statistically weak dataset by sharing statistical strength from similar datasets.

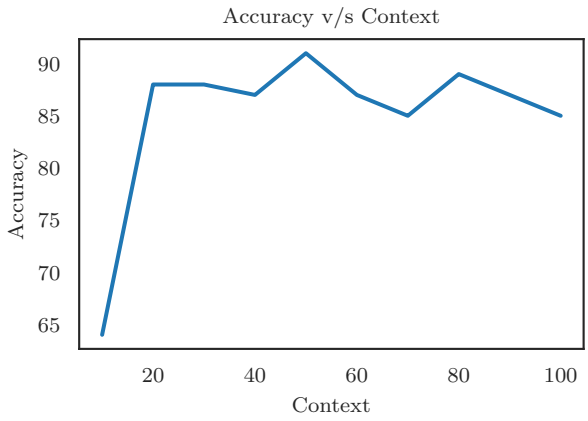

Figure 20: Assessing HBNN's Performance Across Various Context Points in Predicting the Rotated Makemoon Dataset at 270 Degrees.

CNPs and HBNNs can be considered pooled models sharing information across different datasets and tasks. While HBNN uses complex sampling techniques like MCMC, CNPs are neural networks which allow us to predict results using simple forward pass after training.

We conducted an experiment to compare how well two models, CNP and HBNN, with different contexts. For comparison we kept the number of parameters the same for both CNP and HBNN. In this experiment, we divided the models into three categories based on their size: small (1,000 parameters), medium (10,000 parameters), and large (50,000 parameters). We adjusted the model architecture based on these categories. We tested the models with eight different angles (0, 45, 90, 135, 180, 225, 270, 315). The first seven angles had robust datasets with 100 data points each. However, for the last angle, we varied the number of data points from 10 to 100 to observe how both models performed with different model sizes.

From the Figure 23 suggests that CNP's excel in learning from limited data, demonstrating a higher proficiency in leveraging sparse information. In contrast, HBNN tends to require a more substantial data context to perform optimally. Both models exhibit strong performance when presented with richer information, as increased context enables them to harness their full potential. Interestingly, their performance improves with growing model complexity, highlighting their capacity to learn more effectively as the model intricacy increases.

Figures Figure 24 and Figure 25 visually depict the

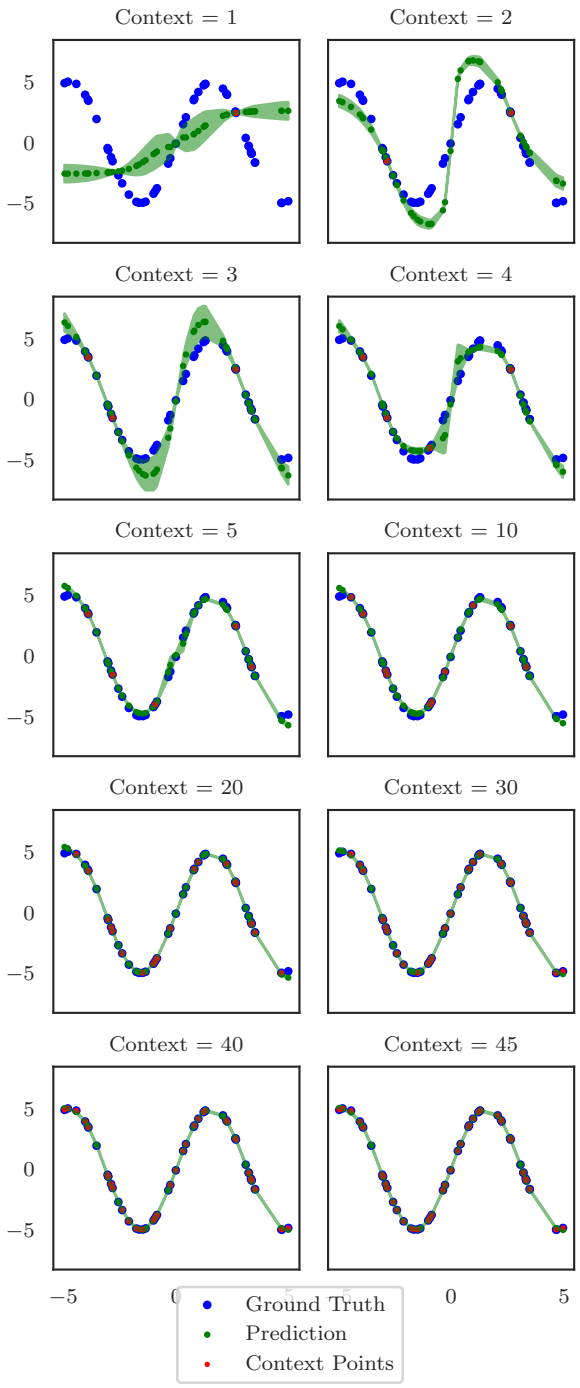

Figure 21: Predictive Outcomes on Regression Task Using HBNN Across Diverse Contextual Settings.

transformation of contour surfaces as the context increases. Specifically, with the large HBNN model, and similarly with the large CNP model, these figures illustrate significant changes. Notably, in the case of the CNP, the contour surface takes on a distinctive crescent moon-like shape, highlighting the unique characteristics and behaviors exhibited by these models under varying context conditions.

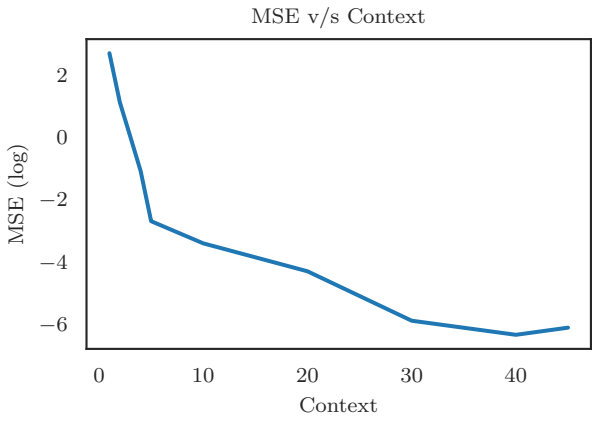

Figure 22: Evaluating the Performance of HBNN for Sin-wave Prediction Across Varying Context Sizes

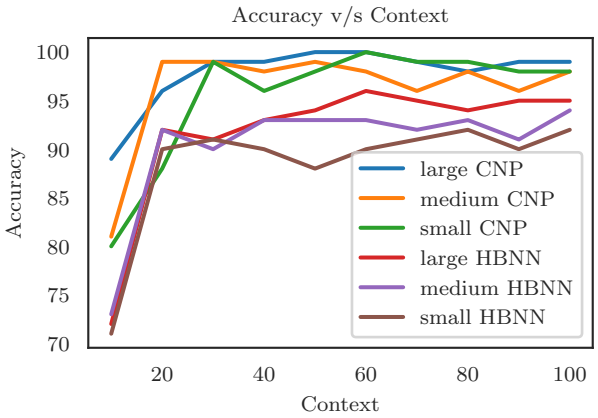

Figure 23: Evaluating the Performance of different size CNP and HBNN Across Varying Context Sizes.

7.1 Imange Completion for MNIST using HBNN and CNP

In our experiment, we examined the performance of HBNNs versus CNPs in image completion tasks, focusing on completing numeral 2 images using various contextual inputs of numeral 2. In Figure 26, the first seven images were given all the data points, while the last image showcases image completion by giving different context sizes. Specifically, Figure 27 illustrates image completion employing the Large HBNN, while Figure 28 presents image completion on MNIST using Large CNPs.

From the analysis of Figure 27 and Figure 28, it's evident that CNPs outperform HBNNs in completing numeral 2 images, especially with smaller context sizes. Notably, CNPs generate highly good numeral 2 images

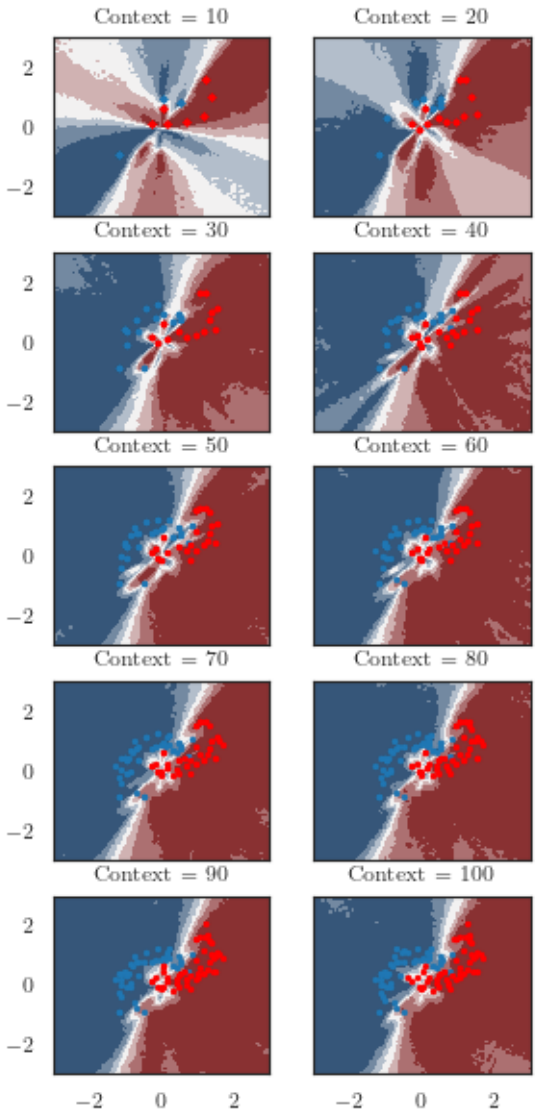

Figure 24: The Evolution of Contour Surfaces at 315 Degrees with Expanding Context Size, Employing a Large HBNN Model.

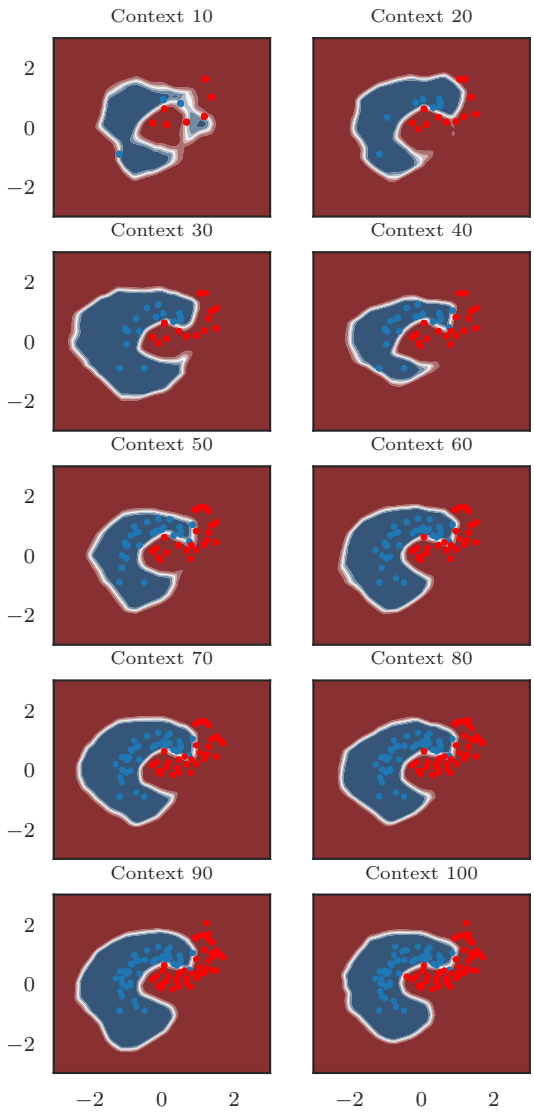

Figure 25: The Evolution of Contour Surfaces at 315 Degrees with Expanding Context Size, Employing a Large CNP Model

with just over 100 context points, whereas HBNNs require more than 700 points (out of a total of 784) to produce faint recognizable numeral 2. This comparison highlights the superior knowledge-sharing capability of CNPs over HBNNs in image completion tasks.

In our experiment the contexts set contains the image of a numeral 2 only, we didn't do the generalisation because even for small similar contexts HBNNs were not constructing a good image. However, we experimented with the CNPs for a generalised context and found that they were able to complete the image as discussed in subsection 5.1.

8 Effect of Additional Knowledge

We experimented on the Makemoons dataset to see the effect of some extra knowledge. In our experiment, we pass the sin of the angle on which the dataset is rotated with its location coordinates. We tested the both CNP and HBNN with eight different angles (0, 45, 90, 135, 180, 225, 270, 315). The first seven angles had robust datasets with 100 data points each. However, for the last angle, we varied the number of data points from 10 to 100 to observe how the models performed with different model sizes.

Figure 29 and Figure 30 show that both HBNN and CNP significantly benefit from the inclusion of addi-

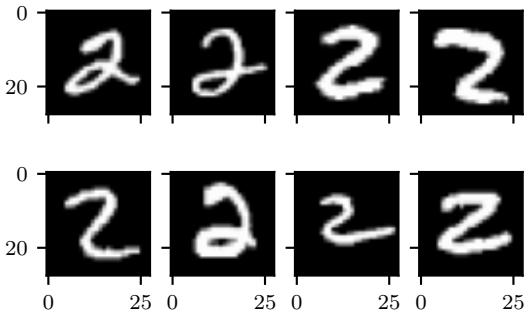

Figure 26: Illustration of Different Styles of letter '2', in this first 7 were given full image while for the last image, few contexts were given and model was tasked to predict the image.

tional information. Notably, HBNN exhibits a pronounced improvement in performance with the introduction of extra data, often surpassing its baseline performance even with a small model. Moreover, it becomes evident that providing more information empowers these models to yield more accurate estimates, particularly when the initial context or information is limited, such as in scenarios with low context. This enhancement underscores the adaptability and responsiveness of both HBNN and CNP to the quantity of information available, ultimately leading to more reliable and robust model predictions.

9 Conclusion

In conclusion, our observations reveal that both CNP and HBNN exhibit improved performance as the number of context points increases. The additional information proves particularly valuable for enhancing predictions, especially when dealing with limited context points. Notably, CNP outperforms HBNN, highlighting its superior predictive capabilities.

10 Future Work

In future, we aim to explore how conditional neural processes (CNP) perform on more intricate datasets. Additionally, our focus will involve analyzing the efficacy of CNP across diverse types of complex datasets, aiming to discern their adaptability and performance in varying scenarios. We'll investigate various CNP variants such as AttnCNP and ConvCNP to understand how they share information and handle complex data. By studying how these different CNP variants share and process information, we aim to uncover insights into their strengths and potential applications in real-world contexts.

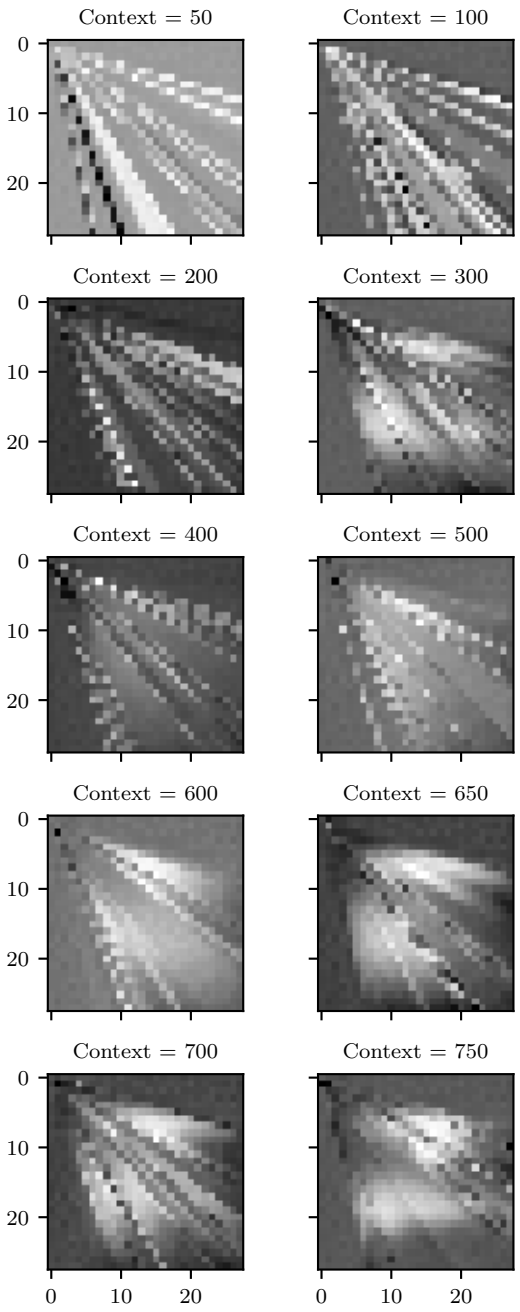

Figure 27: Predicting for the last image from the Figure 26 with Expanding Context Size, Employing a Large HBNN Model

References

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, S. M. Ali Eslami (July 4, 2018) *Conditional Neural Processes* arXiv:1807.01613

Chelsea Finn, Pieter Abbeel, Sergey Levine *Model-*

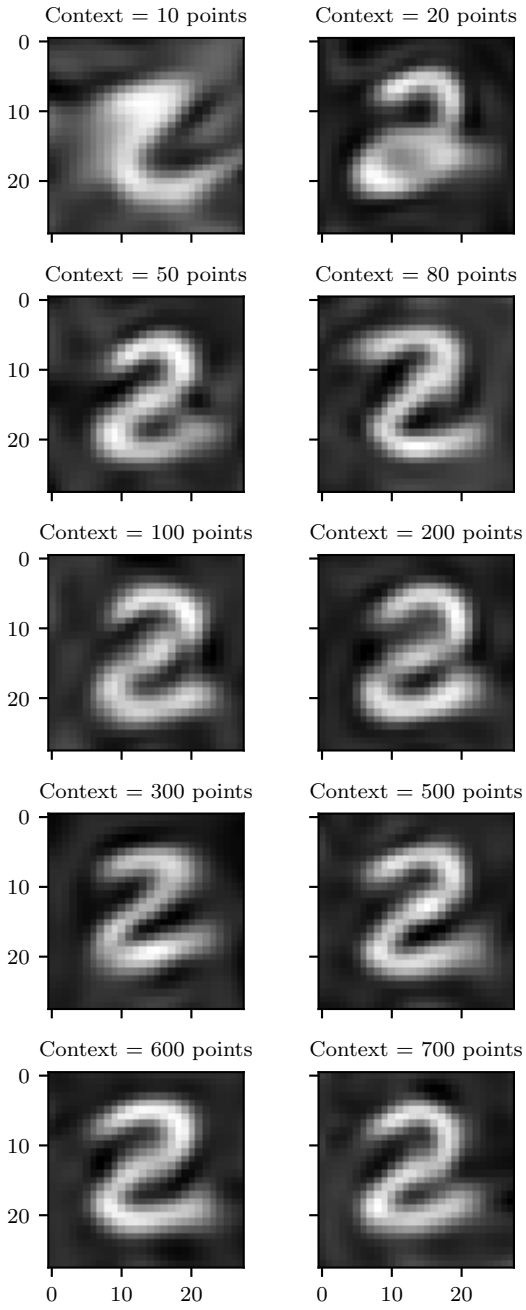

Figure 28: Predicting for the last image from the Figure 26 with Expanding Context Size, Employing a Large CNP Model

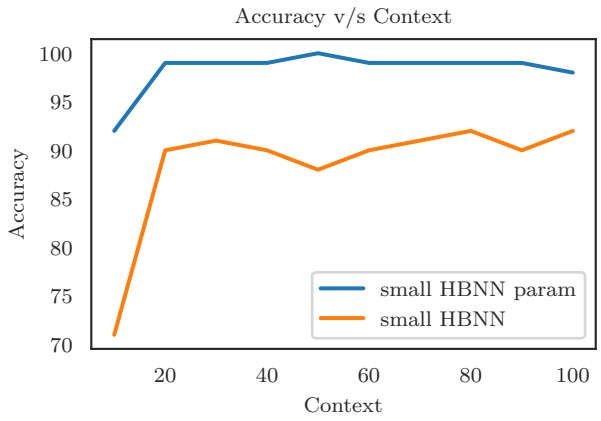

Figure 29: Performance of HBNN on adding extra information (here sin of the angle)

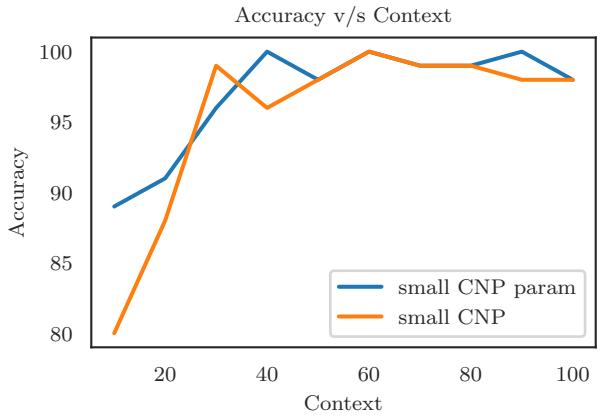

Figure 30: Performance of CNP on adding extra information (here sin of the angle)

Agnostic Meta-Learning for Fast Adaptation of Deep Networks arXiv:1703.03400

Thomas Wiecki (August 13, 2018). *Hierarchical Bayesian Neural Networks with Informative Prior* https://twiecki.io/blog/2018/08/13/hierarchical_bayesian_neural_network/

Lilian Weng (November 30, 2018) *Meta-Learning: Learning to Learn Fast* https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Marta Garnelo - Meta-Learning and Neural Processes (January 22, 2021) https://www.youtube.com/watch?v=q-4lo5luKgc

Dubois, Yann and Gordon, Jonathan and Foong, Andrew YK (September, 2020) *Neural Process Family* https://yanndubs.github.io/Neural-Process-Family/text/Intro.html

Carlos Souza and Chris Stoafer *Bayesian Hierarchical Linear Regression* https://num.pyro.ai/en/stable/tutorials/bayesian_hierarchical_linear_regression.html