A. Complex-Valued Non-Stationary Kernels

Beyond stationary kernels, Generalized Bochner's Theorem (Yaglom, 2012; Kakihara, 1985; Genton, 2001) presents a spectral representation for a larger class of kernels. As stated in Yaglom (2012), Generalized Bochner's Theorem applies for most bounded kernels except for some special counter-examples.

Theorem 5. *(Generalized Bochner) A complex-valued bounded continuous function k on $\mathbb{R}^d$ is the covariance function of a mean square continuous complex-valued random process on $\mathbb{R}^d$ if and only if it can be represented as*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \int_{\mathbb{R}^d \times \mathbb{R}^d} e^{2\pi i (\boldsymbol{w_1}^\top \boldsymbol{x} - \boldsymbol{w_2}^\top \boldsymbol{y})} \psi(\mathrm{d}\boldsymbol{w_1}, \mathrm{d}\boldsymbol{w_2}) \quad (12)$$

where ψ is a Lebesgue-Stieltjes measure associated with some positive semi-definite bounded symmetric function $S(\boldsymbol{w_1}, \boldsymbol{w_2})$.

We call kernels with this property *complex-valued non-stationary* (CvNs) kernels.

Actually, when the spectral measure ψ has mass concentrated along the diagonal $\boldsymbol{w_1} = \boldsymbol{w_2}$, Bochner's theorem is recovered. Similarly, for such a CvNs kernel, the closure under summation and multiplication still holds (Proof in Appendix C):

Lemma 6. *For CvNs kernels k, k_1, k_2,*

- *For $\lambda_1, \lambda_2 \in \mathbb{R}^*$, $\lambda_1 k_1 + \lambda_2 k_2$ is a CvNs kernel.*
- *Product $k_1 k_2$ is a CvNs kernel.*
- *Real part $\Re\{k\}$ is a real-valued kernel.*

Finally, the next theorem justifies the use of complex-valued kernels and real-valued kernels together in an NKN.

Theorem 7. *An NKN with primitive kernels which are either real-valued kernels or CvNs kernels has all the nodes' real parts be valid real-valued kernels.*

Proof. Assume primitive kernels have real-valued kernels $r_1, \cdots, r_t$ and CvNs kernels $c_1, \cdots, c_s$. Any node in an NKN can be written as $\mathrm{Poly}_+(r_1, \cdots, r_t, c_1, \cdots, c_s)$.[3] It is sufficient to show that for every purely multiplication term in $\mathrm{Poly}_+(r_1, \cdots, r_t, c_1, \cdots, c_s)$, its real part is a valid kernel. As the real part equals $\prod_{i=1}^t r_i^{n_i} \times \Re(\prod_{j=1}^s c_j^{m_j})$ and Lemma 8 shows $\Re(\prod_{j=1}^s c_j^{m_j})$ is a valid real-valued kernel, therefore the whole real part is also a valid kernel. □

[3] Poly_+ represents a positive-weighted polynomial of primitive kernels.

B. Complex-Valued Stationary-Kernel Closure

Lemma 8. *For complex-valued stationary kernels k, k_1, k_2,*

- *For $\lambda_1, \lambda_2 \in \mathbb{R}^*$, $\lambda_1 k_1 + \lambda_2 k_2$ is a complex-valued stationary kernel.*
- *Product $k_1 k_2$ is a complex-valued stationary kernel.*
- *Real part $\Re\{k\}$ is a real-valued kernel.*

Proof. A complex-valued stationary kernel k is equivalent to being the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^P$, which also means that for any $N \in \mathbb{Z}^+$, $\{\boldsymbol{x_i}\}_{i=1}^N$, we have $\boldsymbol{K} = \{k(\boldsymbol{x_i} - \boldsymbol{x_j})\}_{0 \le i,j \le N}$ is a complex-valued positive semi-definite matrix.

Thus for complex-valued stationary kernels k_1 and k_2, and any $N \in \mathbb{Z}^+$, $\{\boldsymbol{x_i}\}_{i=1}^N$, we have complex-valued PSD matrices $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$. Then product $k = k_1 k_2$ will have matrix $\boldsymbol{K} = \boldsymbol{K}_1 \circ \boldsymbol{K}_2$. For PSD matrices, assume eigenvalue decompositions

$$\boldsymbol{K}_1 = \sum_{i=1}^N \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^* \quad \boldsymbol{K}_2 = \sum_{i=1}^N \gamma_i \boldsymbol{v}_i \boldsymbol{v}_i^* \quad (13)$$

where eigenvalues λ_i and γ_i are non-negative real numbers based on positive semi-definiteness. Then

$$\begin{aligned} \boldsymbol{K} = \boldsymbol{K}_1 \circ \boldsymbol{K}_2 &= \sum_{i=1}^N \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^* \circ \sum_{i=1}^N \gamma_i \boldsymbol{v}_i \boldsymbol{v}_i^* \\ &= \sum_{i,j=1}^N \lambda_i \gamma_j (\boldsymbol{u}_i \circ \boldsymbol{v}_j)(\boldsymbol{u}_i \circ \boldsymbol{v}_j)^* \end{aligned} \quad (14)$$

$\boldsymbol{K}$ is also a complex-valued PSD matrix, thus $\boldsymbol{K}$ is also a complex-valued stationary kernel.

For sum $k = k_1 + k_2$, $\boldsymbol{K} = \boldsymbol{K}_1 + \boldsymbol{K}_1$ is also a complex-valued PSD.

Hence we proved the closure of complex-valued kernels under summation and multiplication. For a complex-valued kernel k and arbitrarily N complex-valued numbers, as shown above

$$\begin{aligned} \boldsymbol{K} &= \sum_{k=1}^N \lambda_i \boldsymbol{u}_k \boldsymbol{u}_k^* = \sum_{k=1}^N \lambda_i (\boldsymbol{a}_k + i\boldsymbol{b}_k)(\boldsymbol{a}_k - i\boldsymbol{b}_k)^\top \\ &= \sum_{k=1}^N \lambda_i (\boldsymbol{a}_k \boldsymbol{a}_k^\top + \boldsymbol{b}_k \boldsymbol{b}_k^\top + i(\boldsymbol{b}_k \boldsymbol{a}_k^\top - \boldsymbol{a}_k \boldsymbol{b}_k^\top)) \\ &= \sum_{k=1}^N \lambda_i (\boldsymbol{a}_k \boldsymbol{a}_k^\top + \boldsymbol{b}_k \boldsymbol{b}_k^\top) + i \sum_{k=1}^N \lambda_i (\boldsymbol{b}_k \boldsymbol{a}_k^\top - \boldsymbol{a}_k \boldsymbol{b}_k^\top). \end{aligned} \quad (15)$$

Here $\boldsymbol{a}_k, \boldsymbol{b}_k$ are all real vectors. Therefore, $\Re(\boldsymbol{K})$ is a real PSD matrix, thus $\Re\{k\}$ is a real-valued stationary kernel.

Note: Kernel's summation, multiplication and real part correspond to summation, convolution and even part in the spectral domain. Because the results of applying these spectral-domain operations to a positive function are still positive, and the only requirement for spectral representation is being positive as in Theorem 1, the closure is natural. □

C. Proof for Complex-Valued Kernel Closure

This section provides proof for Lemma 6.

Proof. First, for $k(\boldsymbol{x},\boldsymbol{y})$ to be a valid kernel ($k(\boldsymbol{x},\boldsymbol{y}) = k(\boldsymbol{y},\boldsymbol{x})^*$), $S(\boldsymbol{w_1},\boldsymbol{w_2})$ has to be symmetric, which means $S(\boldsymbol{w_1},\boldsymbol{w_2}) = S(\boldsymbol{w_2},\boldsymbol{w_1})$.

$$\begin{aligned} k(\boldsymbol{y},\boldsymbol{x})^* &= [\int e^{2\pi i(\boldsymbol{w_1}^\top \boldsymbol{y} - \boldsymbol{w_2}^\top \boldsymbol{x})} S(\boldsymbol{w_1},\boldsymbol{w_2}) \mathrm{d}\boldsymbol{w_1}\mathrm{d}\boldsymbol{w_2}]* \\ &= \int e^{2\pi i(\boldsymbol{w_2}^\top \boldsymbol{x} - \boldsymbol{w_1}^\top \boldsymbol{y})} S(\boldsymbol{w_1},\boldsymbol{w_2}) \mathrm{d}\boldsymbol{w_1}\mathrm{d}\boldsymbol{w_2} \\ &= \int e^{2\pi i(\boldsymbol{w_1}^\top \boldsymbol{x} - \boldsymbol{w_2}^\top \boldsymbol{y})} S(\boldsymbol{w_2},\boldsymbol{w_1}) \mathrm{d}\boldsymbol{w_1}\mathrm{d}\boldsymbol{w_2} \end{aligned} \tag{16}$$

Compared to

$$k(\boldsymbol{x},\boldsymbol{y}) = \int e^{2\pi i(\boldsymbol{w_1}^\top \boldsymbol{x} - \boldsymbol{w_2}^\top \boldsymbol{y})} S(\boldsymbol{w_1},\boldsymbol{w_2}) \mathrm{d}\boldsymbol{w_1}\mathrm{d}\boldsymbol{w_2} \tag{17}$$

We have the requirement that $S(\boldsymbol{w_1},\boldsymbol{w_2}) = S(\boldsymbol{w_2},\boldsymbol{w_1})$.

Next we will show the correspondence between kernel operations and spectral operations. Denote $\boldsymbol{z} = \begin{bmatrix} x \\ -y \end{bmatrix}$.

$$\begin{aligned} &k_1(\boldsymbol{x},\boldsymbol{y}) + k_2(\boldsymbol{x},\boldsymbol{y}) \\ &= \int e^{2\pi i(\boldsymbol{w_1}^\top \boldsymbol{x} - \boldsymbol{w_2}^\top \boldsymbol{y})} (S_1(\boldsymbol{w_1},\boldsymbol{w_2}) + S_2(\boldsymbol{w_1},\boldsymbol{w_2})) \mathrm{d}\boldsymbol{w_1}\mathrm{d}\boldsymbol{w_2} \\ &k_1(\boldsymbol{x},\boldsymbol{y}) k_2(\boldsymbol{x},\boldsymbol{y}) \\ &= \int_{\boldsymbol{w^1}} \int_{\boldsymbol{w^2}} e^{2\pi i \boldsymbol{z}^\top \boldsymbol{w}^1} S_1(\boldsymbol{w}^1) e^{2\pi i \boldsymbol{z}^\top \boldsymbol{w}^2} S_1(\boldsymbol{w}^2) \mathrm{d}\boldsymbol{w}^1 \mathrm{d}\boldsymbol{w}^2 \\ &= \int_{\boldsymbol{w}^1} \int_{\boldsymbol{w}_2} e^{2\pi i \boldsymbol{z}^\top (\boldsymbol{w}^1 + \boldsymbol{w}^2)} S_1(\boldsymbol{w}^1) S_2(\boldsymbol{w}^2) \mathrm{d}\boldsymbol{w}^1 \mathrm{d}\boldsymbol{w}^2 \\ &= \int_{\boldsymbol{w}^1} \int_{\boldsymbol{w}} e^{2\pi i \boldsymbol{z}^\top \boldsymbol{w}} S_1(\boldsymbol{w}^1) S_2(\boldsymbol{w} - \boldsymbol{w}^1) \mathrm{d}\boldsymbol{w}^1 \mathrm{d}\boldsymbol{w} \\ &= \int_{w} e^{2\pi i \boldsymbol{z}^\top \boldsymbol{w}} \int_{\boldsymbol{w}^1} S_1(\boldsymbol{w}^1) S_2(\boldsymbol{w} - \boldsymbol{w}^1) \mathrm{d}\boldsymbol{w}^1 \mathrm{d}\boldsymbol{w} \\ &= \int_{w} e^{2\pi i \boldsymbol{z}^\top \boldsymbol{w}} S_1 * S_2 \mathrm{d}\boldsymbol{w} \end{aligned} \tag{18}$$

Where $\boldsymbol{w^1} = \begin{bmatrix} w_1^1 \\ w_2^1 \end{bmatrix}$, and $\boldsymbol{w^2} = \begin{bmatrix} w_1^2 \\ w_2^2 \end{bmatrix}$. Therefore, summation in the kernel domain corresponds to summation in the spectral domain, and multiplication in the kernel domain corresponds to convolution in the spectral domain.

For symmetric S_1 and S_2, their convolution

$$\begin{aligned} &(S_1 * S_2)(\boldsymbol{w_2},\boldsymbol{w_1}) \\ &= \int_{\boldsymbol{\tau}_1} \int_{\boldsymbol{\tau}_2} S_1(\boldsymbol{\tau}_2,\boldsymbol{\tau}_1) S_2(\boldsymbol{w}_2 - \boldsymbol{\tau}_2, \boldsymbol{w}_1 - \boldsymbol{\tau}_1) \mathrm{d}\boldsymbol{\tau}_1 \mathrm{d}\boldsymbol{\tau}_2 \\ &= \int_{\boldsymbol{\tau}_1} \int_{\boldsymbol{\tau}_2} S_1(\boldsymbol{\tau}_1,\boldsymbol{\tau}_2) S_2(\boldsymbol{w}_1 - \boldsymbol{\tau}_1, \boldsymbol{w}_2 - \boldsymbol{\tau}_2) \mathrm{d}\boldsymbol{\tau}_1 \mathrm{d}\boldsymbol{\tau}_2 \\ &= (S_1 * S_2)(\boldsymbol{w_1},\boldsymbol{w_2}) \end{aligned} \tag{19}$$

is still symmetric, and their sum is also symmetric. Therefore, the sum and multiplication of complex-valued non-stationary kernels is still a complex-valued non-stationary kernel.

Finally, decompose $S(\boldsymbol{w}_1,\boldsymbol{w}_2)$ into even and odd parts $S(\boldsymbol{w}_1,\boldsymbol{w}_2) = E(\boldsymbol{w}_1,\boldsymbol{w}_2) + O(\boldsymbol{w}_1,\boldsymbol{w}_2)$, that $E(\boldsymbol{w}_1,\boldsymbol{w}_2) = E(-\boldsymbol{w}_1,-\boldsymbol{w}_2)$, $O(\boldsymbol{w}_1,\boldsymbol{w}_2) = -O(\boldsymbol{w}_2,\boldsymbol{w}_1)$. Inherently, E and O are symmetric.
Then the part corresponded with E

$$\begin{aligned} &\int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} E(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= \frac{1}{2} \int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &+ \frac{1}{2} \int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} S(-\boldsymbol{w}_1,-\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= \frac{1}{2} \int [e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} + e^{2\pi i(-\boldsymbol{w}_1^\top \boldsymbol{x} + \boldsymbol{w}_2^\top \boldsymbol{y})}] \\ &\qquad S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= \int \cos(2\pi(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})) S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \end{aligned} \tag{20}$$

is real. The part corresponded with O

$$\begin{aligned} &\int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} O(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= \frac{1}{2} \int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &- \frac{1}{2} \int e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} S(-\boldsymbol{w}_1,-\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= \frac{1}{2} \int [e^{2\pi i(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})} - e^{2\pi i(-\boldsymbol{w}_1^\top \boldsymbol{x} + \boldsymbol{w}_2^\top \boldsymbol{y})}] \\ &\qquad S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \\ &= i \int \sin(2\pi(\boldsymbol{w}_1^\top \boldsymbol{x} - \boldsymbol{w}_2^\top \boldsymbol{y})) S(\boldsymbol{w}_1,\boldsymbol{w}_2) \mathrm{d}\boldsymbol{w}_1 \mathrm{d}\boldsymbol{w}_2 \end{aligned} \tag{21}$$

is purely imaginary. Therefore $E(\boldsymbol{w}_1,\boldsymbol{w}_2)$ corresponds to $\Re(k)$, the real part of kernel k. As E is bounded, symmetric and positive, $\Re(k)$ is a valid real-valued kernel. □

D. Proof for NKN Polynomial Universalities

This section provides proof for Theorem 3.

Proof. *For limited width* For limited width, we will demonstrate a special stucture of NKN which can represent any positive-weighted polynomial of primitive kernels. As a polynomial is summation of several multiplication terms, if NKN can represent any multiplication term and add them iteratively to the outputs, it can represent any positive-weighted polynomial of primitive kernels.

Figure 6 demonstrates an example of how to generate multiplication term $0.3k_1k_2^2$, in which k_1, k_2, k_3 represent primitive kernels and 1 represents bias. In every Linear-Product module, we add a primitive kernel to the fourth neuron on the right and multiply it with the third neuron on the right. Iteratively, we can generate the multiplication term. Finally, we can add this multiplication term to the rightmost output.

Note that, after adding the multiplication term to the output, we keep every kernel except the output kernel unchanged. Therefore, we can continue adding new multiplication terms to the output and generate the whole polynomial.

Although this structure demonstrates the example with 3 primitive kernel, it can easily extend to deal with any number of primitive kernels as can be seen. What's more, as a special case, this structure only uses specific edges of a NKN. In practice, NKN can be more flexible and efficient in representing kernels.

For limited depth, the example can be proved by recursion.

As a Lemma, we firstly prove, a NKN with one primitive kernel k_0 and p Linear-Product modules, can use width $2(p+1)$ to represent $k_0^q, \forall q \leq 2^p$.

It is obvious that NKN can use width 2 to represent $k_0^{2^t}, \forall t \leq p$. Then we can write $q = \sum_{t=0}^{p} q_t 2^t$ with $q_t \in \{0, 1\}$. Therefore, in order to represent k_0^q, we can represent these terms separately, using width $2(p+1)$. Lemma proved.

Now we start the recusion proof,

If $B = 1$, according to the lemma above, we can use width $2(p+1) \leq 2^{Bp+1}$. The theorem also holds easily when $p = 1$.

Now assume the statement holds for in cases less than B. We can write the polynomial according to the degree of k_0, which ranges from 0 to 2^p. According to the lemma, these degrees of k_0 can be represented with width $2(p+1)$. And according to the recusion assumption, the width we need is at most

$$2^{p-1}(2p+2^{(B-1)(p-1)+1})+(2^{p-1}+1)(2(p+1)+2^{(B-1)p+1})$$

Where the first term corresponds to degree of k_0 larger than 2^{p-1} and the second term corresponds to degree of k_0 no larger than 2^{p-1}. The formula above is less than 2^{Bp+1}. According to the recursion, the theorem is proved.

□

E. Proof for Example 1

This section provides proof for Example 1.

Proof. As all dimensions are independent, we only need to prove for the one-dimensional case.

We consider these stationary kernels in spectral domain. Then cosine kernel $\cos(\mu\tau)$ corresponds to $\delta(w-\mu) + \delta(w+\mu)$ as spectral density. Note that summation and multiplication in kernel domain correspond to summation and convolution in spectral domain, respectively.

The example can be proved by recursion.

If $n = 0$, the conclusion is obvious.

Now assume the conclusion holds when less than n.

Consider case n,

Then spectral density of target kernel k^* will be

$$f^* = \sum_{t=1}^{n+1} \binom{n}{2}^{2t} (\delta(w - 4^t) + \delta(w + 4^t))$$

Assume there exists a PWP $\bar{k}$ of cosine kernels $\{\cos(\mu_t\tau)\}_{t=1}^{n}$ that have

$$\max_{\tau} |\bar{k}(\tau) - k^*(\tau)| < \epsilon$$

Then we can find $\xi > 0$ such that

$$\max_{w} |\bar{f}(w) - f^*(w)| < \xi$$

Because $\bar{k}$ is a PWP of cosine kernels. Its spectral density will be summations of convolutions of even delta funtions $\delta(w-\mu) + \delta(w+\mu)$. As convolution $\delta(w - w_1) * \delta(w - w_2) = \delta(w - (w_1 + w_2))$, $\bar{f}$ can be represented as summations of delta functions, where location in every delta function has form $\sum_{t=1}^{n} p_t\mu_t$ with $p_t \in \mathbb{Z}$ and we use $\text{coeff}(\sum_{t=1}^{n} p_t\mu_t)$ to represent the coefficient of this convolution term.

Denote the biggest location of $\bar{f}$ as $\sum_{t=1}^{n} p_t\mu_t$, then,

$$\sum_{t=1}^{n} p_t\mu_t = 4^{n+1} \tag{22}$$

If $\exists t_0$ such that $p_{t_0} < 0$, according to symmetry of $\delta(w-\mu) + \delta(w+\mu)$, there must be another term

$$\sum_{t=1}^{t_0-1} p_t\mu_t - p_t\mu_t + \sum_{t=t_0+1}^{n} p_t\mu_t$$

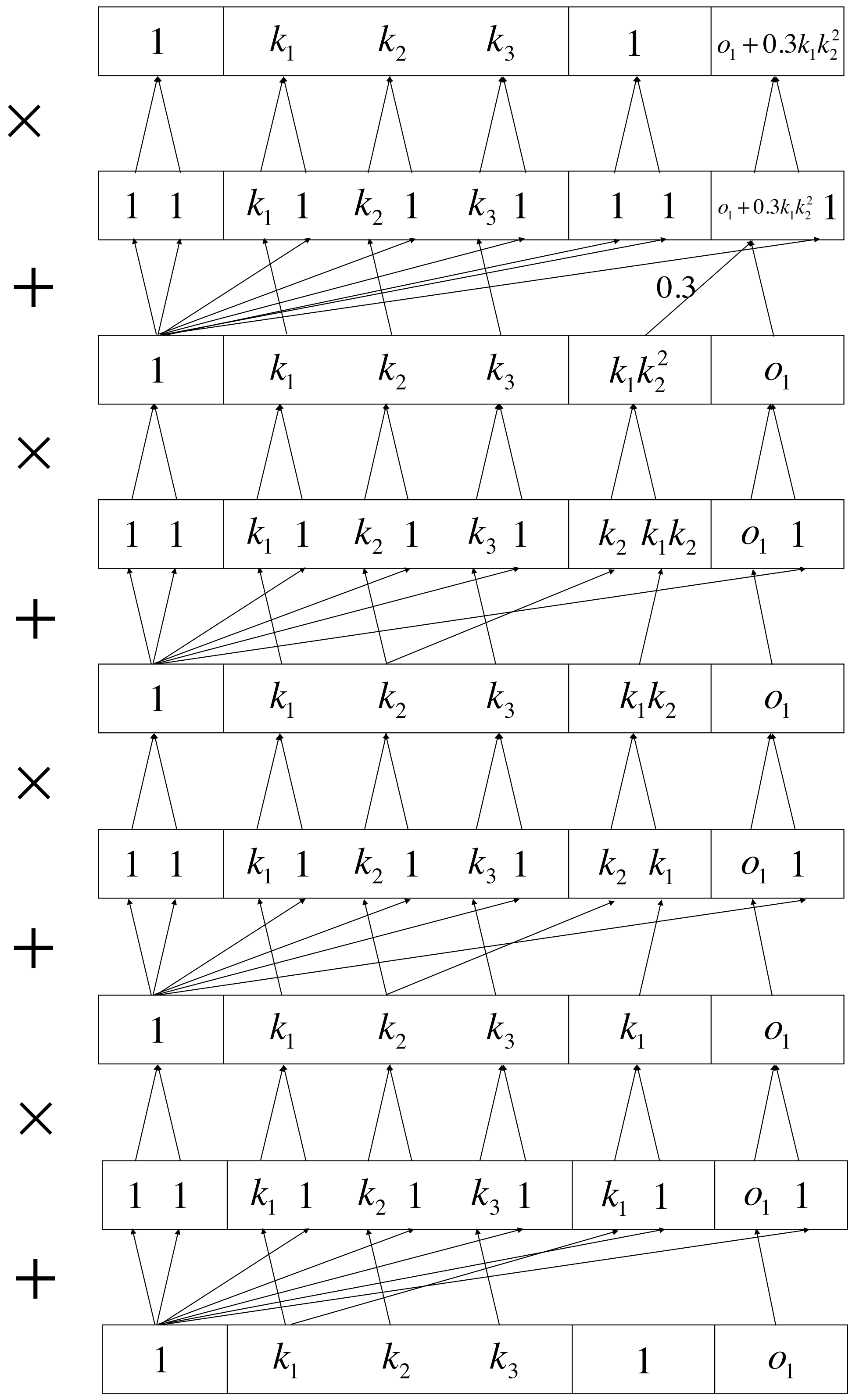

1 k_1 k_2 k_3 1 $o_1 + 0.3k_1k_2^2$
×
1 1 k_1 1 k_2 1 k_3 1 1 1 $o_1 + 0.3k_1k_2^2$ 1
+
0.3
1 k_1 k_2 k_3 $k_1k_2^2$ o_1
×
1 1 k_1 1 k_2 1 k_3 1 k_2 k_1k_2 o_1 1
+
1 k_1 k_2 k_3 k_1k_2 o_1
×
1 1 k_1 1 k_2 1 k_3 1 k_2 k_1 o_1 1
+
1 k_1 k_2 k_3 k_1 o_1
×
1 1 k_1 1 k_2 1 k_3 1 k_1 1 o_1 1
+
1 k_1 k_2 k_3 1 o_1

Figure 6. A building block of NKN to represent any positive-weighed polynomial of primitive kernels. This structure shows how NKN can add a multiplication term to the output kernel as well as keeping primitive kernels unchanged.

which has the same coefficient in $\bar{f}$ as $\sum_{t=1}^{n} p_t \mu_t$. However

$$\sum_{t=1}^{t_0-1} p_{jt}\mu_{jt} - p_{jt}\mu_{jt} + \sum_{t=t_0+1}^{n} p_{jt}\mu_{jt} > \sum_{t=1}^{n} p_{jt} = 4^{n+1}$$

Contradiction!

If $\sum_{t=1}^{n} p_t \geq 3$, we can find the smallest μ and reverse its sign. For example, if the location is $2\mu_1 + \mu_2$, then according to symmetry, there must be another location $-\mu_1 + \mu_1 + \mu_2$ in f with the same coefficient. However,

$$-\mu_1 + \mu_1 + \mu_2 > \frac{1}{3}(\mu_1 + \mu_1 + \mu_2) = \frac{1}{3}4^{n+1} > 4^n$$

Contradiction.

Therefore, $\sum_{t=1}^{n} p_t \leq 2$. If one μ has degree 2, $2\mu = 4^{n+1}$. According to symmetry, there will be a term with location $-\mu + \mu = 0$, contradiction. If $\mu_{t_1} + \mu_{t_2} = 4^{n+1}$. Then there must be term $\mu_{t-1} - \mu_{t-2}$. However, the coefficient

$$\binom{n}{2}^{2n} >= \text{coeff}(\mu_{t-1} - \mu_{t-2}) = \text{coeff}(\mu_{t_1} + \mu_{t_2})$$

While there have at most $2\binom{n}{2}$ 2-item terms, thus

$$\text{coeff}(\mu_{t_1} + \mu_{t_2}) >= \frac{\binom{n}{2}^{2(n+1)}}{2\binom{n}{2}} > \binom{n}{2}^{2n}$$

Contradiction!

Therefore, there exists t such that $p_t = 1, \mu_t = 4^{n+1}$. Because 4^{n+1} is the biggest location in f^*, μ_t cannot appear in any other term of $\bar{f}$, which induces to case of $n-1$. According to induction, the statement is proved.

□

F. Proof of NKN's Stationary Universality

This section provides proof for Theorem 4.

Proof. Start at 1-d primitive kernels. For a given complex-valued stationary kernel k with Fourier transform g, according to Lemma 10, we can find $v \in \mathbb{R}, \mu \in \mathbb{R}^+$, such that $\exists v_q = n_q^1 v \in \{v, 2v, \cdots\}, \mu_q = n_q^2 \mu \in \{\pm\mu, \pm 2\mu, \cdots\}$,

$$\begin{aligned}\bar{g}(w) &= \sum_{q=1}^{Q} \lambda_q \exp(-\frac{(w-\mu_q)^2}{2v_q}) \\ &\max(|\bar{g}(w) - g(w)|) \leq \epsilon\end{aligned} \quad (23)$$

Thus the corresponding kernel

$$\begin{aligned}\bar{k}(\tau) &= \sum_{q=1}^{Q} \lambda_q \exp(\frac{1}{2}\tau^2 v_q) e^{i\mu_q x} \\ &= \sum_{q=1}^{Q} \lambda_q \exp(-\frac{1}{2}\tau^2 v)^{n_q^1} (e^{i\mu x})^{n_q^2} \\ &= \sum_{q=1, n_q^2>0}^{Q} \lambda_q \exp(-\frac{1}{2}\tau^2 v)^{n_q^1} (e^{i\mu x})^{|n_q^2|} \\ &+ \sum_{q=1, n_q^2<0}^{Q} \lambda_q \exp(-\frac{1}{2}\tau^2 v)^{n_q^1} (e^{-i\mu x})^{|n_q^2|} \\ &= \text{Poly}_+(\exp(-\frac{1}{2}\tau^2 v), e^{i\mu x}, e^{-i\mu x})\end{aligned} \quad (24)$$

Here, Poly_+ denotes a positive-weighted polynomial of primitive kernels. Therefore, any complex-valued stationary kernel can be well approximated by a positive-weighted polynomial of a RBF and two $e^{i\mu x}$. As Theorem 3 shows, an NKN with limited width can approximate any positive-weighted polynomial of primitive kernels. Thus, taking these three kernels as primitive kernels, an NKN with width no more than $2 \times 3 + 6 = 12$ can approximate any 1-dimensional complex-valued stationary kernel. To be noted, as cos is an even function, we only need two primitive kernels (RBF, $e^{i\mu x}$) (width 10) to approximate any real-valued stationary kernel.

For d dimensional inputs, Wilson & Adams (2013) provides an SM kernel which is a universal approximator for all real-valued stationary kernels.

$$k(\boldsymbol{\tau}) = \sum_{q=1}^{Q} \lambda_q \exp(-\frac{1}{2}\boldsymbol{\tau}^\top \text{diag}(\boldsymbol{v}_q)\boldsymbol{\tau}) \cos(\boldsymbol{\mu}_q^\top \boldsymbol{\tau}) \quad (25)$$

For complex-valued stationary kernels corresponding to

$$\begin{aligned}k(\boldsymbol{\tau}) &= \sum_{q=1}^{Q} \lambda_q \exp(-\frac{1}{2}\boldsymbol{\tau}^\top \text{diag}(\boldsymbol{v}_q)\boldsymbol{\tau}) e^{i\boldsymbol{\mu}_q^\top \boldsymbol{\tau}} \\ &= \sum_{q=1}^{Q} \lambda_q \prod_{j=1}^{d} \exp(-\frac{1}{2}\tau_j^2 v_{qj}) e^{i\mu_{qj}\tau_j},\end{aligned} \quad (26)$$

the formula above is a positive-weighted polynomial of 1-dimensional case. Similarly, we can approximate them well using an NKN with d RBF and $2d$ $e^{i\boldsymbol{\mu}^\top \boldsymbol{x}}$ as primitive kernels and with width no more than $6d + 6$. For a real-valued stationary kernel, we only need an NKN with d RBF and d $e^{i\boldsymbol{\mu}^\top \boldsymbol{x}}$ as primitive kernels and with width no more than $4d + 6$. □

G. NKN on Approximating Bounded Kernels

Theorem 9. *By using d RBF kernels, $2d$ exponential kernels $e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}}$ and d functions $e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}}$ that only support multiplication and always apply the same operation on $e^{-i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}}$, NKN with width no more than $8d+6$ can approximate all d-dimensional generalized spectral kernels in Kom Samo & Roberts (2015).*

Proof. This proof refers to the proof in Kom Samo & Roberts (2015).

Let $(\boldsymbol{x},\boldsymbol{y}) \to k^*(\boldsymbol{x},\boldsymbol{y})$ be a real-valued positive semi-definite, continuous, and integrable function such that $\forall \boldsymbol{x},\boldsymbol{y}, k^*(\boldsymbol{x},\boldsymbol{y}) > 0$.

k^* being integrable, it admits a Fourier transform

$$K^*(\boldsymbol{w}_1,\boldsymbol{w}_2) = \mathcal{F}(k^*)(\boldsymbol{w}_1,\boldsymbol{w}_2),$$

therefore,

$$\forall \boldsymbol{x},\boldsymbol{y}, \quad \mathcal{F}(K^*)(\boldsymbol{x},\boldsymbol{y}) = k^*(-\boldsymbol{x},-\boldsymbol{y}) > 0.$$

Hence k^* suffices Wiener's tauberian theorem, so that any integrable function on $\mathbb{R}^d \times \mathbb{R}^b$ can be approximated well with linear combinations of translations of K^*.

Let k be any continuous bounded kernel[4], according to Theorem 5,

$$k(\boldsymbol{x},\boldsymbol{y}) = \int e^{i(\boldsymbol{x}^\top\boldsymbol{w}_1-\boldsymbol{y}^\top\boldsymbol{w}_2)} f(\boldsymbol{w}_1,\boldsymbol{w}_2)\mathrm{d}\boldsymbol{w}_1\mathrm{d}\boldsymbol{w}_2.$$

Note that we can approximate f as

$$f(\boldsymbol{w}_1,\boldsymbol{w}_2) \approx \sum_{k=1}^{K} \lambda_k K^*(\boldsymbol{w}_1+\boldsymbol{w}_k^1, \boldsymbol{w}_2+\boldsymbol{w}_k^2).$$

Therefore, we have an approximator for $k(\boldsymbol{x},\boldsymbol{y})$:

$$\begin{aligned}
&\hat{k}(\boldsymbol{x},\boldsymbol{y})\\
&= \int e^{i\begin{bmatrix}\boldsymbol{x}\\-\boldsymbol{y}\end{bmatrix}^\top\begin{bmatrix}\boldsymbol{w}_1\\\boldsymbol{w}_2\end{bmatrix}} \sum_{k=1}^{K}\lambda_k K^*(\boldsymbol{w}_1+\boldsymbol{w}_k^1,\boldsymbol{w}_2+\boldsymbol{w}_k^2)\mathrm{d}\boldsymbol{w}\\
&= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y}) e^{i(\boldsymbol{x}^\top\boldsymbol{w}_k^1-\boldsymbol{y}^\top\boldsymbol{w}_k^2)}.
\end{aligned} \tag{27}$$

Where $\boldsymbol{w} = \begin{bmatrix}\boldsymbol{w_1}\\\boldsymbol{w_2}\end{bmatrix}$. Plus, for $k(\boldsymbol{x},\boldsymbol{y})$ being a kernel, we must have $k(\boldsymbol{x},\boldsymbol{y}) = k(\boldsymbol{y},\boldsymbol{x})*$, corresponding to $f(\boldsymbol{w}_1,\boldsymbol{w}_2) = f(\boldsymbol{w}_2,\boldsymbol{w}_1)$, therefore we have the universal approximator

$$\begin{aligned}
\bar{k}(\boldsymbol{x},\boldsymbol{y}) &= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y})(e^{i(\boldsymbol{x}^\top\boldsymbol{w}_k^1-\boldsymbol{y}^\top\boldsymbol{w}_k^2)} + e^{i(\boldsymbol{x}^\top\boldsymbol{w}_k^2-\boldsymbol{y}^\top\boldsymbol{w}_k^1)})\\
&= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y}) e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\frac{\boldsymbol{w}_k^1+\boldsymbol{w}_k^2}{2}}\\
&\quad (e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\frac{\boldsymbol{w}_k^1-\boldsymbol{w}_k^2}{2}} + e^{-i(\boldsymbol{x}+\boldsymbol{y})^\top\frac{\boldsymbol{w}_k^1-\boldsymbol{w}_k^2}{2}})\\
&= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y}) e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}_k}(e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_k} + e^{-i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_k})\\
&= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y})(e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}_0})^{n_k}\Re[(e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0})^{m_k}]
\end{aligned} \tag{28}$$

where $\lambda \in \mathbb{R}$ and for $k^*(\boldsymbol{x},\boldsymbol{y})$ we can use RBF. However, not every λ makes $\bar{k}$ a kernel; therefore, for practical use, we limit $\lambda \in \mathbb{R}^+$. If we only focus on one part of $\bar{k}(\boldsymbol{x},\boldsymbol{y})$, that is

$$\begin{aligned}
\bar{k}_h(\boldsymbol{x},\boldsymbol{y}) &= \sum_{k=1}^{K}\lambda_k k^*(\boldsymbol{x},\boldsymbol{y})(e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}_0})^{n_k}(e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0})^{m_k}\\
&= \mathrm{Poly}_+\{k^*(\boldsymbol{x},\boldsymbol{y}), e^{i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}_0}, e^{-i(\boldsymbol{x}-\boldsymbol{y})^\top\boldsymbol{\mu}_0},\\
&\qquad e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0}, e^{-i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0}\}
\end{aligned} \tag{29}$$

according to 3, $\bar{k}_h(\boldsymbol{x},\boldsymbol{y})$ can be represented with an NKN. In order to generate $\bar{k}(\boldsymbol{x},\boldsymbol{y})$, we only increment a class with two items $e^{i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0}, e^{-i(\boldsymbol{x}+\boldsymbol{y})^\top\boldsymbol{\nu}_0}$ and apply the same transformation to these two items each time. In order to keep the output of NKN as a kernel, we can limit this class to only be enabled for multiplication.

□

H. Discrete Gaussian Mixture Approximation

Lemma 10. *Gaussian mixtures with form*

$$G(x) = \sum_{q=1}^{Q} w_q \exp(-\frac{(x-\psi_q)^2}{2f_q}) \tag{30}$$

with $Q \in \mathbb{Z}^+, w_q, \psi, f \in \mathbb{R}^+$, and ψ_q and f_q selected discretely in $\mathcal{S}_\psi = \{\psi, \pm 2\psi, \cdots\}, \mathcal{S}_f = \{f, 2f, \cdots\}$ can approximate any positive continuous function.

Proof. For any continuous positive function $g(x)$ and $\epsilon > 0$, the Gaussian mixture approximation theorem ensures existing

$$\hat{g}(x) = \sum_{q=1}^{Q} w_q \exp(-\frac{(x-\mu_q)^2}{2v_q}) \tag{31}$$

[4] Yaglom (2012) points out that there are some exception cases that cannot be written as this, but these are specially designed and hardly ever seen in practice.

that $\max_x(|g(x) - \hat{g}(x)|) < \frac{\epsilon}{2}$,
For small enough $\mathrm{d}v$ and $\mathrm{d}\mu$,

$$\begin{aligned}
&\exp(-\frac{(x-\mu_q-\mathrm{d}\mu)^2}{2(v_q+\mathrm{d}v)})/\exp(-\frac{(x-\mu_q)^2}{2v_q}) \\
&= \exp(-\frac{v_q\mathrm{d}\mu^2 - 2v_q(x-\mu_q)\mathrm{d}\mu - (x-\mu_q)^2\mathrm{d}v}{2v_q(v_q+\mathrm{d}v)}) \\
&\approx \exp(\frac{2v(x-\mu_q)\mathrm{d}\mu + (x-\mu_q)^2\mathrm{d}v}{2v_q^2}) \\
&\approx 1 + \frac{(x-\mu_q)^2}{2v_q^2}\mathrm{d}v + \frac{v_q(x-\mu_q)}{v_q^2}\mathrm{d}\mu
\end{aligned} \tag{32}$$

Therefore,

$$\begin{aligned}
&|\exp(-\frac{(x-\mu_q-\mathrm{d}\mu)^2}{2(v_q+\mathrm{d}v)}) - \exp(-\frac{(x-\mu_q)^2}{2v_q})| \\
&\approx \exp(-\frac{(x-\mu_q)^2}{2v_q})[\frac{(x-\mu_q)^2}{2v_q^2}\mathrm{d}v + \frac{v_q(x-\mu_q)}{v_q^2}\mathrm{d}\mu] \\
&\leq \exp(-1)\frac{1}{v_q}\mathrm{d}v + \exp(-\frac{1}{2})\frac{1}{\sqrt{v_q}}\mathrm{d}\mu
\end{aligned} \tag{33}$$

Let $V := \min_q\{v_q\}$, $f := \frac{e^1 V}{2\sum_{q=1}^Q w_q}$, $\psi := \frac{e^{0.5}\sqrt{V}}{2\sum_{q=1}^Q w_q}$. Then for any v_i, μ_i, we can find $f_i \in F = \{f, 2f, \cdots\}$, $\psi_i \in \Psi = \{\psi, \pm 2\psi, \cdots\}$ such that $|v_i - f_i| < \frac{f}{2}$, $|\mu_i - \psi_i| < \frac{\psi}{2}$, then define discrete Gaussian mixture

$$\bar{g}(x) := \sum_{q=1}^{Q} w_q \exp(-\frac{(x-\psi_q)^2}{2f_q}) \tag{34}$$

According to triangular inequality,

$$\begin{aligned}
&|\bar{g}(x) - g(x)| \leq |\bar{g}(x) - \hat{g}(x)| + |\hat{g}(x) - g(x)| \\
&< \sum_{q=1}^{Q} w_q \frac{1}{e^1 V}\frac{f}{2} + \sum_{q=1}^{Q} w_q \frac{1}{e^{0.5}\sqrt{V}}\frac{\psi}{2} + \frac{\epsilon}{2} \\
&< \frac{\epsilon}{4} + \frac{\epsilon}{4} + \frac{\epsilon}{2} = \epsilon
\end{aligned} \tag{35}$$

□

I. Additional Results

I.1. Synthetic Data

We first considered a synthetic regression dataset for illustrative purposes. Specifically, we randomly sampled a function from a GP with mean 0 and a sum of RBF and periodic kernels. We randomly selected 100 training points from $[-12, 0] \cup [6, 14]$.

As shown in Figure 3, the NKN is able to fit the training data, and its extrapolations are consistent with the ground truth function. We also observe that the NKN assigns small predictive variance near the training data and larger predictive variance far away from it.

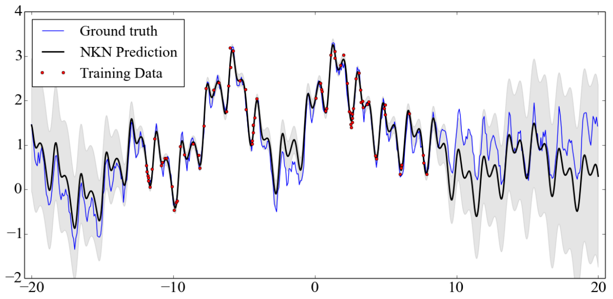


Figure 7. Synthesized 1-d toy experiment. We compare NKN prediction with the ground truth function. Shaded area represents the standard deviation of NKN prediction.

Table 3. Time (seconds) in 1D Time-series experiments.

DATASETS	AIRLINE	MAUNA	SOLAR
AS	6147	51065	37716
NKN	201	576	962

I.2. 1D Time-Series

Figure 8 and Figure 9 plot the 1D Time-Series extrapolations result, along with Figure 3. For these experiments, the running time of AS and NKN are shown in 3.

I.3. Neuron Pattern Analysis

We perform another synthetic toy experiment to analyze individual neurons' prediction patterns in NKN. The kernel for generating random functions is LIN + RBF $*$ PER. We use NKN with LIN, PER, and RBF as primitive kernels and the following layers organized as Linear4–Product2–Linear1. The prediction result is shown in Figure 10. As we can see, NKN fits the data well with plausible patterns. Beyond that, we also visualize the pattern learned by each neuron in first layer of NKN in Figure 11.

As shown in Figure 11, these four neurons show different patterns in terms of frequencies. The third neuron learns the highest frequency and the first neuron learns the lowest frequency. This shows that NKN can automatically learn basis with low correlations.

The question of how to steer the NKN framework to generate more interpretable patterns is an interesting subject for future investigation.

I.4. Kernel Recovery and Interpretability

As we proved in the paper theoretically, NKN is capable to represent sophisticated combinations of primitive kernels. We perform another experiment analyzing this kernel recovery ability in practice.

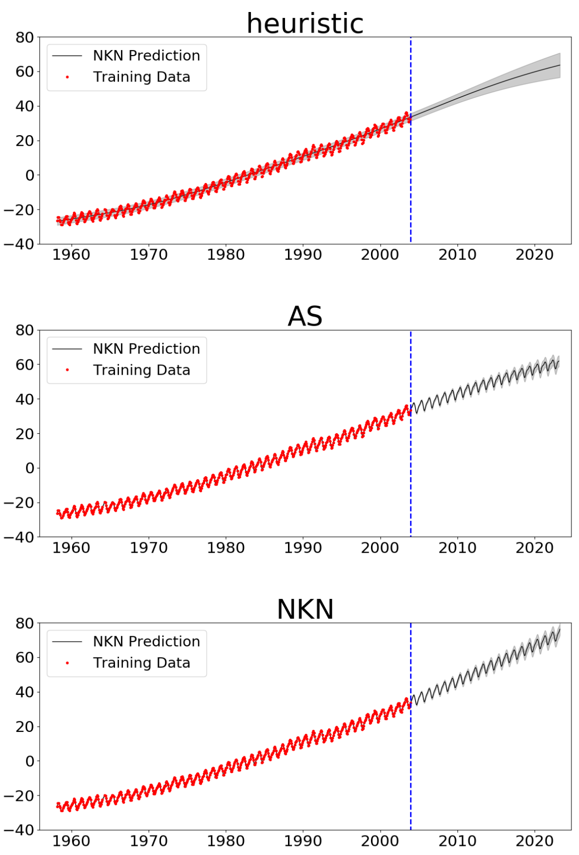


Figure 8. Extrapolation results of NKN on the Mauna datasets. Heuristic kernel represents linear combination of RBF, PER, LIN, and Constant kernels. AS represents Automatic Statistician (Duvenaud et al., 2013). The **red circles** are the training points, and the curve after the **blue dashed line** is the extrapolation result. Shaded areas represent 1 standard deviation.

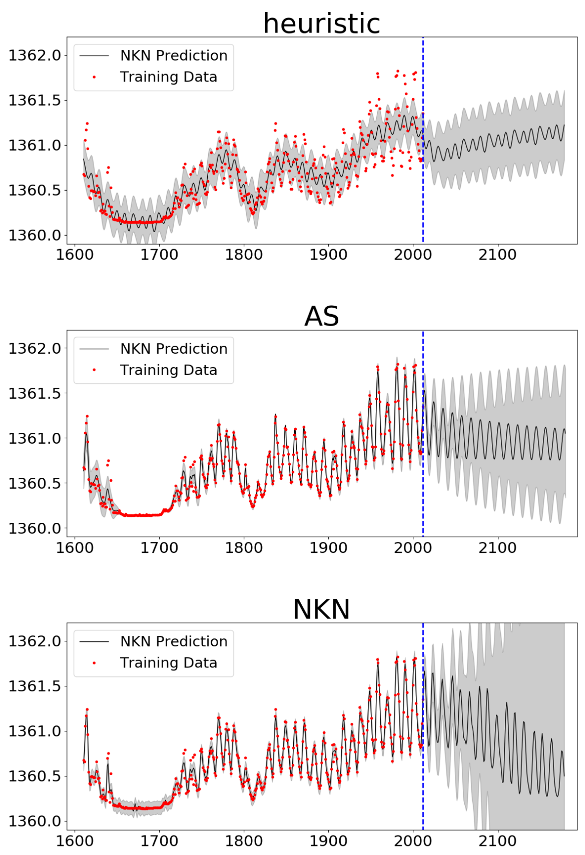


Figure 9. Extrapolation results of NKN on the Solar datasets. Heuristic kernel represents linear combination of RBF, PER, LIN, and Constant kernels. AS represents Automatic Statistician (Duvenaud et al., 2013). The **red circles** are the training points, and the curve after the **blue dashed line** is the extrapolation result. Shaded areas represent 1 standard deviation.

The experiment settings are the same as Sec 5.3, that we use individual dimensional RBF as primitive kernels of a NKN to fit the additive function. We set input dimension as 10 and use 100 data points. Ideally, a well-performing model will learn the final kernel as summation of these additive groups. After training NKN for 20,000 iterations, we print 20 terms with biggest coefficients of the overall kernel polynomial. Here $k0, k1 \cdots, k9$ represents the 10 primitive RBF kernels, respectively.

For fully additive Stybtang function, the 20 biggest terms in the final NKN kernel polynomial is

1, k6, k9, k5, k1, k2, k3, k0, k8, k4, k7, k2*k8, k6*k9, k5*k6, k1*k6, k5*k9, k1*k9, k1*k5, k6**2, k3*k9

For group additive Stybtang-transformed function with $k0, k1, k2, k3, k4$ as one group and $k5, k6, k7, k8, k9$ as one group, the 20 biggest terms in the final NKN kernel polynomial is

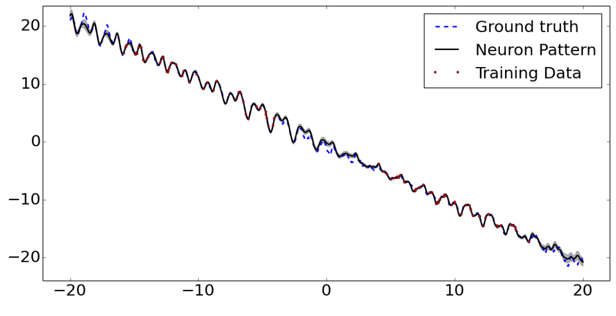
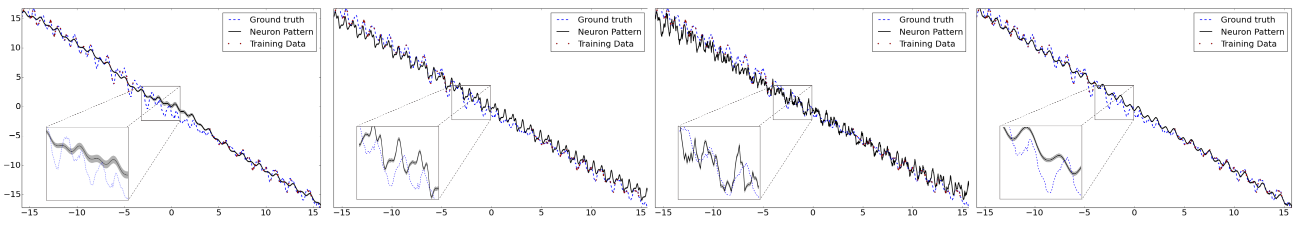


Figure 10. Synthesized 1-*d* toy experiment.

k1*k3*k4*k7, k2*k5*k7*k8, k5*k7*k8*k9, k1*k3*k4**2, k5*k6*k7*k8, k3**2*k4*k7, k1**2*k4*k7, k2*k7**2*k9, k2*k6*k7**2, k3**2*k4**2, k1**2*k4**2, k6*k7**2*k9, k1*k3*k7**2, k5**2*k8**2, k2**2*k7**2, k3*k4*k7**2,



(a) The first neuron (b) The second neuron (c) The third neuron (d) The fourth neuron

Figure 11. The visualization of learned pattern of each neuron in the first layer of NKN.

k1*k4*k7**2, k7**2*k9**2, k3**2*k7**2, k3*k4**2*k7

We can see, for Stybtang function, the biggest ones are exactly all the linear terms. This shows NKN learns the additive structure from 100 data points in this 10 dimensional space. For group additive Stybtang-transformed function, except $k7$ appears in all terms, the kernels within the same group basically appears in the same term. This demonstrates again that NKN recovers the underlying data structure.

Michalewicz function is more complicated with steep valleys. The final polynomial with 100 data points didn't show clear patterns. Therefore we change to model it with 500 data points. The 20 biggest terms are shown below,

k2*k4, k0*k9, k2*k8, k4*k7, k2*k3, k1, k3*k7, k7*k8, k0*k7, k3*k4, k6, k9**2, k7, k0*k2, k3*k8, k3*k9, k3**2, k2, k1*k5, k5*k6

We can see, although the biggest terms are not linear terms, all of them are either linear terms or quadratic terms. Note that for this NKN architecture with 2 Linear-Product modules, most common terms are fourth power. Therefore, this polynomial can show that this function is of low-correlation across dimensions to some degree.

Note that although NKN can produce sensible extrapolations, it is less interpretable compared to AS. However, as shown above, by inspecting the final kernel polynomial, we can indeed find interpretable information about the data distribution. Probably, combing with some clustering algorithms, this information recovery can become more Automatic. What's more, NKN's fast speed makes it possible to try different primitive kernel and network structure configurations, which might offer interpretable information from different aspects. This can be an interesting future research topic.

I.5. Bayesian Optimization

We perform another Bayesian Optimization experiment compared to Sec 5.3. Sec 5.3 models the function but optimizes function for all dimensions together, which doesn't take advantage of the additive strucutre. However, because additive kernels correspond to additive functions, we can optimize the function within each additive groups and combine them together, which is much more sample efficiency (Kandasamy et al., 2015; Gardner et al., 2017; Wang et al., 2017).

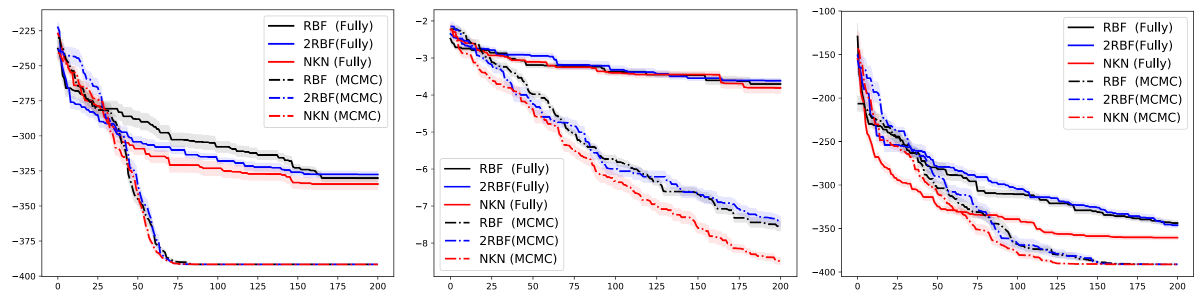
In this experiment, we adopt a small NKN network which takes 2 RBF as primitive kernels, following layers arranged as Linear4-Product2-Linear1. We compare NKN with standard RBF and RBF + RBF, which we denote as 2RBF.

We compare these kernels on two models. The fully dependent model uses a single d-dimensional kernel for fitting, which is the standard approach in Bayesian optimization. The Model MCMC (Gardner et al., 2017) samples partitions using MCMC from the posterior. We plot the optimization results in Figure 12. Each curve shows the mean cumulative minimum function evaluation up to the current iteration across 10 runs, while the shaded region shows 0.2 the standard deviation.

We found that for the fully dependent model, NKN distinctly outperforms RBF and 2RBF in terms of not only the minimum found but also convergence speed for Styblinski and Styblinski-transformed. This is because the fully dependent algorithm models correlations between all dimensions, which is beyond the capacity of RBF and 2RBF. Though NKN is expressive enough to model above two functions well, it fails to model Michalewicz function well. Therefore, we switch to use Model MCMC which can explore the additive structure of the underlying function. Stybtang task has only one local minimum, thus all three kernels perform well. In contrast, Michalewicz has many steep valleys and is much more difficult to model with simple kernels like RBF and 2RBF. Compared with Stybtang and Michalewicz, Stybtang-transformed's function introduces correlation between dimensions, which calls for a more expressive kernel structure to represent. This explains the improved convergence speed of NKN compared to the other two baselines.

J. Implementation Details

In this section we introduce the implementation details of NKN. For better analyzing NKN's parameter scale, we first list the parameter numbers for all the primitive kernels we use.



(a) Stybtang (b) Michalewicz (c) Stybtang-transform

Figure 12. Bayesian optimization on three tasks. Here 2RBF represents RBF + RBF. *Fully* and *MCMC* correspond to Fully dependent and MCMC (Gardner et al., 2017) models, respectively. Shaded error bars are 0.2 standard deviation over 10 runs.

Table 4. Number of kernel parameters for d dimensional inputs.

RBF	LIN	COS	PER	RQ
$d+1$	1	$d+1$	$2d+1$	$d+1$

J.1. Toy experiments

Across all 1-D and 2-D toy experiments, we adopt the same architecture and the same hyper-parameters initialization strategy. Concretely, we provide 8 primitive kernels (2 RBFs, 2 PERs, 2 LINs, and 2 RQs respectively) as the primitive kernels. The NKN has a Linear8-Product4-Linear4-Product2-Linear1 structure. This NKN has overall 111 trainable parameters. We optimize our model using Adam with an initial learning rate of 0.001 for $20,000$ iterations.

J.2. Regression

For the regression benchmarks, we follow the standard settings of (Hernández-Lobato & Adams, 2015) for bayesian neural networks. We use the network with 1 hidden layer of 50 units for all the datasets, except for *protein* we use 1 hidden layer of 100 units. We also compare NKN with standard RBF kernel and SM kernels. For the SM kernel, we use 4 mixtures which has $8d + 12$ trainable parameters for d dimensional inputs. The NKN has 6 primitive kernels including 2 RQ, 2 RBF, and 2 LIN. The following layers organize as Linear8-Product4-Linear4-Product2-Linear1. This architecture has $4d + 85$ trainable parameters.

Because GP suffers from $O(N^3)$ computational cost, which brings up difficulties training for large datasets. Therefore, we use Variational Free Energy (VFE) (Titsias, 2009) to train the GP models for datasets with more than 2000 data points, while for the small datasets we use the vanilla GP.

J.3. Bayesian Optimization

We first introduce two standard optimization benchmarks that have fully additive structure: the Styblinski-Tang function and the Michalewicz function. The d-dimensional Styblinski-Tang function is defined as

$$\text{Stybtang}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{d} x_i^4 - 16x_i^2 + 5x_i \tag{36}$$

which obtains its global minimum at approximately at $\mathbf{x}^* \approx (-2.9, \cdots, -2.9)$ with value $-39.166d$. In practice , we limit the exploring domain within $[-4, 4]^d$ and set $d = 10$.

The d-dimensional Michalewicz function is defined as

$$\text{Michalewicz}(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i)\sin^{2m}(\frac{ix_i^2}{\pi}) \tag{37}$$

Here m controls the steepness of valleys and ridges; a larger m leads to a more difficult search. We set $m = 10, d = 10$, then the global minimum is approximately -9.66 over the domain $[0, \pi]^d$.

In addition, we extend the Styblinski-Tang function to a transformed Styblinski-Tang function that is not fully additive. We sample a random partition P and for each part i of P, we sample a random orthonormal matrix Q_i over the dimensions of part i. If Q is the block diagonal matrix formed by placing each Q_i on the diagonal, then Stybtang(Qx) is no longer fully additive, but instead is additive across the components of P. This evaluates performance of BO algorithms when the true function has corrlations between inputs dimensions.

For the Bayesian optimization process, we firsly sample 10 initial data points randomly. Then at each step, we use a GP with the proposed kernel to fit all data points available. We choose the next point from all candidate points in the exploring domain according to *expected improvement*. For sampling in Model MCMC, we perform 30 steps in the first 30 iterations, while we perform 10 steps after. Among all models and tasks, we use L-BFGS optimizer for speed. In this experiment, we use shared lengthscale for all input dimensions, in which case the trainable parameters for RBF is 2.

J.4. Texture Extrapolation

In texture extraplation, all observations don't exactly on a grid. Following the algorithm in, we complete the grid using extra W imaginary observations, $\mathbf{y}_W \sim \mathcal{N}(\mathbf{f}_W, \epsilon^{-1}\mathbf{I}), \epsilon \to 0$. In practice, ϵ is set to $1e^{-6}$. The total observation vector $\mathbf{y} = [\mathbf{y}_M, \mathbf{y}_W]^\top$ has $N = M + W$ entries.

We use preconditioned conjugate gradients (PCG) to compute $(K_N + D_N)^{-1}\mathbf{y}$, we use the preconditioning matrix $C = D_N^{-1/2}$ to solve $C^\top (K_N + D_N) C\mathbf{z} = C^\top \mathbf{y}$.

To compute the log-determinant term in marginal likelihood, we cannot efficiently decompose $K_M + D_M$ since K_M is not a Kronecker matrix, Considering

$$\log |K_M + D_M| = \sum_{i=1}^{M} \log(\lambda_i^M + \sigma^2) \approx \sum_{i=1}^{M} \log(\tilde{\lambda}_i^M + \sigma^2), \tag{38}$$

where σ is the noise standard deviation of the data. We approximate the eigenvalues λ_i^M of K_M using the eigenvalues of K_N such that $\tilde{\lambda}_i^M = \frac{M}{N}\lambda_i^N$. Since λ_i^n is the eigenvalue of a matrix with Kronecker structure, it can be computed efficiently.