Conditional Neural Process for Uncertainty Quantification

Swapnil Kashyap, Aditya Agrawal
Prof. Souvik Chakraborty
Applied Mechanics
Indian Institute of Technology, Delhi
am1210782@iitd.ac.in, am1210198@iitd.ac.in

Abstract

Our project is aimed to reproduce the Conditional Neural Processes (CNPs) framework from Garnelo et al. [2018]. This framework is introduced to address the limitations of Deep Neural Networks (NNs) and Gaussian Processes (GPs), while combining the benefits of both. The most significant advantage of CNPs is that they can predict accurately with just a few training data points and handle both complex functions and big datasets efficiently.

1 Review

Our previous study covered the fundamental equations used in implementing a Conditional Neural Process (CNP) framework. We highlighted the manner in which these principles contribute to the construction of a robust architecture. CNPs possess distinctive characteristics by combining Neural Networks (NNs) and Gaussian Processes (GPs), while also preserving the significant benefits of both NNs and GPs.

In addition, we were able to implement the architecture of the CNP model. It required defining the methods for the encoder, aggregator, and decoder. Our entire implementation is conducted using the PyTorch library, whereas the authors' code is accessible in an outdated version of TensorFlow. This study presents an analysis of the findings obtained from our test on the classical one-dimensional regression task, which serves as a standard benchmark for GPs. As part of our analysis, we compare the predictions and uncertainty provided by CNPs and GPs and showcase the essential advantages of CNPs, such as permutation invariance and scalability.

2 Function Regression

2.1 Implementation

The steps involved in implementing a regression task using CNPs are given below:

1. We generate two different datasets that consist of functions generated from a GP with an exponential kernel.

2. In the first dataset we use a kernel with fixed parameters, and in the second dataset the function switches at some random point on the real line between two functions each sampled with different kernel parameters.

3. At every training step we sample a curve from the GP, select a subset of n points (x_i, y_i) as observations, and a subset of points (x_t, y_t) as target points.

4. Using the model described in our previous reports, the observed points are encoded using a three layer **MLP** encoder h with a 128 dimensional output representation r_i.

5. The representations are aggregated into a single representation $r = \frac{1}{n}\sum r_i$ which is concatenated to x_t and passed to a decoder g consisting of a five layer **MLP**.

6. The decoder outputs a Gaussian mean and variance for the target outputs $\hat{y}_t$.

7. We train the model to maximize the log-likelihood of the target points using the Adam optimizer.

2.2 Result Analysis

We evaluate the model by comparing it to the predictions provided by a Gaussian Process (GP) using the correct hyperparameters. This serves as an upper limit for our performance. While the GP's prediction is smoother than the CNP's prediction in terms of both the mean and variance, the model is capable of learning to regress from a small number of context points for both fixed kernels and flipping kernels. As the quantity of context points increases, the precision of the model increases and the estimated uncertainty of the model diminishes. Importantly, we observe that the model acquires the ability to appropriately predict its own uncertainty based on the data. However, it offers a reliable estimation that improves in precision as the quantity of context points rises. In addition, the model demonstrates comparable high performance on the switching kernel task.

3 Experimental Results

3.1 Single Kernel

3.1.1 50 Context Points

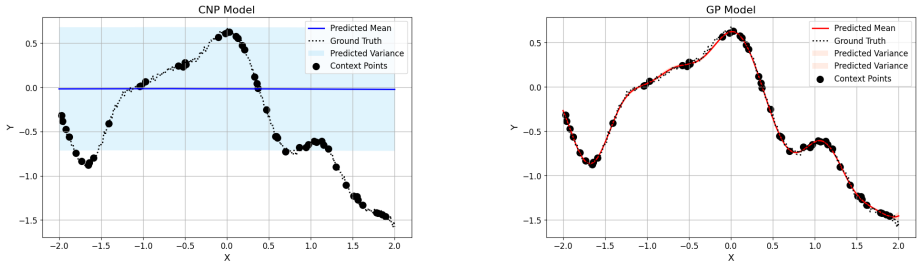


Figure 1: No. of Iterations = 0

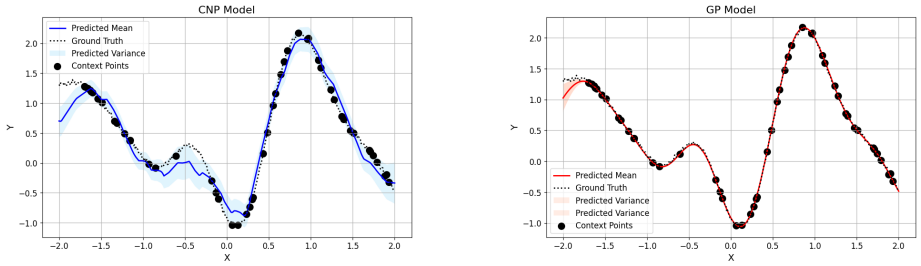


Figure 2: No. of Iterations = 100,000

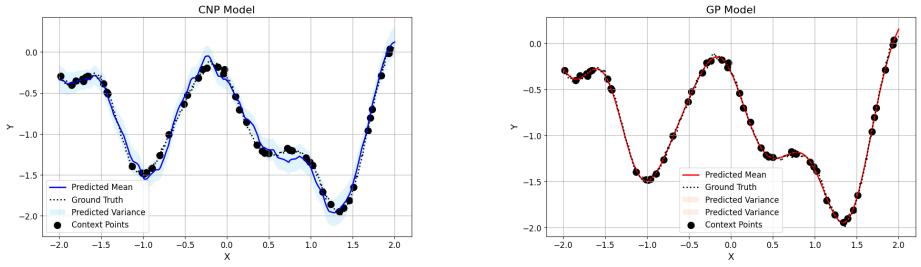


Figure 3: No. of Iterations = 190,000

3.1.2 5 Context Points

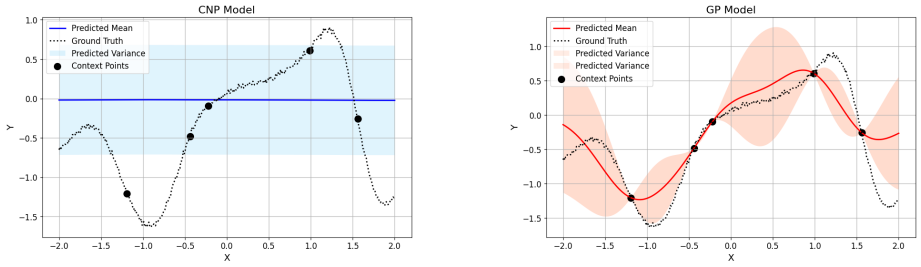


Figure 4: No. of Iterations = 0

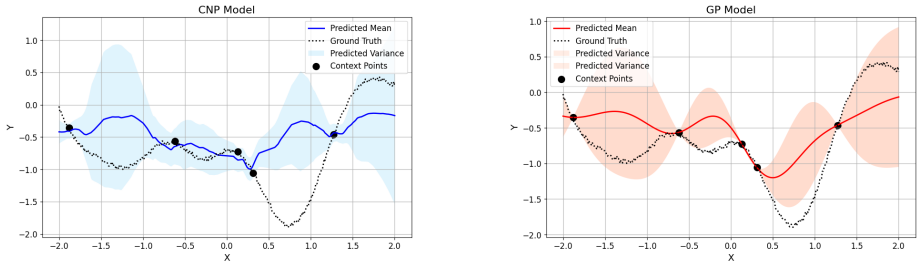


Figure 5: No. of Iterations = 100,000

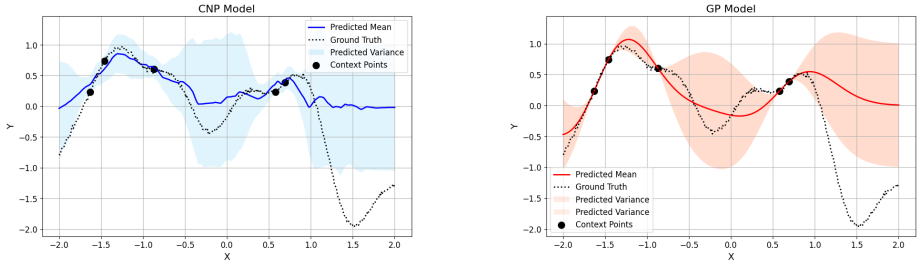
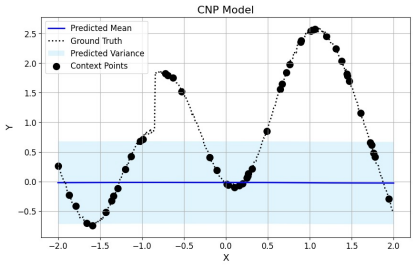


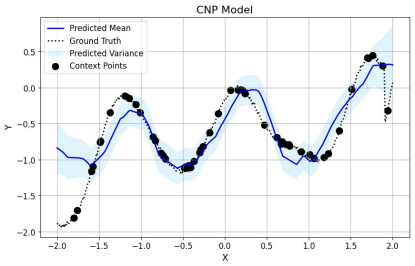
Figure 6: No. of Iterations = 190,000

3.2 Mixed Kernels

3.2.1 50 Context Points



(a) No. of Iterations = 0



(b) No. of Iterations = 70,000

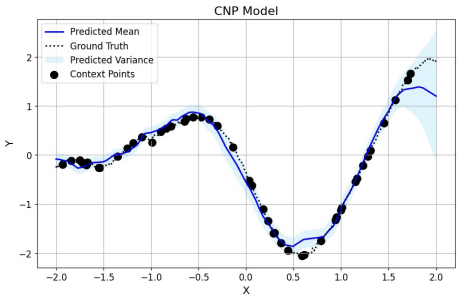
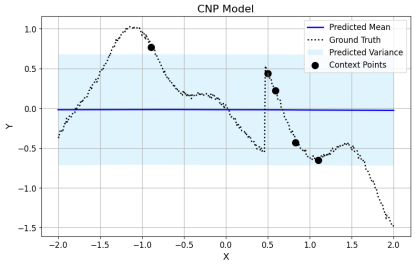
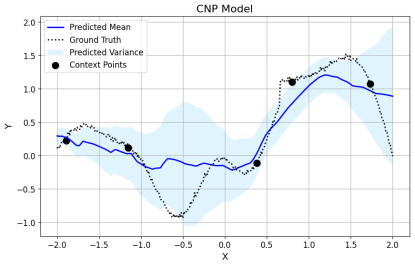


Figure 8: No. of Iterations = 150,000

3.2.2 5 Context Points



(a) No. of Iterations = 0



(b) No. of Iterations = 70,000

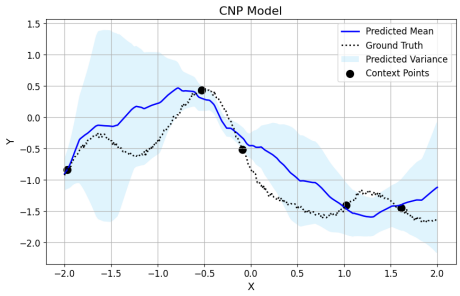


Figure 10: No. of Iterations = 140,000

4 Image Completion

We tried to test CNP on the MNIST dataset but due to lack of time, we could not modify the code to run on the dataset. There were issues with the dimensions of input data while feeding it to the model.

5 Contribution

- Swapnil Kashyap :
  1. Literature survey of the main and reference papers
  2. Development of the entire CNP architecture
  3. Attempted to test the image completion task (MNIST)
- Aditya Agrawal :
  1. Literature Survey of the main paper
  2. Integration of GP generating function in the architecture
  3. Testing of the CNP for the 1-D regression task

References

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Jimenez Rezende, and SM Eslami. Conditional neural processes. In *International Conference on Machine Learning*, 2018. URL `https://github.com/google-deepmind/neural-processes?tab=readme-ov-file`.