0.1 Introduction

Convolutional Neural Networks (CNNs) have been the state of the art in many image processing tasks [He et al. 2015; Simonyan and Zisserman 2015; Krizhevsky, Sutskever, and Hinton 2012], due to their ability to learn spatial patterns in the data via convolutional filters. Since convolutions are translation equivariant, CNNs learns to recognize patterns regardless of their position in the image, making them desirable for a wide range of tasks, such as image classification, object detection, and segmentation. The performance of CNNs has made them to be a desirable backbone for a Neural Process, motivating the development of the Convolutional Neural Processes (ConvNPs) [Gordon et al. 2020]. We will briefly discuss the ConvNP model and its architecture however for a more comprehensive explanation, we refer the reader to the original paper.

0.2 Model

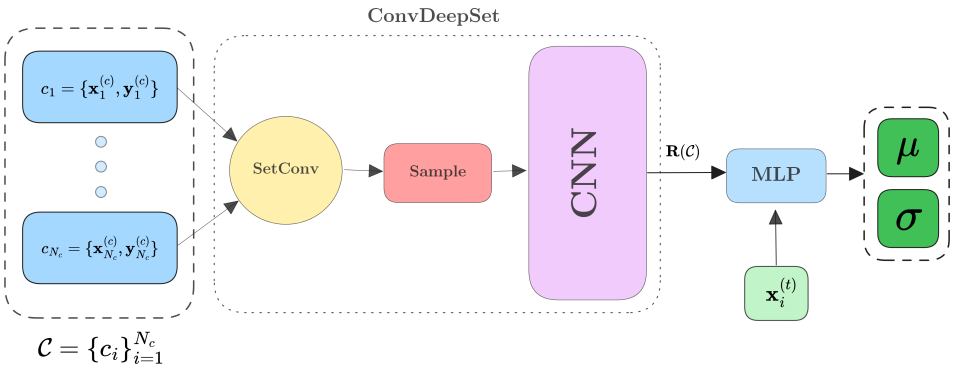

Figure 0.2.1: ConvNP Architecture. All the data points are transformed into a continuous functional embedding using the SetConv, then discretely sampled to be fed into a CNN. The CNN processes the data whose output is converted back to a continuous function using the SetConv operation.

There are a few hurdles are preventing us from directly plugging data into a CNN when working with off-grid data.

CNNs operate on **on-grid data**, implying the data is on a regular grid, such as an image. However, the data we are working with is sometimes **off-grid data**, e.g. time series dataset with irregular timestamps. Furthermore, to 'bake-in' the translation equivariance property, we need to be able to shift the data in the input space and the output space consistently. This is not trivial when using standard vector representations of the data as their is no inherent notion translation in a vector. [Gordon et al. 2020] propose a solution to this problem by using **functional embeddings** to model the discrete data as *continuous functions*. With functional embeddings, the data can be translated in the input space and the output space consistently thus permitting translation equivariance.

The **SetConv** operation takes a set of input-output pairs and outputs a continuous functional representation of the data as follows:

$$\text{SetConv}(\{\boldsymbol{x_i}, \boldsymbol{y_i}\}_{i=1}^{N})(x) = \sum_{i=1}^{N} [1, \boldsymbol{y_i}]^T \mathcal{K}(\boldsymbol{x} - \boldsymbol{x_i}) \tag{0.2.1}$$

Where $\mathcal{K}$ is a Kernel function that measures the distance between the query x and the data point $\boldsymbol{x_i}$.

This operation has some key properties:

- We append 1 to output $\boldsymbol{y_i}$ when computing the SetConv, this acts as a flag allowing the model to knows which data points are observed and which are not. Say we have a data point of $\boldsymbol{y_i} = 0$, then if we did not append the 1, the model would not be able to distinguish between an observed data point and an unobserved data point (as both would be 0).
- The 'weight' of the Kernel depends only on the *relative distance* between points on the input space hence the model is translation equivariant.
- The summation over the data points naturally introduces **Permutation Invariance** to the model.

The function representation of the context set, is sampled at **evenly** spaced points giving us discrete data in a **on-grid** format. It is then fed into a CNN. After the CNN processes the data, it need to convert it back to a continuous function by using the SetConv operation. The final output of the encoder is a continuous function that represents the context set which can be queried at any point in the input space using the target set, given by:

$$R(\boldsymbol{x_t}) = \text{SetConv}(\text{CNN}(\{\text{SetConv}(\mathcal{C})(\boldsymbol{x_d})\}_{d=1}^{D}))(\boldsymbol{x_t}) \tag{0.2.2}$$

The decoder is a simple MLP that takes the output of the encoder and the target set as input and generates the mean and variance of the predictive distribution $\boldsymbol{\mu}(\boldsymbol{x_t}), \boldsymbol{\sigma}(\boldsymbol{x_t}) = \text{MLP}(R(\boldsymbol{x_t}))$

Bibliography

Gordon, Jonathan et al. (2020). *Convolutional Conditional Neural Processes*. arXiv: 1910.13556 [stat.ML].

He, Kaiming et al. (2015). *Deep Residual Learning for Image Recognition*. arXiv: 1512.03385 [cs.CV].

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Simonyan, Karen and Andrew Zisserman (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. arXiv: 1409.1556 [cs.CV].