A. Appendix

A.1. The EPK is a Kernel

Lemma 3.5. *The exact path kernel (EPK) is a kernel.*

Proof. We must show that the associated kernel matrix $K_{\text{EPK}} \in \mathbb{R}^{n\times n}$ defined for an arbitrary subset of data $\{x_i\}_{i=1}^M \subset X$ as $K_{\text{EPK},i,j} = \int_0^1 \langle \phi_{s,t}(x_i), \phi_{s,t}(x_j)\rangle dt$ is both symmetric and positive semi-definite.

Since the inner product on a Hilbert space $\langle \cdot, \cdot \rangle$ is symmetric and since the same mapping φ is used on the left and right, K_{EPK} is **symmetric**.

To see that K_{EPK} is **Positive Semi-Definite**, let $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n)^\top \in \mathbb{R}^n$ be any vector. We need to show that $\alpha^\top K_{\text{EPK}} \alpha \geq 0$. We have

$$\alpha^\top K_{\text{EPK}}\alpha = \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \int_0^1 \langle \phi_{s,t}(x_i), \phi_{s,t}(x_j)\rangle dt \tag{19}$$

$$= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \int_0^1 \langle \nabla_w \hat{y}_{w_s(t,x_i)}, \nabla_w \hat{y}_{w_s(t,x_j)}\rangle dt \tag{20}$$

$$= \int_0^1 \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \langle \nabla_w \hat{y}_{w_s(t,x_i)}, \nabla_w \hat{y}_{w_s(t,x_j)}\rangle dt \tag{21}$$

$$= \int_0^1 \sum_{i=1}^n \sum_{j=1}^n \langle \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)}, \alpha_j \nabla_w \hat{y}_{w_s(t,x_j)}\rangle dt \tag{22}$$

$$= \int_0^1 \langle \sum_{i=1}^n \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)}, \sum_{j=1}^n \alpha_j \nabla_w \hat{y}_{w_s(t,x_j)}\rangle dt \tag{23}$$

Re-ordering the sums so that their indices match, we have (24)

$$= \int_0^1 \left\| \sum_{i=1}^n \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)} \right\|^2 dt \tag{25}$$

$$\geq 0, \tag{26}$$

Note that this reordering does not depend on the continuity of our mapping function $\phi_{s,t}(x_i)$.

□

Remark 3 In the case that our mapping function φ is not symmetric, after re-ordering, we still yield something of the form:

$$= \int_0^1 \left\| \sum_{i=1}^n \alpha_i \nabla_w \hat{y}_{w_s(t,x_i)} \right\|^2 dt \tag{27}$$

$$\tag{28}$$

The natural asymmetric ϕ is symmetric for every non-training point, so we can partition this sum. For the non-training points, we have symmetry, so for those points we yield exactly the L^2 metric. For the remaining points, if we can pick a Lipschitz constant E along the entire gradient field, then if training steps are enough, then the integral and the discrete step side of the asymmetric kernel will necessarily have positive inner product. In practice, this Lipschitz constant will change during training and for appropriately chosen step size (smaller early in training, larger later in training) we can guarantee positive-definiteness. In particular this only needs to be checked for training points.

A.2. The EPK gives an Exact Representation

Theorem 3.6 (Exact Kernel Ensemble Representation)**.** *A model f_{w_N} trained using discrete steps matching the conditions of the exact path kernel has the following exact representation as an ensemble of N kernel machines:*

$$f_{w_N} = KE(x) := \sum_{s=1}^{N} \sum_{i=1}^{M} a_{i,s} K_{EPK}(x, x', s) + b \tag{7}$$

where

$$a_{i,s} = -\varepsilon \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \tag{8}$$

$$b = f_{w_0}(x) \tag{9}$$

Proof. Let f_w be a differentiable function parameterized by parameters w which is trained via N forward Euler steps of fixed step size ε on a training dataset X with labels Y, with initial parameters w_0 so that there is a constant b such that for every x, $f_{w_0}(x) = b$, and weights at each step $w_s : 0 \le s \le N$. Let $x \in X$ be arbitrary and within the domain of f_w for every w. For the final trained state of this model f_{w_N}, let $y = f_{w_N}(x)$.

For one step of training, we consider $y_s = f_{w_s(0)}(x)$ and $y_{s+1} = f_{w_{s+1}}(x)$. We wish to account for the change $y_{s+1} - y_s$ in terms of a gradient flow, so we must compute $\dfrac{\partial y}{dt}$ for a continuously varying parameter t. Since f is trained using forward Euler with a step size of $\varepsilon > 0$, this derivative is determined by a step of fixed size of the weights w_s to w_{s+1}. We parameterize this step in terms of the weights:

$$\frac{dw_s(t)}{dt} = (w_{s+1} - w_s) \tag{29}$$

$$\int \frac{dw_s(t)}{dt} dt = \int (w_{s+1} - w_s) dt \tag{30}$$

$$w_s(t) = w_s + t(w_{s+1} - w_s) \tag{31}$$

$$\tag{32}$$

Since f is being trained using forward Euler, across the entire training set X we can write:

$$\frac{dw_s(t)}{dt} = -\varepsilon \nabla_w L(f_{w_s(0)}(X), y_i) = -\varepsilon \sum_{j=1}^{d} \sum_{i=1}^{M} \frac{\partial L(f_{w_s(0)}(x_i), y_i)}{\partial w_j} \tag{33}$$

Applying chain rule and the above substitution, we can write

$$\frac{d\hat{y}}{dt} = \frac{df_{w_s(t)}}{dt} = \sum_{j=1}^{d} \frac{df}{\partial w_j} \frac{\partial w_j}{dt} \tag{34}$$

$$= \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \left(-\varepsilon \frac{\partial L(f_{w_s(0)}(X_T), Y_T)}{\partial w_j} \right) \tag{35}$$

$$= \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \left(-\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \frac{\partial f_{w_s(0)}(x_i)}{\partial w_j} \right) \tag{36}$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \sum_{j=1}^{d} \frac{df_{w_s(t)}(x)}{\partial w_j} \frac{df_{w_s(0)}(x_i)}{\partial w_j} \tag{37}$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \nabla_w f_{w_s(t)}(x) \cdot \nabla_w f_{w_s(0)}(x_i) \tag{38}$$

$$\tag{39}$$

Using the fundamental theorem of calculus, we can compute the change in the model's output over step s

$$y_{s+1} - y_s = \int_0^1 -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \nabla_w f_{w_s(t)}(x) \cdot \nabla_w f_{w_s(0)}(x_i) dt \tag{40}$$

$$= -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \left(\int_0^1 \nabla_w f_{w_s(t)}(x) dt \right) \cdot \nabla_w f_{w_s(0)}(x_i) \tag{41}$$

$$\tag{42}$$

For all N training steps, we have

$$\begin{aligned}
y_N &= b + \sum_{s=1}^{N} y_{s+1} - y_s \\
y_N &= b + \sum_{s=1}^{N} -\varepsilon \sum_{i=1}^{M} \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \left(\int_0^1 \nabla_w f_{w_s(t)}(x) dt \right) \cdot \nabla_w f_{w_s(0)}(x_i) \\
&= b + \sum_{i=1}^{M} \sum_{s=1}^{N} -\varepsilon \frac{dL(f_{w_s(0)}(x_i), y_i)}{df_{w_s(0)}(x_i)} \int_0^1 \left\langle \nabla_w f_{w_s(t,x)}(x), \nabla_w f_{w_s(t,x_i)}(x_i) \right\rangle dt \\
&= b + \sum_{i=1}^{M} \sum_{s=1}^{N} a_{i,s} \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle dt
\end{aligned}$$

Since an integral of a symmetric positive semi-definite function is still symmetric and positive-definite, each step is thus represented by a kernel machine.

□

A.3. When is an Ensemble of Kernel Machines itself a Kernel Machine?

Here we investigate when our derived ensemble of kernel machines composes to a single kernel machine. In order to show that a linear combination of kernels also equates to a kernel it is sufficient to show that $a_{i,s} = a_{i,0}$ for all $a_{i,s}$. The a_i terms in our kernel machine are determined by the gradient the training loss function. This statement then implies that the gradient of the loss term must be constant throughout training in order to form a kernel. Here we show that this is the case when we consider a log softmax activation on the final layer and a negative log likelihood loss function.

Proof. Assume a two class problem. In the case of a function with multiple outputs, we consider each output to be a kernel. We define our network output $\hat{y}_i$ as all layers up to and including the log softmax and y_i is a one-hot encoded vector.

$$L(\hat{y}_i, y_i) = \sum_{k=1}^{K} -y_i^k (\hat{y}_i^k) \tag{43}$$

$$\tag{44}$$

For a given output indexed by k, if $y_i^k = 1$ then we have

$$L(\hat{y}_i, y_i) = -1(\hat{y}_i^k) \tag{45}$$

$$\frac{\partial L(\hat{y}_i, y_i)}{\partial \hat{y}_i} = -1 \tag{46}$$

$$\tag{47}$$

If $y_i^k = 0$ then we have

$$L(\hat{y}_i, y_i) = 0(\hat{y}_i^k) \tag{48}$$

$$\frac{\partial L(\hat{y}_i, y_i)}{\partial \hat{y}_i} = 0 \tag{49}$$

$$\tag{50}$$

In this case, since the loss is scaled directly by the output, and the only other term is an indicator function deciding which class label to take, we get a constant gradient. This shows that the gradient of the loss function does not depend on $\hat{y}_i$. Therefore:

$$y = b - \varepsilon \sum_{i=1}^{N} \sum_{s=1}^{S} a_{i,s} \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle \, dt \tag{51}$$

$$= b - \varepsilon \sum_{i=1}^{N} a_{i,0} \sum_{s=1}^{S} \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle \, dt \tag{52}$$

This formulates a kernel machine where

$$a_{i,0} = \frac{\partial L(f_{w_0}(x_i), y_i)}{\partial f_i} \tag{53}$$

$$K(x, x_i) = \sum_{s=1}^{S} \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle \, dt \tag{54}$$

$$\phi_{s,t}(x) = \nabla_w f_{w_s(t,x)}(x) \tag{55}$$

$$w_s(t, x) = \begin{cases} w_s, x \in X_T \\ w_s(t), x \notin X_T \end{cases} \tag{56}$$

$$b = 0 \tag{57}$$

□

It is important to note that this does not hold up if we consider the log softmax function to be part of the loss instead of the network. In addition, there are loss structures which can not be rearranged to allow this property. In the simple case of linear regression, we can not disentangle the loss gradients from the kernel formulation, preventing the construction of a valid kernel. For example assume our loss is instead squared error. Our labels are continuous on $\mathbb{R}$ and our activation is the identity function.

$$L(f_i, y_i) = (y_i - f_{i,s})^2 \tag{58}$$

$$\frac{\partial L(f_i, y_i)}{\partial f_i} = 2(y_i - f_{i,s}) \tag{59}$$

This quantity is dependent on f_i and its value is changing throughout training.

In order for

$$\sum_{s=1}^{S} a_{i,s} \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle dt \tag{60}$$

to be a kernel on its own, we need it to be a positive (or negative) definite operator and symetric. In the specific case of our practical path kernel, i.e. that in $K(x, x')$ if x' happens to be equal to x_i, then positive semi-definiteness can be accounted for:

$$= \sum_{s=1}^{S} 2(y_i - f_{i,s}) \int_0^1 \langle \phi_{s,t}(x), \phi_{s,t}(x_i) \rangle dt \tag{61}$$

$$= \sum_{s=1}^{S} 2(y_i - f_{i,s}) \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \nabla_w f_{w_s(0)}(x_i) \rangle dt \tag{62}$$

$$= \sum_{s=1}^{S} 2 \left(y_i \cdot \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \nabla_w f_{w_s(0)}(x_i) \rangle dt - f_{i,s} \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \nabla_w f_{w_s(0)}(x_i) \rangle dt \right) \tag{63}$$

$$= \sum_{s=1}^{S} 2 \left(y_i \cdot \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \nabla_w f_{w_s(0)}(x_i) \rangle dt - \int_0^1 \langle \nabla_w f_{w_s(t))}(x), f_{i,s} \nabla_w f_{w_s(0)}(x_i) \rangle dt \right) \tag{64}$$

$$= \sum_{s=1}^{S} 2 \left(y_i \cdot \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \nabla_w f_{w_s(0)}(x_i) \rangle dt - \int_0^1 \langle \nabla_w f_{w_s(t))}(x), \frac{1}{2} \nabla_w (f_{w_s(0)}(x_i))^2 \rangle dt \right) \tag{65}$$

$$\tag{66}$$

Otherwise, we get the usual

$$= \sum_{s=1}^{S} 2(y_i - f_{i,s}) \int_0^1 \langle \nabla_w f_{w_s(t,x))}(x), \nabla_w f_{w_s(t,x)}(x') \rangle dt \tag{67}$$

$$\tag{68}$$

The question is two fold. One, in general theory (i.e. the lower example), can we contrive two pairs (x_1, x_1') and (x_2, x_2') that don't necessarily need to be training or test images for which this sum is positive for 1 and negative for 2. Second, in the case that we are always comparing against training images, do we get something more predictable since there is greater dependence on x_i and we get the above way of re-writing using the gradient of the square of $f(x_i)$.

However, even accounting for this by removing the sign of the loss will still produce a non-symmetric function. This limitation is more difficult to overcome.

A.4. Multi-Class Case

There are two ways of treating our loss function L for a number of classes (or number of output activations) K:

$$\text{Case 1: } L : \mathbb{R}^K \to \mathbb{R} \tag{69}$$

$$\text{Case 2: } L : \mathbb{R}^K \to \mathbb{R}^K \tag{70}$$

$$\tag{71}$$

A.4.1. Case 1 Scalar Loss

Let $L : \mathbb{R}^K \to \mathbb{R}$. We use the chain rule $D(g \circ f)(x) = Dg(f(x))Df(x)$.

Let f be a vector valued function so that $f : \mathbb{R}^D \to \mathbb{R}^K$ satisfying the conditions from [representation theorem above] with $x \in \mathbb{R}^D$ and $y_i \in \mathbb{R}^K$ for every i. We note that $\dfrac{\partial f}{\partial t}$ is a column and has shape $Kx1$ and our first chain rule can be done the old fashioned way on each row of that column:

$$\frac{df}{dt} = \sum_{j=1}^{M} \frac{df(x)}{\partial w_j} \frac{dw_j}{dt} \tag{72}$$

$$= -\varepsilon \sum_{j=1}^{M} \frac{df(x)}{\partial w_j} \sum_{i=1}^{N} \frac{\partial L(f(x_i), y_i)}{\partial w_j} \tag{73}$$

$$\text{Apply chain rule} \tag{74}$$

$$= -\varepsilon \sum_{j=1}^{M} \frac{df(x)}{\partial w_j} \sum_{i=1}^{N} \frac{\partial L(f(x_i), y_i)}{\partial f} \frac{df(x_i)}{\partial w_j} \tag{75}$$

$$\text{Let} \tag{76}$$

$$A = \frac{df(x)}{\partial w_j} \in \mathbb{R}^{K \times 1} \tag{77}$$

$$B = \frac{dL(f(x_i), y_i)}{df} \in \mathbb{R}^{1 \times K} \tag{78}$$

$$C = \frac{df(x_i)}{\partial w_j} \in \mathbb{R}^{K \times 1} \tag{79}$$

We have a matrix multiplication ABC and we wish to swap the order so somehow we can pull B out, leaving A and C to compose our product for the representation. Since $BC \in \mathbb{R}$, we have $(BC) = (BC)^T$ and we can write

$$(ABC)^T = (BC)^T A^T = BCA^T \tag{80}$$

$$ABC = (BCA^T)^T \tag{81}$$

Note: This condition needs to be checked carefully for other formulations so that we can re-order the product as follows:

$$= -\varepsilon \sum_{j=1}^{M} \sum_{i=1}^{N} \left(\frac{dL(f(x_i), y_i)}{df} \frac{df(x_i)}{\partial w_j} \left(\frac{df(x)}{\partial w_j} \right)^T \right)^T \tag{82}$$

$$= -\varepsilon \sum_{i=1}^{N} \left(\frac{dL(f(x_i), y_i)}{df} \sum_{j=1}^{M} \frac{df(x_i)}{\partial w_j} \left(\frac{df(x)}{\partial w_j} \right)^T \right)^T \tag{83}$$

$$\tag{84}$$

Note, now that we are summing over j, so we can write this as an inner product on j with the ∇ operator which in this case is computing the jacobian of f along the dimensions of class (index k) and weight (index j). We can define

$$(\nabla f(x))_{k,j} = \frac{df_k(x)}{\partial w_j} \tag{85}$$

$$= -\varepsilon \sum_{i=1}^{N} \left(\frac{dL(f(x_i), y_i)}{df} \nabla f(x_i) (\nabla f(x))^T \right)^T \tag{86}$$

$$\tag{87}$$

We note that the dimensions of each of these matrices in order are $[1, K]$, $[K, M]$, and $[M, K]$ which will yield a matrix of dimension $[1, K]$ i.e. a row vector which we then transpose to get back a column of shape $[K, 1]$. Also, we note that our kernel inner product now has shape $[K, K]$.

A.5. Schemes Other than Forward Euler (SGD)

Variable Step Size: Suppose f is being trained using Variable step sizes so that across the training set X:

$$\frac{dw_s(t)}{dt} = -\varepsilon_s \nabla_w L(f_{w_s(0)}(X), y_i) = -\varepsilon \sum_{j=1}^{d} \sum_{i=1}^{M} \frac{\partial L(f_{w_s(0)}(X), y_i)}{\partial w_j} \tag{88}$$

This additional dependence of ε on s simply forces us to keep ε inside the summation in equation 11.

Other Numerical Schemes: Suppose f is being trained using another numerical scheme so that:

$$\frac{dw_s(t)}{dt} = \varepsilon_{s,l}\nabla_w L(f_{w_s(0)}(x_i), y_i) + \varepsilon_{s-1,l}\nabla_w L(f_{w_{s-1}}(x_i), y_i) + \cdots \tag{89}$$

$$= \varepsilon_{s,l}\sum_{j=1}^{d}\sum_{i=1}^{M}\frac{\partial L(f_{w_s(0)}(x_i), y_i)}{\partial w_j} + \varepsilon_{s-1,l}\sum_{j=1}^{d}\sum_{i=1}^{M}\frac{\partial L(f_{w_{s-1}(0)}(x_i), y_i)}{\partial w_j} + \cdots \tag{90}$$

This additional dependence of ε on s and l simply results in an additional summation in equation 11. Since addition commutes through kernels, this allows separation into a separate kernel for each step contribution. Leapfrog and other first order schemes will fit this category.

Higher Order Schemes: Luckily these are intractable for for most machine-learning models because they would require introducing dependence of the kernel on input data or require drastic changes. It is an open but intractable problem to derive kernels corresponding to higher order methods.

A.6. Variance Estimation

In order to estimate variance we treat our derived kernel function K as the covariance function for Gaussian process regression. Given training data X and test data X', we can use the Kriging to write the mean prediction and it variance from true $\mu(x)$ as

$$\bar{\mu}(X') = [K(X', X)]\,[K(X, X)]^{-1}\,[Y] \tag{91}$$

$$\text{Var}(\bar{\mu}(X') - \mu(X')) = [K(X', X')] - [K(X', X)]\,[K(X, X)]^{-1}\,[K(X, X')] \tag{92}$$

$$\tag{93}$$

Where

$$[K(A, B)] = \begin{bmatrix} K(A_1, B_1) & K(A_1, B_2) & \cdots \\ K(A_2, B_1) & K(A_2, B_2) & \\ \vdots & & \ddots \end{bmatrix} \tag{94}$$

To finish our Gaussian estimation, we note that each $K(A_i, B_i)$ will me a k by k matrix where k is the number of classes. We take tr$(K(A_i, B_i)$ to determine total variance for each prediction.

Acknowledgements

This material is based upon work supported by the Department of Energy (National Nuclear Security Administration Minority Serving Institution Partnership Program's CONNECT - the COnsortium on Nuclear sECurity Technologies) DE-NA0004107. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This material is based upon work supported by the National Science Foundation under Grant No. 2134237. We would like to additionally acknowledge funding from NSF TRIPODS Award Number 1740858 and NSF RTG Applied Mathematics and Statistics for Data-Driven Discovery Award Number 1937229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.