SUPPLEMENTARY MATERIAL

A. PROOF OF THEOREM 2

First, we prove that (II) implies (I). We define the action of G on the set of univariate maps $f : \mathcal{S} \to \mathbb{R}$ by

$$(g \cdot f)(\boldsymbol{x}) := f(g^{-1} \cdot \boldsymbol{x}).$$

and define the action of G on the set of bivariate maps $\psi : \mathcal{S}^2 \to \mathbb{R}$ by

$$(g \cdot \psi)(\boldsymbol{x}, \boldsymbol{x}') := \psi(g \cdot \boldsymbol{x}, g \cdot \boldsymbol{x}').$$

Lemma 4. *For a map $\psi : \mathcal{X}^2 \to \mathbb{R}^d$ and a sample $\boldsymbol{x} \in \mathcal{X}$, a sample dependent function $\psi_{\boldsymbol{x}} : \mathcal{X} \to \mathbb{R}^d$ is defined by*

$$\psi_{\boldsymbol{x}}(\boldsymbol{x}') := \psi(\boldsymbol{x}', \boldsymbol{x})$$

Here, ψ is G-invariant if and only if the map $\mathcal{X} \ni \boldsymbol{x} \mapsto \psi_{\boldsymbol{x}} \in \mathsf{Map}(\mathcal{X}, \mathbb{R}^{\mathsf{d}})$ is G-equivariant.

The above lemma is derived as follows:

$$\psi_{g \cdot \boldsymbol{x}'}(\boldsymbol{x}) = \psi(\boldsymbol{x}, g \cdot \boldsymbol{x}') = \psi(g^{-1} \cdot \boldsymbol{x}, \boldsymbol{x}') = \psi_{\boldsymbol{x}'}(g^{-1} \cdot \boldsymbol{x}) = (g \cdot \psi_{\boldsymbol{x}'})(\boldsymbol{x}).$$

The left hand side represents the action on the sample space and the right hand side does the action on the function space.

When we denote the set of all G-equivariant maps from $\mathcal{S}$ to $\mathcal{S}'$ by $\mathsf{Equiv}(\mathcal{S}, \mathcal{S}')$, the above lemma is represented as

$$\mathsf{Inv}(\mathcal{X}^2, \mathbb{R}^{\mathsf{d}}) \cong \mathsf{Equiv}(\mathcal{X}, \mathsf{Map}(\mathcal{X}, \mathbb{R}^{\mathsf{d}})).$$

Next, we prove that (I) implies (II). We prepare some notations and lemmas in the following. Let ψ be an interpolating continuous kernel that satisfies $\psi(\boldsymbol{x}, \boldsymbol{x}') \geq 0$. Then, for $m \in \mathbb{N}$ and $\mathcal{Z}'_m \subseteq (\mathcal{S} \times \mathcal{Y})^m$, define

$$\mathcal{H}_m(\mathcal{Z}'_m) := \left\{ \sum_{i=1}^m \phi_{K+1}(y_i)\, \psi(\cdot, \boldsymbol{x}_i) : (\boldsymbol{x}_i, y_i)_{i=1}^m \subseteq \mathcal{Z}'_m \right\} \subseteq \mathcal{H}^{K+1},$$

where $\mathcal{H}^{K+1} = \mathcal{H} \times \cdots \times \mathcal{H}$ is the $(K+1)$-dimensional-vector-valued-function Hilbert space constructed from the RKHS $\mathcal{H}$ for which ψ is a reproducing kernel and endowed with the inner product $\langle f, g \rangle_{\mathcal{H}^{K+1}} = \sum_{i=1}^{K+1} \langle f_i, g_i \rangle_{\mathcal{H}}$. Denote

$$\left[\mathcal{Z}'_{\leq M}\right] = \bigcup_{m=1}^M \left[\mathcal{Z}'_m\right] \quad \text{and} \quad \mathcal{H}_{\leq M} = \bigcup_{m=1}^M \mathcal{H}_m(\mathcal{Z}'_m).$$

Lemma 1 and Lemma 3 in Gordon et al. (2019) provides the following lemma.

Lemma 5. *For $m \in \mathbb{N}$, let $\mathcal{Z}'_m \subseteq (\mathcal{S} \times \mathcal{Y})^m$ be a set with multiplicity K and ψ be an interpolating continuous kernel. Then, $(\mathcal{H}_m(\mathcal{Z}'_m))_{m=1}^M$ are pairwise disjoint and the embedding E is injective and continuous:*

$$E : \left[\mathcal{Z}'_{\leq M}\right] \to \mathcal{H}_{\leq M}(\mathcal{Z}'_m), \quad E([Z]) := E_m([Z]) \quad \textit{if} \quad [Z] \in \left[\mathcal{Z}'_m\right],$$

where

$$E_m : \left[\mathcal{Z}'_m\right] \to \mathcal{H}_m(\mathcal{Z}'_m), \quad E_m\left([(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)]\right) := \sum_{i=1}^m \phi_{K+1}(y_i)\, \psi(\cdot, \boldsymbol{x}_i)$$

Similarly, Lemma 2 and Lemma 4 in Gordon et al. (2019) provides the following lemma.

Lemma 6. *Suppose that $\mathcal{Z}'_M$ is a topologically closed set in $(\mathcal{S} \times \mathcal{Y})^M$ and permutation-invariant, and that ψ satisfies (i) $\psi \geq 0$, (ii) $\psi(x, x) = \sigma^2 > 0$ for any x, and (iii) $\psi(x, x') \to 0$ as $\|x\| \to \infty$. Let $\Phi : \left[\mathcal{Z}'_{<M}\right] \to C_b(\mathcal{S}, \mathcal{Y})$ be a map such that every restriction $\Phi|_{[\boldsymbol{z}'_m]}$ is continuous. Then, $\Phi \circ E^{-1} : \mathcal{H}_{<M} \to C_b(\mathcal{S}, \mathcal{Y})$ is continuous.*

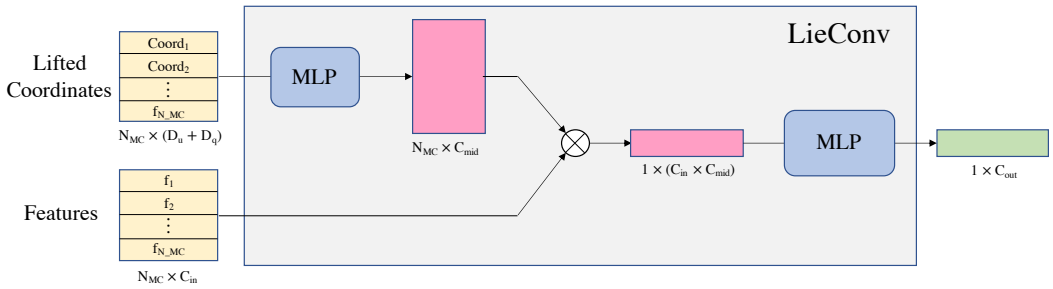
When a G-equivariant function f is injective, $f^{-1}|_{\mathrm{Im} f}$ is also G-equivariant on the image of f. Denoting $\Phi \circ E^{-1}$ by ρ, we can rewrite as $\Phi = \rho \circ E$.

B. SEPARABLE LIECONV

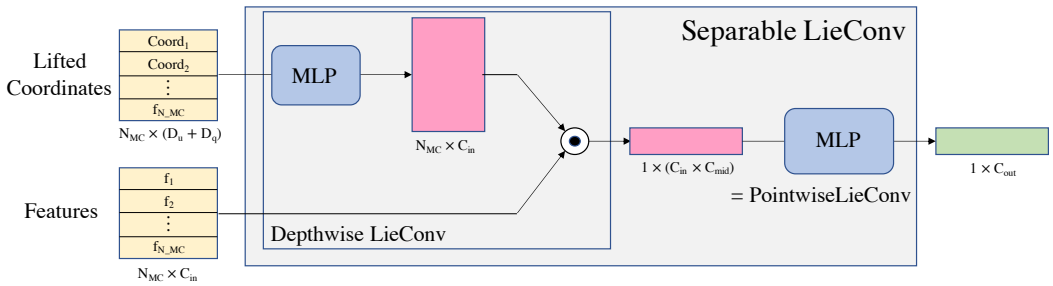
In this section, we introduce the separable LieConv that we design and implement for EquivCNP. LieConv (Finzi et al., 2020) is based on PointConv (Wu et al., 2019), which is proposed for point cloud convolution. That is, the lifted inputs are convolved by PointConv. Therefore, we can adopt techniques that are used for general convolution. One of such techniques is separable convolution(Chollet, 2017). Separable convolution consists of depthwise convolution and pointwise convolution (as known as 1 x 1 convolution). The mathematical formulation of normal convolution, the pointwise convolution, and the depthwise convolution is as follow:

$$\begin{aligned}
\text{Conv}(W, y)(i, j) &= \sum_{k,l,m}^{K,L,M} W_{(k,l,m)} \cdot y_{(i+k,j+l,m)} \\
\text{PointwiseConv}(W, y)(i, j) &= \sum_{m}^{M} W_m \cdot y_{(i,j,m)} \\
\text{DepthwiseConv}(W, y)(i, j) &= \sum_{k=1}^{K,L} W_{(k,l)} \odot y_{(i+k,j+l)} \\
\text{SepConv}\left(W_p, W_d, y\right)(i, j) &= \text{PointwiseConv}(i, j)\left(W_p, \text{Depthwise Conv}\left(i, j\right)\left(Wd, y\right)\right)
\end{aligned}$$

Thanks to the assumption that convolution operation is separable to the spatial direction and the channel direction, the separable convolution provides the way to operate convolution more efficiently than general convolution. Note that the difference of the efficiency between LieConv and separable LieConv is slight; the difference between the matrix production and element-wise product. Following the equation above, we design and implemented separable LieConv. Figure 5 illustrates the processing of (a) normal LieConv and (b) separable LieConv. The memory consumption is also different that the output shape of after the convolutional weights (kernel) is calculated in normal LieConv is $B \times N_{MC} \times C_{mid}$ while that of separable LieConv is $B \times N_{MC} \times C_{in}$.



(a) LieConv



(b) Separable LieConv

Figure 5: Separable LieConv. Difference between (a) normal LieConv and (b) separable LieConv is the matrix product $\otimes$ and elemente-wise product $\odot$.

C. EQUIVCNP ARCHITECTURE

The architecture of EquivCNP is following that of ConvCNP (Gordon et al., 2019), so that we can fairly compare them. It is difficult to determine radius r of LieConv because the radius is varied substantially between the different groups due to the different distance functions. Instead, we parametrized the radius by specifying the average fraction of the total number of convolved elements that would fall into this radius. Therefore, we describe the value of the average fraction instead of kernel size that is described in other papers as usual. Simultaneously, while the conventional convolutional layer has a parameter called *stride* that determines the target elements (pixels) to be convolved, LieConv has a parameter sampling fraction instead of stride to subsample the group elements; sampling fraction is 1.0.

C.1 1D SYNTHETIC REGRESSION TASK

For 1D regression tasks, we use 4-layer LieConv architecture with ReLU activations. The average fraction of those LieConv is $\frac{5}{32}$ and the number of MC sampling is 25. The channels of LieConv are $[16, 32, 16, 8]$. Functional representation $E(Z)$ is concatenated with target point x_T, followed by lifting. After operating convolution to the lifted inputs, we use a softplus activation following the last fully-connected layer (FC) as a standard deviation. Note that the output of EquivCNP, mean and standard deviation, is sliced to get those of y_T. The architecture of EquivCNP for a 1D regression task is illustrated in Figure 6.

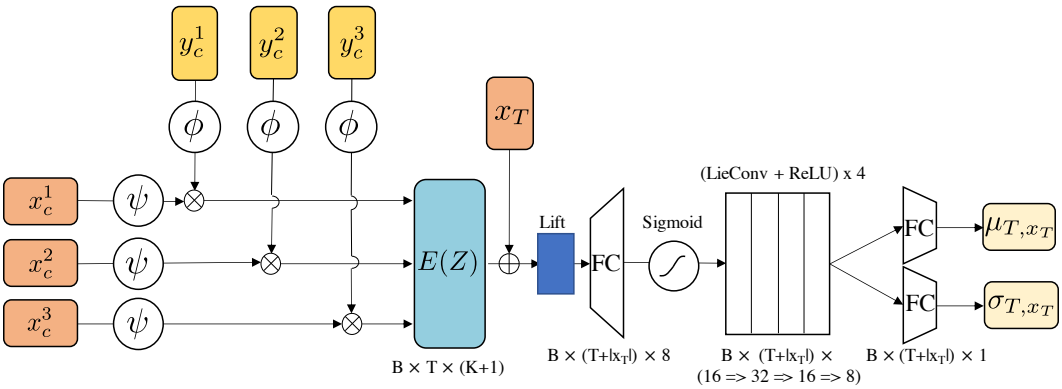


Figure 6: The architecture of EquivCNP for a 1D regression task. $\otimes$ represents dot product and $\oplus$ represents concatenation. ψ is a RBF kernel and $\phi = [y^0, y^1, \ldots, y^K]$.

C.2 2D IMAGE-COMPLETION TASK

For the 2D image-completion task, we use LieConv Conv_θ instead of RBF kernels as ψ. The channels of this LieConv is 128, the average fraction is $\frac{1}{10}$, and the number of MC sampling is 121. After the LieConv of ψ, we use four residual blocks. Each block is composed by two separable LieConv layers and residual connections as shown in Figure 7. The channel of each residual block is 128, the average fraction is $\frac{1}{15}$, and the number of MC sampling is 81.

We employ the same procedure of ConvCNP (Gordon et al., 2019) for image-completion as follows:

1. Given an input image $\mathrm{I} \in \mathbb{R}^{C \times H \times W}$, where C is color channel, H and W represents height and width respectively, sample context points features $:= \mathrm{I} \odot \mathrm{M}_c$ from bernoulli distribution. M_c means the density as same as we define ϕ during 1D regression task.
2. After lifting the inputs, apply a LieConv to both $\mathrm{I} \odot \mathrm{M}_c$ and M_c to get functional representation: $E(Z) = \text{Conv}_\theta([\mathrm{M}_c, \mathrm{I} \odot \mathrm{M}_c]) \in \mathbb{R}^{(128+128) \times H \times W}$.
3. Then, functional representation $E(Z)$ is passed through one FC followed by four residual blocks: $h = \text{ResBlocks}(\text{FC}(E(Z))) \in \mathbb{R}^{128 \times H \times W}$.

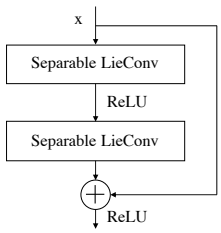


Figure 7: Residual Block

4. Finally, we use one FC to get mean and standard deviation channels and split the output $\in \mathbb{R}^{2C \times H \times W}$ into those statistics.

D. Experiment Details

In this section, we describe the experiments in more detail. Code and dataset will be made available upon publication.

D.1 1D Synthetic Regression Task

The kernels used in Section 5.1 for generating the data via Gaussian Processes are defined as follows:

- RBFKernel:

$$k(x_1, x_2) = \exp\left(-\frac{(x_1 - x_2)^2}{2}\right)$$

- Matern-$\frac{5}{2}$

$$k(x_1, x_2) = \left(1 + \sqrt{5}d + \frac{5}{3}d^2\right)\exp\left(-\sqrt{\frac{5}{2}}d\right) \quad \text{with} \quad d = \|x_1 - x_2\|_2$$

- Periodic

$$k(x_1, x_2) = \exp\left(-2\sin(\pi\|x_1 - x_2\|_2)\right)$$

To train all NPs, the GPs generate the context and target points; the number of context points and target points is random-sampled uniformly from $[3, 50]$ respectively. All NPs were trained for 200 epochs by 256 batches per epoch and the size of each batch is 16, We used Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-3}. An architecture of CNP was based on the original code[2]. We visualize the result of periodic kernel regression at Figure 8.

We also demonstrate EquivCNP with the algorithm following that of ConvCNP (Gordon et al., 2019); regarding the output of EquivCNP as weights for evenly-spaced basis functions (i.e. RBF kernel) in Figure 9. The result of predictive distribution is much smoother than the result of our Algorithm 1 though using RBF kernel is redundant.

D.2 2D Image-Completion Task

The original image of the digital clock number is shown in Figure 10. We first inverted in colors of black and white of the image. Then, we cropped the image so that each cropped image contains one digit and resize them to 64×64. Note that the vertical size of each number is set up to 56, while the horizontal size is not fixed. The values of all pixels are devided by 255 to rescale them to the $[0, 1]$ range.

[2] https://github.com/deepmind/neural-processes

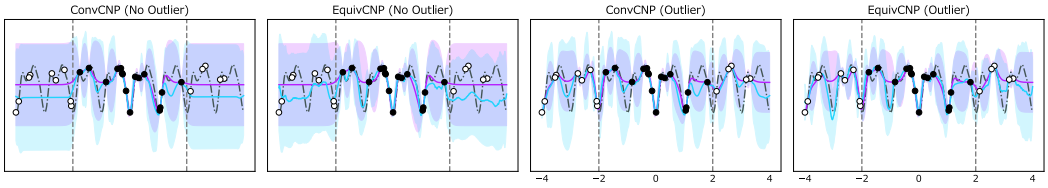


Figure 8: Predictive mean and variance of ConvCNP and EquivCNP at periodic kernels. First two columns show the result without outlier observation and last two columns show the result with outlier observation.

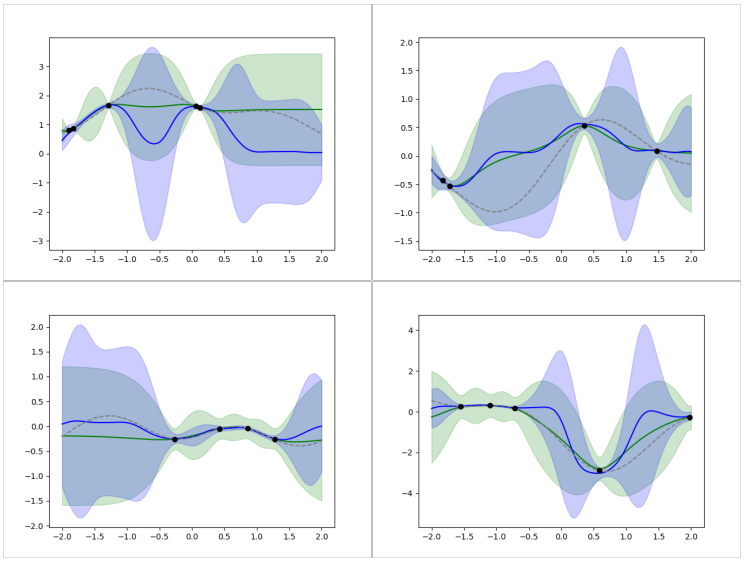


Figure 9: Predictive mean and variance of EquivCNP that using algorithm proposed in (Gordon et al., 2019). Blue line and region represents EquivCNP and green line and region represents Gaussian Process. Each plot shows diffent sampled data. Although the algorithm is redundant compared with our proposed Algorithm 1 due to using RBF kernel to map the output of LieConv back to a continuous function, the result is much smoother than Figure 2 and 8.

As we mentioned in Section C.2, the context points are sampled from bernoulli distribution. The parameter of bernoulli distribution, probability p that the value is 1, is determined at a rate of the number uniformly from $\mathcal{U}(\frac{n_{\text{total}}}{100}, \frac{n_{\text{total}}}{2})$ per n_{total}. The batch size is 4, epoch is 100, and the optimizer is Adam (Kingma & Ba, 2014) whose learning rate is 5×10^{-4}.

Figure 10: The original data that is used for 2D image-completion task.