Efficient Statistical Tests: A Neural Tangent Kernel Approach Supplementary Materials

Sheng Jia [1 2] **Ehsan Nezhadarya** [3] **Yuhuai Wu** [1 2] **Jimmy Ba** [1 2]

A. Notations and Convolutional Neural Tangent Kernel

We give a brief overview on the basics of neural tangent kernel for fully-connected neural networks and convolutional neural networks. We mostly use the same notations from Arora et al. (2019) so that the reader could refer to the original paper and their detailed derivations.

Fully-connected network: We denote the l-th pre-activation layer by $\boldsymbol{f}^{(l)}(\mathbf{x}) = \boldsymbol{W}^{(l)}\boldsymbol{h}^{(l-1)}(\mathbf{x}) \in \mathbb{R}^{d_l}$, and l-th activation layer by $\boldsymbol{h}^{(l)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{d_l}}\sigma\big(f^{(l)}(\mathbf{x})\big) \in \mathbb{R}^{d_l}$ with NTK parameterization (Jacot et al., 2018). c_σ is the standard deviation generally used for weight initializations. and d_l is the number of output neurons. A fully-connected network is then defined recursively starting from $h^{(0)}(\mathbf{x}) = \mathbf{x}$. In the large width limit, Arora et al. (2019) has shown that each pre-activation unit from $\boldsymbol{f}^{(l)}(\mathbf{x})$ is an i.i.d. gaussian process with zero mean and covariance $\Sigma^{(l-1)}$ derived recursively as following:

$$\Sigma^{(0)}(\mathbf{x},\mathbf{x}') = \mathbf{x}^\top\mathbf{x}' \tag{A.1}$$

$$\boldsymbol{\Lambda}^{(l)}(\mathbf{x},\mathbf{x}') = \begin{bmatrix}\Sigma^{(l-1)}(\mathbf{x},\mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x},\mathbf{x}') \\ \Sigma^{(l-1)}(\mathbf{x}',\mathbf{x}) & \Sigma^{(l-1)}(\mathbf{x}',\mathbf{x}')\end{bmatrix} \tag{A.2}$$

$$\Sigma^{(l)}(\mathbf{x},\mathbf{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim \mathsf{N}\left(\mathbf{0},\boldsymbol{\Lambda}^{(l)}\right)}[\sigma(u)\sigma(v)] \tag{A.3}$$

where $\Sigma^{(l)}(\mathbf{x},\mathbf{x}')$ is the covariance function for the i.i.d gaussian process for $f^{(l)}(\mathbf{x})$. The NTK expression $\Theta^{(L)}(\mathbf{x},\mathbf{x}')$ in Equation A.4 is the sum of the covariance at each layer multiplied by the product of derivative covariances from the layers above. This has been proven rigorously by Arora et al. (2019) under ReLU activations for all the layers. In our case, we will make an assumption noted in section B, and use this result.

$$\Theta^{(L)}(\mathbf{x},\mathbf{x}') = \sum_{l=1}^{L+1}\left(\Sigma^{(l-1)}(\mathbf{x},\mathbf{x}')\prod_{l'=l}^{L+1}\dot{\Sigma}^{(l')}(\mathbf{x},\mathbf{x}')\right) \tag{A.4}$$

where the derivative covariance $\dot{\Sigma}(\cdot,\cdot)$ is similar to the covariance at each pre-activation layer except using the derivative of the activation:

$$\dot{\Sigma}^{(l)}(\mathbf{x},\mathbf{x}') = c_\sigma \mathop{\mathbb{E}}_{(u,v)\sim \mathsf{N}\left(\mathbf{0},\boldsymbol{\Lambda}^{(l)}\right)}[\dot{\sigma}(u)\dot{\sigma}(v)] \tag{A.5}$$

Convolutional neural network (CNN): With a filter size q, we denote the indices of an image patch by:

$$\mathcal{D}_{ij} = \{(i+a, j+b) | -(q-1)/2 \le a, b \le (q-1)/2\}$$

We use $[\cdot]_{\mathcal{D}_{ij}}$ to denote the flattened representation of the patch centred around the image indices (i,j). This tensor is flattened over height h, width w, and channel c dimensions.

A vanilla CNN with L convolution layers and last fully-connected layer is then defined recursively with pre-activation layers

[1]University of Toronto [2]Vector Institute [3]LG Electronics. Correspondence to: Sheng Jia <sheng@cs.toronto.edu>.

$\boldsymbol{f}^{(l)}_{(\beta)}(\mathbf{x})$ and activation layers $\boldsymbol{h}^{(l)}_{(\beta)}(\mathbf{x})$, starting from $\boldsymbol{h}^{(0)}(\mathbf{x}) = \mathbf{x}$. q is the filter size, and α and β represent the incoming channel and the outgoing channel respectively:

$$\boldsymbol{f}^{(l)}_{(\beta)}(\mathbf{x}) = \sum_{\alpha=1}^{C^{(l-1)}} \boldsymbol{W}^{(l)}_{(\alpha),(\beta)} * \boldsymbol{h}^{(l-1)}_{(\alpha)}(\mathbf{x}) \tag{A.6}$$

$$\boldsymbol{h}^{(l)}_{(\beta)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{C^{(l)} \times q \times q}}\,\sigma\Big(\boldsymbol{f}^{(l)}_{(\beta)}(\mathbf{x})\Big) \tag{A.7}$$

$$\boldsymbol{f}^{(L+1)}(\mathbf{x}) = \sum_{\alpha=1}^{C^{(L)}} \langle \boldsymbol{W}^{(L+1)}_{(\alpha)}, \boldsymbol{h}^{(L)}_{(\alpha)}(\mathbf{x})\rangle \tag{A.8}$$

In the large width limit, large number of channels limit, each pre-activation covariance $\boldsymbol{\Sigma}^{(h)}$ at each layer is also defined recursively starting from $\boldsymbol{K}^{(1)}(\mathbf{x}, \mathbf{x}')$. Due to weight sharings, pre-activation neurons at different neuron indices for $\mathbf{x}$ and $\mathbf{x}'$ can still be correlated, so we keep track of the tensor $\boldsymbol{K} \in \mathbb{R}^{H\times W\times H\times W}$. We need four indices (i, j) from $\mathbf{x}$ and $(i'j')$ from $\mathbf{x}'$ to locate a specific kernel computation. Now, the covariance function $\boldsymbol{\Sigma}(\cdot, \cdot)$ can be recursively derived:

$$\Big[\boldsymbol{K}^{(0)}(\mathbf{x}, \mathbf{x}')\Big] = \mathbf{x} \otimes \mathbf{x}' \tag{A.9}$$

$$\Big[\boldsymbol{K}^{(l)}(\mathbf{x}, \mathbf{x}')\Big]_{ij,i'j'} = \frac{c_\sigma}{q^2} \mathop{\mathbb{E}}_{(u,v)\sim \mathsf{N}(\mathbf{0}, \boldsymbol{\Lambda}^{(l)}_{ij,i'j'}(\mathbf{x},\mathbf{x}'))} [\sigma(u)\sigma(v)] \tag{A.10}$$

$$\boldsymbol{\Lambda}^{(l)}_{ij,i'j'}(\mathbf{x}, \mathbf{x}') = \begin{pmatrix} \big[\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x})\big]_{ij,ij} & \big[\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}')\big]_{ij,i'j'} \\ \big[\Sigma^{(l-1)}(\mathbf{x}', \mathbf{x})\big]_{i'j',ij} & \big[\Sigma^{(l-1)}(\mathbf{x}', \mathbf{x}')\big]_{i'j',i'j'} \end{pmatrix} \tag{A.11}$$

$$\Big[\boldsymbol{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')\Big]_{ij,i'j'} = \sum_{a=-\frac{q-1}{2}}^{\frac{q-1}{2}} \sum_{b=-\frac{q-1}{2}}^{\frac{q-1}{2}} \Big[\boldsymbol{K}^{(l)}(\mathbf{x}, \mathbf{x}')\Big]_{i+\alpha,j+\beta,i'+\alpha,j'+\beta} \tag{A.12}$$

Derivative covariances are similarly derived in the same form except for replacing the activation $\sigma(\cdot)$ with its derivative $\dot{\sigma}(\cdot)$ similar to the results under fully-connected networks. Note that we can write the computation of each output pre-activation neuron in conv operation by either of following two forms:

$$\Big[\boldsymbol{f}^{(l)}_{(\beta)}(\mathbf{x})\Big]_{ij} = \langle \boldsymbol{W}^{(l)}_{(\beta)}, \Big[\boldsymbol{h}^{(l-1)}(\mathbf{x})\Big]_{\mathcal{D}_{ij}}\rangle \tag{A.13}$$

$$\Big[\boldsymbol{f}^{(l)}_{(\beta)}(\mathbf{x})\Big]_{ij} = \sum_{\alpha=1}^{C^{(L)}} \langle \boldsymbol{W}^{(l)}_{(\alpha),(\beta)}, \Big[\boldsymbol{h}^{(l-1)}_{(\alpha)}(\mathbf{x})\Big]_{\mathcal{D}_{ij}}\rangle \tag{A.14}$$

where $\big[\boldsymbol{h}^{(l-1)}(\mathbf{x})\big]_{\mathcal{D}_{ij}}$ is the flattend vector of the patch at $(l-1)$-th layer centered around (i, j). These forms will be helpful for showing the shift-invariance of the first layer covariance associated with CNTK under cosine activations.

In the large width limit, the SCNTK expression is derived by Arora et al. (2019) as :

$$K(\mathbf{x}, \mathbf{x}') = \sum_{l=1}^{L} \mathrm{Sum}\Big(\mathcal{L}\Big(\boldsymbol{K}^{(l-1)}\Big) \odot \dot{\boldsymbol{K}}^{(l)} \odot \mathcal{L}\Big(\dot{\boldsymbol{K}}^{(l+1)} \cdots \odot \cdots \mathcal{L}\Big(\mathbf{I} \odot \dot{\boldsymbol{K}}^{(L)}\Big)\Big)\Big) \tag{A.15}$$

where the linear operator $\mathcal{L}$ applies the kernel operation associated with the convolution in the equation A.12 to each element of the kernel matrix $\boldsymbol{K}^{(l)} \in \mathbb{R}^{H\times W\times H\times W}$; that is,

$$[\mathcal{L}(\boldsymbol{K})]_{ij,i'j'} = \frac{c_\sigma}{q^2} \sum_{a=-\frac{q-1}{2}}^{\frac{q-1}{2}} \sum_{b=-\frac{q-1}{2}}^{\frac{q-1}{2}} \boldsymbol{K}_{i+a,j+b,i'+a,j'+b} \tag{A.16}$$

$\mathbf{I} \in \mathbb{R}^{H\times W\times H\times W}$ is a tensor with entries $\mathbf{I}_{i,j,i',j'} = \mathbf{1}\{i = i', j = j'\}$; that is, spatially matching pixels between the intermediate features of $\mathbf{x}$ and $\mathbf{x}'$. This will ensure the patches at the same indices of two images are compared.

A.1. Side notes about NTK

In practice, we approximate the CNTK of the last fully-connected layer output with the empirical neural tangent kernel under a finite-width neural network, $K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$

$$K(\mathbf{x}, \mathbf{x}') = \left\langle \frac{\partial f(\mathbf{x}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0}, \frac{\partial f(\mathbf{x}', \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0} \right\rangle \tag{A.17}$$

Note that there are two types of kernels associated with a randomly initialized neural network. One is the covariance matrix of a GP for each pre-activation layer $l+1$. Each $l+1$-th pre-activation layer covariance matrix can be approximated by the linear kernel of random features of the activation layer l when the width goes infinity (Rahimi & Recht, 2008). This is also called NNGP kernel (Lee et al., 2017b). Another type is the tangent kernel described above, which can be written as the sum of $(L+1)$ terms with each term representing the inner product between gradients w.r.t weight matrix of one layer.

B. Discussions about the effect of using first-layer cosine activation on NTK expression

Validity of replacing weights with a fresh new sample assumption: Note that Equation A.4 is the asymptotic form of NTK $K(\mathbf{x}, \mathbf{x}) = \langle \frac{\partial f(\mathbf{x}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0}, \frac{\partial f(\mathbf{x}', \boldsymbol{\theta}_0)}{\partial \boldsymbol{\theta}_0} \rangle$ in the large width limit. This has been shown under the ReLU activations. We claim that this form can be still useful in our case with first-layer cosine activations as long as we do not include the inner products of partial derivatives w.r.t the first-layer weights $\boldsymbol{W}^{(1)}$ for our kernel.

We first note that the partial derivative w.r.t a weight matrix at layer l is (Arora et al., 2019)

$$\frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x})}{\partial \boldsymbol{W}^{(l)}} = \mathbf{b}^{(l)}(\mathbf{x}) \Big(\boldsymbol{h}^{(l-1)}(\mathbf{x}) \Big)^\top, \qquad l = 1, 2, ..., L+1 \tag{B.1}$$

$$\mathbf{b}^{(l)}(\mathbf{x}) = \sqrt{\frac{c_\sigma}{d_l}} \mathbf{D}^{(l)}(\mathbf{x}) \Big(\boldsymbol{W}^{(l+1)} \Big)^\top \mathbf{b}^{(l+1)}(\mathbf{x}) \qquad l = 1, ..., L \tag{B.2}$$

$$\mathbf{D}^{(l)}(\mathbf{x}) = diag\Big(\dot{\sigma}\Big(f^{(l)}(\mathbf{x}) \Big) \Big) \qquad l = 1, ..., L \tag{B.3}$$

With simple algebras, in the infinite width limite as $d_l \to \infty$,

$$\left\langle \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x})}{\partial \boldsymbol{W}^{(l)}}, \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x}')}{\partial \boldsymbol{W}^{(l)}} \right\rangle = \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \left\langle \sqrt{\frac{c_\sigma}{d_l}} \mathbf{D}^{(l)}(\mathbf{x}) \Big(\boldsymbol{W}^{(l+1)} \Big)^\top \mathbf{b}^{(l+1)}(\mathbf{x}), \sqrt{\frac{c_\sigma}{d_l}} \mathbf{D}^{(l)}(\mathbf{x}') \Big(\boldsymbol{W}^{(l+1)} \Big)^\top \mathbf{b}^{(l+1)}(\mathbf{x}') \right\rangle \tag{B.4}$$

Under ReLU activations for the l-th layer used in $\mathbf{D}^{(l)}(\mathbf{x})$ and **ReLU activations for the layers above**, $\boldsymbol{W}^{(l+1)}$ can be replaced with a fresh uncorrelated sample and this expression will converge (Arora et al., 2019):

$$\left\langle \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x})}{\partial \boldsymbol{W}^{(l)}}, \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x}')}{\partial \boldsymbol{W}^{(l)}} \right\rangle \to \Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \prod_{l'=l}^{L+1} \dot{\Sigma}^{(l')}(\mathbf{x}, \mathbf{x}') \qquad \forall l = 2, ..., L \tag{B.5}$$

We cannot make the same argument for $\left\langle \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x})}{\partial \boldsymbol{W}^{(1)}}, \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x}')}{\partial \boldsymbol{W}^{(1)}} \right\rangle$ due to using cosine activations but we do not use this in our SNTK or SCNTK kernels. We only use gradients inner products w.r.t weights $W^{(l)}, l \in \{2, 3, ..., L\}$ except for the first layer, so we could still directly use the result by Arora et al. (2019) for our SNTK form. Similar arguments also apply for SCNTK without using the first layer gradient inner products.

$$K_s(\mathbf{x}, \mathbf{x}') = \sum_{l=2}^{L+1} \left\langle \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x})}{\partial \boldsymbol{W}^{(l)}}, \frac{\partial f(\boldsymbol{\theta}_0, \mathbf{x}')}{\partial \boldsymbol{W}^{(l)}} \right\rangle = \sum_{l=2}^{L+1} \left(\Sigma^{(l-1)}(\mathbf{x}, \mathbf{x}') \prod_{l'=l}^{L+1} \dot{\Sigma}^{(l')}(\mathbf{x}, \mathbf{x}') \right) \tag{B.6}$$

C. Propositions: scntk is shift-invariant

We remind the proposition shown in the main paper.

Lemma 1. *If the covariance $\Sigma^{(1)}(\mathbf{x}, \mathbf{x}')$ associated with the second pre-activation layer is shift-invariant for a fully-connected network or a CNN, then the activation covariances $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}')$ and the derivative covariances $\dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')$ $\forall l = 2, ..., L$ are shift-invariant.*

Proposition 1. *In the infinite width limit, CNTK computed by the inner products of gradients w.r.t parameters for $i = 2, ..., L$ layers under first-layer cos activations is shift-invariant; $K_s(\mathbf{x}, \mathbf{x}') = K_s(\mathbf{x} - \mathbf{x}'), K_{sc}(\mathbf{x}, \mathbf{x}') = K_{sc}(\mathbf{x} - \mathbf{x}')$ if the first-layer activation $\sigma(\cdot)$ is $\sin(\cdot)$ or $\cos(\cdot)$ and the bias is uniformly sampled $w_0 \sim Unif(0, 2\pi)$*

C.1. Proof of Shift-invariant property

To show that SCNTK has a shift-invariant property, we establish results associated with the covaraince and derivative covariance at each pre-activation layers that behave as Gaussian Process (GP). This simple proof relies on taking the layer width to infinity sequentially from the first layer. Such a technique is also used in prior works that study the NNGP kernels associated with a random neural network (Lee et al., 2017a; Novak et al., 2018) One important feature is that we only need to use cosine activations for the first layer for the shift-invariance but keep the rest of activations as ReLU. This is due to the observation that most successful neural network architectures are designed and tested under the commonly used ReLU activation. In addition, using ReLU activations allow us to use the previous result by Arora et al. (2019) and get the explicit form as discussed in section B.

C.1.1. Proof part 1

Firstly, we note from Lemma 2 and Lemma 3 that both fully-connected and CNNs have shift-invariant covariance matrices $\mathbf{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')$, $\dot{\mathbf{\Sigma}}^{(l)}(\mathbf{x}, \mathbf{x}')$ for each layer as long as the first pre-activation covariance $\mathbf{\Sigma}^{(1)}(\mathbf{x}, \mathbf{x}')$ is shift-invariant.

Lemma 2. *If the covariance $\Sigma^{(1)}(\mathbf{x}, \mathbf{x}')$ associated with the second pre-activation layer is shift-invariant for a fully-connected network, then $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}')$ and $\dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')$ $\forall l = 2, ..., L$ are shift-invariant; that is, $\Sigma^{(1)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(1)}(\mathbf{x} - \mathbf{x}')$ implies $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(l)}(\mathbf{x} - \mathbf{x}')$ and $\dot{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}') = \dot{\Sigma}^{(l)}(\mathbf{x} - \mathbf{x}') \quad \forall l = 2, ..., L$*

Simple proof: *Proof by induction.*
Base case $\Sigma^{(1)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(1)}(\mathbf{x} - \mathbf{x}') \to \Sigma^{(2)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(2)}(\mathbf{x} - \mathbf{x}')$:
For the base case, we simply observe that $\mathbf{\Sigma}^{(2)}(\mathbf{x}, \mathbf{x}')$ in equation A.3 is an expectation in which the previous hidden units are sampled from GP with a shift-invariant covariance $\Lambda^{(2)}(\mathbf{x}, \mathbf{x}')$.

$$\begin{aligned}
&\because \Sigma^{(1)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(1)}(\mathbf{x} - \mathbf{x}') \\
&\therefore \mathbf{\Lambda}^{(2)}(\mathbf{x}, \mathbf{x}') = \mathbf{\Lambda}^{(2)}(\mathbf{x} - \mathbf{x}') \\
&\therefore \mathbf{\Sigma}^{(2)}(\mathbf{x}, \mathbf{x}') = c_\sigma \underset{(u,v)\sim \mathsf{N}\left(\mathbf{0}, \mathbf{\Lambda}^{(2)}(\mathbf{x}-\mathbf{x}')\right)}{\mathbb{E}} [\sigma(u)\sigma(v)] \\
&= \mathbf{\Sigma}^{(2)}(\mathbf{x} - \mathbf{x}')
\end{aligned}$$

Induction step $\Sigma^{(l)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(l)}(\mathbf{x} - \mathbf{x}') \to \Sigma^{(l+1)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(l+1)}(\mathbf{x} - \mathbf{x}')$
For the induction step, we simply follow the same step to show the shift-invariance of the next pre-activation layer GP covariance.

$$\begin{aligned}
&\because \Sigma^{(l)}(\mathbf{x}, \mathbf{x}') = \Sigma^{(l)}(\mathbf{x} - \mathbf{x}') \\
&\therefore \mathbf{\Lambda}^{(l+1)}(\mathbf{x}, \mathbf{x}') = \mathbf{\Lambda}^{(l+1)}(\mathbf{x} - \mathbf{x}') \\
&\therefore \mathbf{\Sigma}^{(l+1)}(\mathbf{x}, \mathbf{x}') = c_\sigma \underset{(u,v)\sim \mathsf{N}\left(\mathbf{0}, \mathbf{\Lambda}^{(l+1)}(\mathbf{x}-\mathbf{x}')\right)}{\mathbb{E}} [\sigma(u)\sigma(v)] \\
&\qquad = \mathbf{\Sigma}^{(l+1)}(\mathbf{x} - \mathbf{x}')
\end{aligned}$$

The same arguments can be made for the derivative covariance $\dot{\mathbf{\Sigma}}(\mathbf{x}, \mathbf{x}')$ in the equation A.5.

Lemma 3. *If the covariance $\left[\Sigma^{(1)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ associated with the second pre-activation layer is shift-invariant for a vanilla convolutional network, then the covariance functions $\left[\mathbf{\Sigma}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ and $\left[\dot{\mathbf{\Sigma}}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ $\forall l = 2, ..., L$ at indices (i, j) and (i', j') are shift-invariant. Their components $\left[\boldsymbol{K}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ and $\left[\dot{\boldsymbol{K}}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ $\forall l = 2, ..., L$ are also shift-invariant.*

Proof sketch: *Proof by induction:* Following the same step, the covariance of $\left[f^{(l+1)}(\mathbf{x})\right]_{ij}$ and $\left[f^{(l+1)}(\mathbf{x}')\right]_{ij}$ in Equation A.12 is a sum of $\left[\boldsymbol{K}^{l}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'}$ at different indices in a patch. Each correlation in Equation A.10 is again an expectation in which the covariance function $\mathbf{\Sigma}^{(l-1)}(\mathbf{x},\mathbf{x}')$ is assumed to be shift-invariant in the base case and induction step. Base case:

$$\left[\Sigma^{(1)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'} = \left[\Sigma^{(1)}(\mathbf{x}-\mathbf{x}')\right]_{ij,i'j'} \rightarrow \left[\Sigma^{(2)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'} = \left[\Sigma^{(2)}(\mathbf{x}-\mathbf{x}')\right]_{ij,i'j'}$$

Induction step:

$$\left[\Sigma^{(l)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'} = \left[\Sigma^{(l)}(\mathbf{x}-\mathbf{x}')\right]_{ij,i'j'} \rightarrow \left[\Sigma^{(l+1)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'} = \left[\Sigma^{(l+1)}(\mathbf{x}-\mathbf{x}')\right]_{ij,i'j'}$$

As a result, the shift-invariance of $\left[\boldsymbol{K}^{(l)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'}$ and $\left[\dot{\boldsymbol{K}}^{(l)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'}$ are also apparent.

C.1.2. Proof part 2

To show that the use of first-layer cosine activation in Equation (4.6) of the main paper makes the first-layer GP covariance shift-invariant, we apply the result by Rahimi & Recht (2008) to a convolutional network architecture. For a fully-connected network, we rephrase the result by Rahimi & Recht (2008) in Lemma 4.

Lemma 4. *For a fully-connected network, $\Sigma^{(1)}(\mathbf{x},\mathbf{x}') = \Sigma^{(1)}(\mathbf{x}-\mathbf{x}')$ if the first-layer activation $\sigma(\cdot)$ is $\sin(\cdot)$ or $\cos(\cdot)$ and the bias is uniformly sampled $w_0 \sim Unif(0, 2\pi)$.*

In the following Lemma 5, we will see that it leads to a sum of *patch-wise* gaussian kernels (Equation **??**) for the covariance of the second layer pre-activation GP, which is shift-invariant.

Lemma 5. *For a convolutional neural network, the covariance of the second-layer pre-activation $\left[\Sigma^{(1)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'} = \left[\Sigma^{(1)}(\mathbf{x}-\mathbf{x}')\right]_{ij,i'j'}$ if the first-layer activation $\sigma(\cdot)$ is $\sin(\cdot)$ or $\cos(\cdot)$, the bias is uniformly sampled $w_0 \sim Unif(0, 2\pi)$, and indices match $\{i = i', j = j'\}$.*

Proof sketch: Note that the second pre-activation is defined as following:

$$\mathbf{x} \rightarrow \text{pre-activation } f^{(1)} \rightarrow h^{(1)} = \sigma(\cdot) \rightarrow \text{pre-activation} f^{(2)}$$

$\left[\Sigma^{(1)}(\mathbf{x},\mathbf{x}')\right]_{ij,i'j'}$ is the **covariance** of the **neuron** at (i,j) index of pre-activation layer 2 with image $\mathbf{x}$ as the input **and the neuron** at (i',j') index of pre-activation layer 2 with image $\mathbf{x}'$ as the input. To simplify the notations, we denote α-th channel, $(i+a, j+b)$ index unit u^*_α and $(i'+a, j'+b)$ index unit v^*_α for the first layer pre-activation $f^{(1)}$ as following:

$$u^*_{\alpha,i,j} = \left[\boldsymbol{W}^{(1)}_\alpha\right]^\top [\mathbf{x}]_{\mathcal{D}_{i+a,j+b}} + w_0 \quad v^*_{\alpha,i',j'} = \left[\boldsymbol{W}^{(1)}_\alpha\right]^\top [\mathbf{x}']_{\mathcal{D}_{i'+a,j'+b}} + w_0 \tag{C.1}$$

Since $\boldsymbol{W}^{(1)}_\alpha$ is a sample from $\boldsymbol{W}^{(1)} \sim \mathsf{N}(\mathbf{0},\mathbf{1})$, we have the following random variables u^*, v^*:

$$u^*_{i,j} = \left[\boldsymbol{W}^{(1)}\right]^\top [\mathbf{x}]_{\mathcal{D}_{i+a,j+b}} + w_0 \quad v^*_{i',j'} = \left[\boldsymbol{W}^{(1)}\right]^\top [\mathbf{x}']_{\mathcal{D}_{i'+a,j'+b}} + w_0 \tag{C.2}$$

Now, we can simplify the covariance function $\Sigma^{(1)}(\mathbf{x}, \mathbf{x}')$ of the pre-activation $f^{(2)}(\cdot)$ GP:

$$\begin{aligned}
\left[\Sigma^{(1)}(\mathbf{x}, \mathbf{x}')\right]_{ij,i'j'} &= \mathbb{E}\left[\left[\boldsymbol{f}^{(2)}_{(\beta)}(\mathbf{x})\right]_{ij}\left[\boldsymbol{f}^{(2)}_{(\beta)}(\mathbf{x}')\right]_{i'j'}\right] && \text{(C.3)}\\
&= \mathbb{E}\left[\left[\boldsymbol{h}^{(1)}(\mathbf{x})\right]^{\top}_{\mathcal{D}_{ij}}\left[\boldsymbol{W}^{(2)}_{(\beta)}\right]\left[\boldsymbol{W}^{(2)}_{(\beta)}\right]^{\top}\left[\boldsymbol{h}^{(1)}(\mathbf{x}')\right]_{\mathcal{D}_{i'j'}}\right] && \text{(C.4)}\\
&= \sum_{\alpha=1}^{C_1}\mathbb{E}\left[\left[\boldsymbol{h}^{(1)}_{(\alpha)}(\mathbf{x})\right]^{\top}_{\mathcal{D}_{ij}}\left[\boldsymbol{h}^{(1)}_{(\alpha)}(\mathbf{x}')\right]_{\mathcal{D}_{i'j'}}\right] && \text{(C.5)}\\
&= \frac{c_\sigma}{C_1 q^2}\sum_{\alpha=1}^{C_1}\sum_{a=-\frac{q-1}{2}}^{\frac{q-1}{2}}\sum_{b=-\frac{q-1}{2}}^{\frac{q-1}{2}}\mathbb{E}\left[\sigma\left(u^*_{(\alpha,i,j)}\right)\sigma\left(u^*_{(\alpha,i,j)}\right)\right] && \text{(C.6)}\\
&= \frac{c_\sigma}{q^2}\sum_{a=-\frac{q-1}{2}}^{\frac{q-1}{2}}\sum_{b=-\frac{q-1}{2}}^{\frac{q-1}{2}}\mathbb{E}\left[\sigma(u^*_{i,j})\sigma(v^*_{i,j})\right] && \text{(C.7)}\\
&\overset{①}{=} \frac{c_\sigma}{q^2}\sum_{a=-\frac{q-1}{2}}^{\frac{q-1}{2}}\sum_{b=-\frac{q-1}{2}}^{\frac{q-1}{2}}\exp\left(-\frac{\left\|[\mathbf{x}]_{\mathcal{D}_{i+a,j+b}} - [\mathbf{x}']_{\mathcal{D}_{i'+a,j'+b}}\right\|_2^2}{2}\right) && \text{(C.8)}\\
&= \left[\Sigma^{(1)}(\mathbf{x} - \mathbf{x}')\right]_{ij,i'j'} \qquad \text{if } i = i', j = j' && \text{(C.9)}
\end{aligned}$$

For the equality ①, we use the result from (Rahimi & Recht, 2008) to show each expectation is a *patch-wise* gaussian kernel between the image patch $[\mathbf{x}]_{\mathcal{D}_{i+a,j+b}}$ and the image patch $[\mathbf{x}']_{\mathcal{D}_{i'+a,j'+b}}$. Such a *image patch-wise* kernel is **shift-invariant w.r.t the original images** $\mathbf{x}, \mathbf{x}'$ if $i = i', j = j'$. Finally, we show the shift-invariance of SCNTK K_{sc} in the next subsection.

C.1.3. Proof part 3

Proposition 1. In the infinite width limit, CNTK computed by the inner products of gradients w.r.t parameters for $i = 2, ..., L$ layers, under the first-layer cos activation, and the uniform bias is shift-invariant; $K_{sc}(\mathbf{x}, \mathbf{x}') = K_{sc}(\mathbf{x} - \mathbf{x}')$

Proof sketch: In the infinite width limit with the number of channels going infinity, such inner products of gradients converge to the following deterministic form using the result by Arora et al. (2019) and similar arguments in section B:

$$K_{sc}(\mathbf{x}, \mathbf{x}') = \sum_{l=2}^{L}\sum_{\beta=1}^{C^{(\beta)}}\left\langle\frac{\partial f(\mathbf{x}, \boldsymbol{\theta}_0)}{\partial \boldsymbol{W}^{(l)}_{(\beta)}}\frac{\partial f(\mathbf{x}', \boldsymbol{\theta}_0)}{\partial \boldsymbol{W}^{(l)}_{(\beta)}}\right\rangle \xrightarrow[C^{\beta}\to\infty]{}$$
$$\sum_{l=2}^{L}\mathrm{Sum}\Big(\mathcal{L}\Big(\boldsymbol{K}^{(l-1)}\Big)\odot\dot{\boldsymbol{K}}^{(l)}\odot\mathcal{L}\Big(\dot{\boldsymbol{K}}^{(l+1)}\cdots\odot\cdots\mathcal{L}\Big(\mathbf{I}\odot\dot{\boldsymbol{K}}^{(L)}\Big)\Big)\Big)$$

With Lemma 5, $\boldsymbol{K}^{(1)} \in \mathbb{R}^{H\times W\times H\times W}$ is shift-invariant for tensor indices $\{i, j, i' = i, j' = j\}$, matching indices of two images $\mathbf{x}, \mathbf{x}'$. As a reminder, $\mathcal{L}$ is the linear operator that transforms $\boldsymbol{K}$ into $\boldsymbol{\Sigma}$ due to a convolution operation. Hence, by Lemma 3, we see all $\boldsymbol{K}^{(l)}, \dot{\boldsymbol{K}}^{(l)}\ \forall i = 2, ..., L$ are shift-invariant at matching indices $\{i, j, i' = i, j' = j\}$.

$$\begin{aligned}
\left[\boldsymbol{K}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,ij} &= \left[\boldsymbol{K}^{(l)}(\mathbf{x} - \mathbf{x}')\right]_{ij,ij}\\
\left[\dot{\boldsymbol{K}}^{(l)}(\mathbf{x}, \mathbf{x}')\right]_{ij,ij} &= \left[\dot{\boldsymbol{K}}^{(l)}(\mathbf{x} - \mathbf{x}')\right]_{ij,ij} \quad \forall l = 1, ..., L
\end{aligned}$$

Since $\mathbf{I}$ is the diagonal matrix with non-zero entries at matching indices $i = j, i' = j'$, non-matching entries that are not shift-invariant are masked out. Since all the operators involved are linear, we have shown that this product, SCNTK, is shift-invariant; $K_{sc}(\mathbf{x}, \mathbf{x}') = K_{sc}(\mathbf{x} - \mathbf{x}')$ Note that different strides of conv operations will not affect this shift-invariance.

D. Discussions on the characteristic property of SCNTK

We study the characterstic property of SCNTK under a simple convolution $\rightarrow$ cosine activation $\rightarrow$ fully-connected architecture. Consider a stride s and kernel size also s, $q = s$. So $\mathcal{D}_{i,j}$ represents a $s \times s$ patch centered around (i, j). If the data dimension is $H \times W \times C$, we will have HW/s^2 non-overlapping patches the convolution layer will look at due to the stride equal to the filter size. We index these patches by $k = 1, ..., HW/s^2$ and denote the data patch by $[\mathbf{x}]_k$. Now, our SCNTK expression under this one-layer convolution is simply

$$K_{sc}(\mathbf{x}, \mathbf{x}') = \text{Sum}\Big(\mathcal{L}\Big(\boldsymbol{K}^{(1)}\Big) \odot \mathcal{L}\Big(\mathbf{I} \odot \dot{\boldsymbol{K}}^{(2)}\Big)\Big) \tag{D.1}$$

$$= \sum_{\substack{i=1 \\ (i-1)\%s=0}}^{C^{(1)}} \sum_{\substack{j=1 \\ (j-1)\%s=0}}^{C^{(1)}} \exp\left(-\frac{\left\|[\mathbf{x}]_{\mathcal{D}_{i,j}} - [\mathbf{x}']_{\mathcal{D}_{i,j}}\right\|_2^2}{2}\right) \tag{D.2}$$

$$= \sum_{k=1}^{HW/s^2} \exp\left(-\frac{\|[\mathbf{x}]_k - [\mathbf{x}']_k\|_2^2}{2}\right) \tag{D.3}$$

$$= \sum_{k=1}^{HW/s^2} K_g([\mathbf{x}]_k, [\mathbf{x}']_k) \tag{D.4}$$

With this kernel expression, we can simplify and decompose $\text{MMD}^2(\mathbb{P}, \mathbb{Q})$ into HW/s^2 terms, each of which represents the MMD^2 for the data distribution of that k-th patch.

$$\therefore \text{MMD}^2(\mathbb{P}, \mathbb{Q}) = \mathbb{E}[K_{sc}(\mathbf{x}, \mathbf{x}') + K_{sc}(\mathbf{y}, \mathbf{y}') - 2K_{sc}(\mathbf{x}, \mathbf{y})] \tag{D.5}$$

$$= \sum_{k=1}^{HW/s^2} \mathbb{E}[K_g([\mathbf{x}]_k, [\mathbf{x}']_k) + K_g([\mathbf{y}]_k, [\mathbf{y}']_k) - 2K_g([\mathbf{x}]_k, [\mathbf{y}]_k)] \tag{D.6}$$

$$= \sum_{k=1}^{HW/s^2} \text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) \tag{D.7}$$

Now, we need to argue $\text{MMD}^2(\mathbb{P}, \mathbb{Q}) = 0$ iff $\mathbb{P} = \mathbb{Q}$ given that each $\text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) = 0$ iff $\mathbb{P}_k = \mathbb{Q}_k$ due to each gaussian kernel K_g being characteristic. We will assume these patches to be independent.

$$p(\mathbf{x}) = p([\mathbf{x}]_1) \cdots p([\mathbf{x}]_k) \cdots p([\mathbf{x}]_{HW/s^2})$$

$$q(\mathbf{x}) = q([\mathbf{x}]_1) \cdots q([\mathbf{x}]_k) \cdots q([\mathbf{x}]_{HW/s^2})$$

Proof on the forward direction: Since $\text{MMD}^2(\mathbb{P}, \mathbb{Q}) = \sum_{k=1}^{HW/s^2} \text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) = 0$, and $\text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) \geq 0$, it must be the case that $\text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) = 0 \quad \forall k = 1, ..., HW/s^2$. Since K_g is characteristic, we have $\mathbb{P}_k = \mathbb{Q}_k \quad \forall k = 1, ..., HW/s^2$. Now this will indeed imply that

$$p(\mathbf{x}) = p([\mathbf{x}]_1) \cdots p([\mathbf{x}]_k) \cdots p([\mathbf{x}]_{HW/s^2}) = q([\mathbf{x}]_1) \cdots q([\mathbf{x}]_k) \cdots q([\mathbf{x}]_{HW/s^2}) = q(\mathbf{x})$$

Hence, $\mathbb{P} = \mathbb{Q}$.

Proof on the reverse direction: Since $\mathbb{P} = \mathbb{Q}$, we assume that the data distributions for each patch $\mathbb{P}_k$ and $\mathbb{Q}_k$ are the same $\mathbb{P}_k = \mathbb{Q}_k$ for real world data we are interested in. Then $\text{MMD}^2(\mathbb{P}_k, \mathbb{Q}_k) = 0 \quad \forall k = 1, ..., HW/s^2$ due to K_g being characteristic. Hence, its summation $\text{MMD}^2(\mathbb{P}, \mathbb{Q}) = 0$.

Hence, we have shown that SCNTK under a one-layer convolution layer is characteristic with the assumptions that image patches are independent and same data distributions could imply same data patch distributions. Under such assumptions, we could also see that SCNTK with more than one-layer convolution layer is characteristic by observing that MMD^2 could be similarly written as a sum of MMD^2 for patches and the kernel corresponding to each MMD patch is characteristic by the composition theorem used in the main paper. The use of composition theorem has been seen in showing the characteristic of SNTK in the main paper.

E. Implementation Details

We implement our MMD-SCNTK computations on JAX and its Flax framefork with a NVIDIA P100 GPU. Note that we need to compute the per-example gradient in order to compute the unbiased MMD estimate (U-statistics). Since computing the per-example gradients using JAX requires more GPU memory than the usual sum of gradients computation provided by frameworks such as PyTorch and Tensorflow, we compute the per-example gradients for a batch size of 100 for each backpropagation. For example, this means we will separately compute the per-example gradients 8 times for $S_{\mathbb{P}}$ with 800 data.

Our comparison experiments with the dimensionality reduced methods are based on the codebase by Rabanser et al. (2019). We were able to reproduce their results so the baseline results in Figure 2 of the main manuscript and Figure 1, 2, 3, 4 are taken from their paper. Their MNIST and CIFAR10 datasets with various shifts are included in `https://github.com/steverab/failing-loudly`.

Our comparison experiments with the optimized kernel methods are based on the codebase by Liu et al. (2020). We were able to reproduce their results so the baseline results in Table 2 of the main manuscript and Table 2 are taken from their paper. Not MNIST and CIFAR10.1 datasets are included in their repo `https://github.com/fengliu90/DK-for-TST`.

We implement our own experiment setup for the outlier detection experiments in the ablation studies, Table 5 of the main paper. The result is the average AUROC score over 5 different random seeds in Table 1. A three-layer convolutional network with max-pooling and filter size of 3 is used for this ablation study.

F. Notes on experiments

F.1. Two-sample test comparisons with methods using the dimensionality reduction and a fixed kernel

We present the full comparisons between our MMD-SCNTK method and previous MMD-UAE method (Rabanser et al., 2019) on two-sample test experiments under 10 different types of dataset shifts. Each result is averaged over 5 random seeds. We use the same three-layer convolutional network architecture used in Rabanser et al. (2019) but with the first-layer cosine activations. The results are summarized in the 10 plots from Figure 1, Figure 2, Figure 3, and Figure 4. Section 5.1 in the main manuscript provides brief explanations on the experiment setup. In particular, we have MNIST and CIFAR10 experiments but the performance in the form of detection accuracy in the plots is the averaged result over MNIST and CIFAR10 experiments.

MNIST experiments: $S_{\mathbb{P}} \sim$ MNIST dataset, $S_{\mathbb{Q}} \sim$ shifted MNIST dataset

CIFAR10 experiments: $S_{\mathbb{P}} \sim$ CIFAR10 dataset, $S_{\mathbb{Q}} \sim$ shifted CIFAR10 dataset

Dataset shift: gaussian noise. Figure 1 compares the detection performance under small ($\sigma = 1$), medium ($\sigma = 2$), and large ($\sigma = 3$) scale gaussian noise injected to the samples. We could observe that two methods perform similarly when having more than 100 data samples but SCNTK slightly outperforms UAE in a small data regime with a small scale of noise injected.

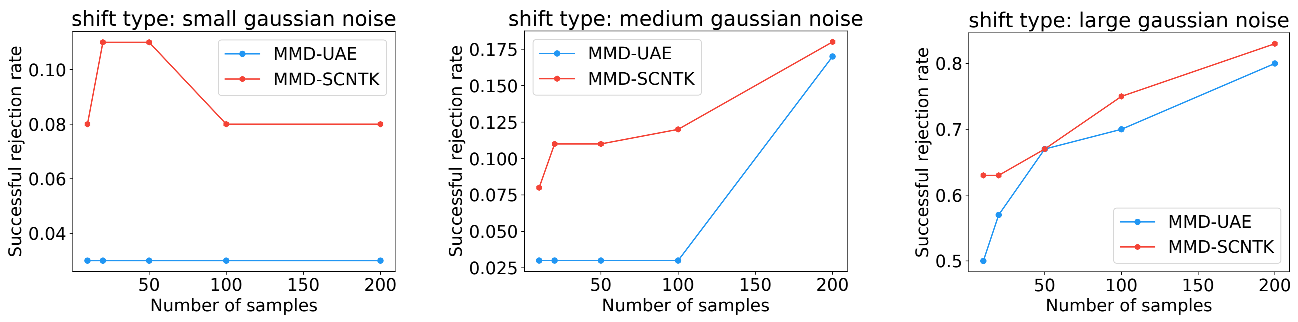

Figure 1. A fraction δ of data is corrupted by the gaussian noise with various scales $\sigma \in \{1, 10, 100\}$, which correspond to *small, medium, large* noise respectively. As the number of samples increases to 200, the gap between two methods shrinks but SCNTK still shows promising results when the number of samples and the scale of the gaussian noise is small.

Dataset shift: rigid transformation. Figure 2 shows a similar comparison but with a fraction δ percentage of the samples rotated by $\{10, 40, 90\}$, translated (by $\{0.05, 0.2, 0.4\}$), and zoomed (by$[0.1, 0.2, 0.4]$). Again, SCNTK slightly outperforms UAE under various degrees of rotations and translations applied.

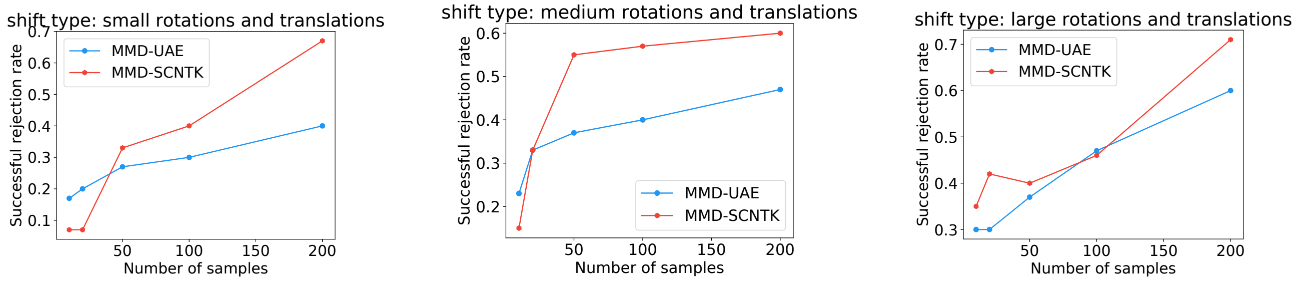

Figure 2. A fraction δ of data is corrupted by combinations of random rotations and translations, which correspond to *small, medium, large* rotations and shifts.

Dataset shift: adversarial samples or class imbalanced data. Figure 3 includes the result when adversrial samples are added (first plot) and some samples are removed for making a class imbalanced dataset (second plot).

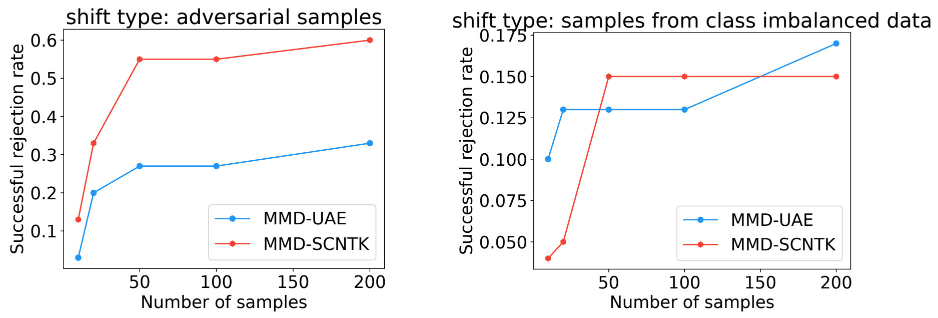

Figure 3. In the first plot, Adversarial samples are generated using the data from Rabanser et al. (2019) The second plot shows the result when some data are removed from the samples of $\mathbb{P}$, $\mathbb{Q}$ distributions.

Dataset shift: combination of gaussian noise and class imbalanced data. Figure 4 shows the comparisons of two methods under the combined effects of the dataset shifts. For the plot on the left, medium scale rotations and translations are applied along with knocking out some data from class 0 for making the class imbalanced samples. For the one on the right, only the images from class 0 are included along with a medium scaled shift to the fraction δ of the data samples.

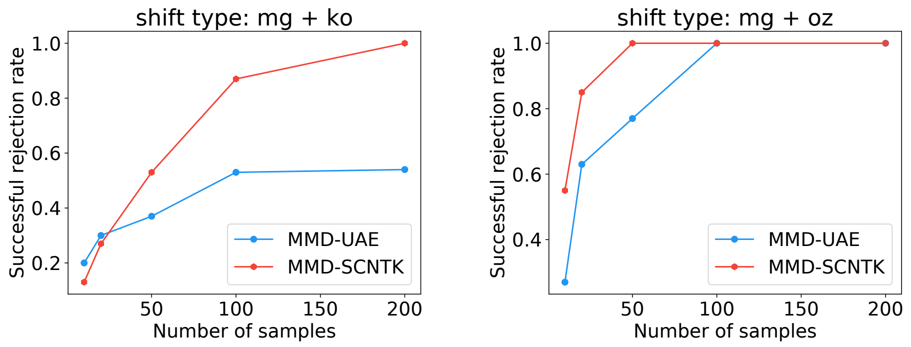

Figure 4. mg+ko represents medium scale rotations and translations previously used in Figure 2 as well as removing δ fraction of the class 0 data used in Figure 3. mg+ko only includes images from class 0 besides applying the same medium scale rotations and translations to δ fraction of the samples.

Overall, we could clearly see an improvement in terms of the detection performance, except for the shift case when the class is imbalanced, by using our gradient inner products for the kernel, which approximates the SCNTK based kernel.

F.2. Two-sample test comparisons with methods using the optimized kernel

MNIST experiments: $S_{\mathbb{P}} \sim$ MNIST dataset, $S_{\mathbb{Q}} \sim$ DCGAN MNIST dataset

CIFAR10 experiments: $S_{\mathbb{P}} \sim$ CIFAR10 dataset, $S_{\mathbb{Q}} \sim$ CIFAR10.1 dataset

Hyperparameter Choice. Our network architecture is a strided convolution network with cosine activations for the first layer and ReLU activations for the rest of layers. The strided convolution without max-pooling follows the discriminator of the DCGAN architecture (Radford et al., 2015). As Liu et al. (2020) mentions about the hyperparameter search for their training procedures using one validation set (with the same size of training set), we also perform a rough hyperparameter search for the strided convolution network architecture we are using. Our search space is rather simple and restricted in the following three items and we believe more studies in the architecture choice could further improve the results in the future.

- filter size in $\{2, 3, 4\}$
- stride size in $\{1, 2, 3, 4\}$
- number of layers in $\{3, 4\}$
- width in $\{100, 200, 300\}$ as in studying the effects of changing widths in Table 1 of the main manuscript

Only valid configurations from the combination of these choices are considered. For example, certain configurations such as having 4 convolution layers with the stride 4 is impossible if the spatial dimensions of the input data is 32 by 32. After 2 layers, the spatial dimension will be 2 by 2 in this case. Filter sizes and strides will be truncated after the 2nd layer in this case. A fully-connected layer will always be added at the last layer since the output is a scalar. Our results in Table 2 of the main manuscript use the same hyperparameters over different number of samples. For each sample size, we conduct the experiment under 10 different random seeds, and each experiment considers 100 different instances of sampled $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$. Furthermore, each permutation test is carried out by 100 different permutations of the specific sampled $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$.

A three-layer strided convolutional network is used for MNIST experiments. The same architecture and hyperparameters are used when exploring the effect of widths on the performance. A four-layer strided convoluitional network is used for CIFAR10 experiments. Widths, the number of channels, are 300 for all the convolution layers. We present other hyperparameters below.

Table 1. Hyperparameters used in MNIST and CIFAR10 experiments. A three-layer convolutional network is used for MNIST experiments and a four-layer network is used for CIFAR10 experiments.

	Stride	Filter size	activations	Weight init under NTK parameterization
MNIST experiments	$(2, 2, 2)$	3	$(cos, ReLU, ReLU)$	$(c_\sigma = 1, c_\sigma = 1, c_\sigma = 1)$
CIFAR10 experiments	$(3, 3, 3, 1)$	4	$(cos, ReLU, ReLU, ReLU)$	$(c_\sigma = 1, c_\sigma = 1, c_\sigma = 1, c_\sigma = 1)$

Table 2. Comparing the type I error, incorrectly rejecting the null hypothesis when the null hypothesis $\mathbb{P} = \mathbb{Q}$ is true.

MNIST	ME	SCF	C2ST-S	C2ST-L	M-O	M-D	SRF	**SCNTK**
200	0.076±0.011	0.075±0.010	0.035±0.006	0.045±0.005	0.068±0.004	0.056±0.003	0.062±0.009	0.055±0.011
400	0.062±0.010	0.056±0.007	0.044±0.006	0.040±0.004	0.053±0.005	0.056±0.005	0.059±0.007	0.058±0.010
600	0.051±0.003	0.049±0.009	0.039±0.005	0.054±0.007	0.066±0.008	0.056±0.008	0.048±0.010	0.058±0.006
800	0.054±0.006	0.046±0.006	0.043±0.005	0.042±0.007	0.051±0.005	0.054±0.007	0.051±0.010	0.054±0.007
1000	0.047±0.006	0.045±0.010	0.038±0.006	0.046±0.006	0.041±0.007	0.062±0.006	0.061±0.008	0.065±0.007
Avg	0.058	0.054	0.040	0.045	0.056	0.057	0.056	0.058

Type I error on MNIST In Table 2 of the main manuscript, we report the accuracy of successfully rejecting the null hypothesis h_0 : when $\mathbb{P} \neq \mathbb{Q}$. We now show the type I error for MNIST experiments as done in Liu et al. (2020), incorrectly rejecting the null hypothesis when both $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$ come from MNIST dataset. To perform this experiment, both $S_{\mathbb{P}}$ and $S_{\mathbb{Q}}$

are sampled from the same dataset MNIST. Same hyperparameters from MNIST experiments are used. In Table 2, we can see SCNTK also does not have high type I errors.

Blob and Higgs experiments
We performed two toy experiments using the Blob and Higgs dataset from Liu et al. (2020). The datapoints from these datasets do not have spatial structure like images so we do not use convolutional kernels. Blob dataset includes data points sampled from two 2D mixtures of Gaussians. For this experiment, testings with a 5-layer SNTK achieves 0.17 accuracy with 10 samples, 0.45 accuracy with 20 samples, and 1.00 accuracy with 40 or more samples. Higgs dataset includes 4-dimensional data points sampled from distributions with or without Higgs bosons (Baldi et al., 2014), We attached the results for Higgs experiments in Table 3.

Table 3. Performance of testing with a 5-layer SNTK in Higgs dataset.

# of samples	1000	2000	3000	5000	8000	10000
Higgs Acc %	$0.156_{\pm 0.021}$	$0.298_{\pm 0.031}$	$0.413_{\pm 0.025}$	$0.712_{\pm 0.018}$	$0.941_{\pm 0.011}$	$1.000_{\pm 0.000}$

The effect of increasing widths for baseline methods:
As a sanity check, we also increased the widths of convolution layers in baselines to 300 since some of our experiment results used widths 300. We include the results for classifier-based methods below.

# of samples	MNIST 200	MNIST 400	MNIST 600	MNIST 800	MNIST 1000	CIFAR10 2000
C2ST-S	$0.183_{\pm 0.038}$	$0.583_{\pm 0.052}$	$0.999_{\pm 0.001}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	0.360
C2ST-L	$0.238_{\pm 0.042}$	$0.612_{\pm 0.038}$	$1.00_{\pm 0.000}$	$1.000_{\pm 0.000}$	$1.000_{\pm 0.000}$	0.392

References

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pp. 8141–8150, 2019.

Baldi, P., Sadowski, P., and Whiteson, D. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):1–9, 2014.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017a.

Lee, K., Lee, H., Lee, K., and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325*, 2017b.

Liu, F., Xu, W., Lu, J., Zhang, G., Gretton, A., and Sutherland, D. J. Learning deep kernels for non-parametric two-sample tests. *arXiv preprint arXiv:2002.09116*, 2020.

Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Bayesian deep convolutional networks with many channels are gaussian processes. *arXiv preprint arXiv:1810.05148*, 2018.

Rabanser, S., Günnemann, S., and Lipton, Z. Failing loudly: An empirical study of methods for detecting dataset shift. In *Advances in Neural Information Processing Systems*, pp. 1396–1408, 2019.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2008.