THE HSIC BOTTLENECK: DEEP LEARNING WITHOUT BACK-PROPAGATION

Wan-Duo Kurt Ma
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
mawand@ecs.vuw.ac.nz

J.P. Lewis
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
jplewis@google.com

W. Bastiaan Kleijn
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
bastiaan.kleijn@ecs.vuw.ac.nz

August 6, 2019

ABSTRACT

We introduce the HSIC (Hilbert-Schmidt independence criterion) bottleneck for training deep neural networks. The HSIC bottleneck is an alternative to conventional backpropagation, that has a number of distinct advantages. The method facilitates parallel processing and requires significantly less operations. It does not suffer from exploding or vanishing gradients. It is biologically more plausible than backpropagation as there is no requirement for symmetric feedback. We find that the HSIC bottleneck provides a performance on the MNIST/FashionMNIST/CIFAR10 classification comparable to backpropagation with a cross-entropy target, even when the system is not encouraged to make the output resemble the classification labels. Appending a single layer trained with SGD (without backpropagation) results in state-of-the-art performance.

1 Introduction

Deep learning has brought a new level of performance to an increasingly wide range of tasks. At the same time, the error backpropagation algorithm underlying current deep learning is generally regarded as not biologically plausible [1, 2, 3]. In practice backpropagation and the associated stochastic gradient descent algorithm SGD and its variants are time consuming, have problems of vanishing and exploding gradients, require sequential computation across layers, and typically require the exploration of learning rates and other hyper-parameters. These considerations are driving research into both theoretical and practical alternatives [4].

In this paper we show that it is possible to learn classification tasks at near competitive accuracy without backpropagation, by maximizing a surrogate of the mutual information between hidden representations and labels and simultaneously minimizing the mutual dependency between hidden representations and the inputs. We further show that the hidden units of a network trained in this way form useful representations. Specifically, fully competitive accuracy can be obtained by freezing the network trained without backpropagation and appending and training a one-layer network using conventional SGD to convert convert the representation to the desired format.

We propose a deep network training method that does not require backpropagation. It consists of training the network using an approximation of the information bottleneck. Due to the difficulties of calculating the mutual information among the random variables, we adopt a non-parametric kernel-based method, the Hilbert-Schmidt independence criterion (HSIC), to characterize the statistical (in)dependence of different layers. That is, for each network layer

we simultaneously maximize HSIC between the layer and the desired output and minimize HSIC between that layer and the input. The usage of this *HSIC bottleneck* leads to fast convergence during training compared to the standard backpropagation algorithm. As the HSIC-bottleneck operates directly on continuous random variables it is more attractive than conventional information bottleneck approaches based on binning. The method uses a shallow conventionally trained post-processing network to convert the resulting representation to the form of output labels. In practice, we use a network comprised of a number of layers and with varying dimensionality (fully connected layer) or different number of kernels (convolutional-layer) as a starting point.

Our work joins an increasing body of recent research that explores deep learning fundamentals from an information theoretical perspective ([5, 6, 7, 8] and others). Our contributions are as follows: We show that it is possible to form useful representations for classification in a deep network without backpropagation, using only an information-bottleneck principle. Following HSIC bottleneck training, performance competitive with backpropagation can be obtained by appending a single classification layer trained with SGD without backpropagation. We show that the computation effort required for training with the HSIC bottleneck is significantly lower than conventional backpropagation. In some cases, the final-layer representations (without the appended classification layer) can be directly used for classification after identifying a suitable permutation. The HSIC bottleneck approach sidesteps the issue of vanishing or exploding gradients in backpropagation (though these issues may still arise in feedforward computation). Because the network training is explicitly based on an information bottleneck principle, it addresses overfitting by design. Our results were obtained with relatively little effort towards finding suitable hyperparameters and architectures, and further improvements are likely possible.

This paper is organized as follows: Section 2 surveys related work. In Section 3, we introduce the proposed supervised HSIC-trained network based on the HSIC bottleneck. In Section 4, we provide our experimental results. The experiments provide insight in how information selected by the HSIC bottleneck helps the training of the post-processing network. Section 5 contains our conclusions.

2 Background and Related Work

Although SGD using the backpropagation algorithm [9] is the predominant approach to optimizing deep neural nets, other approaches have been considered [10, 11, 12, 13, 14]. Kickback follows the local gradient using a direction obtained from the global single-class error. Feedback alignment [11] shows that deep neural networks can be trained using random feedback connections. The alternating minimization [14], a coordinate descent-like approach, breaks the nested objective into a collection of small subproblems by introducing the auxiliary variables associated to each activation which can update the layers in parallel.

Other research has explored the use of random features in a DNN context, by freezing most weights at their initial random value and only training the last layer [15, 16, 17] or a subset of other weights [18]. In some cases this results in nearly competitive performance, although a larger network may be required [17]. In the case where the hidden layers are smaller than the input this can be seen as a form of random dimensionality reduction [19], though the idea has also been employed with hidden layers that enlarge the number of features relative to the input dimensionality [20, 21, 17, 22]. For instance, training on fractional random weights in the network [21] generalizes the accuracy well. [22] uses untrained VGG network successfully changing the style of the target image to the reference image. Tangentially related, a few studies [23, 24] have shown that networks trained on different but related tasks can provide useful features. In these approaches, the previously trained network (say AlexNet [25]) is frozen and the last layer(s) are removed. A new layer is appended and trained on the new task, giving surprisingly competitive performance.

The biological plausibility of backpropagation is a subject of much debate, and is one motivation for exploring alternate approaches. One problem is that synaptic weights are adjusted based on downstream errors, which is unfeasible in a biological system [26, 27]. Another issue is that the feedforward inference and backpropagation share the same weight matrices. This is known as the weight transport problem [28, 11]. Additionally, the backpropagation gradient is computed linearly but brains have complicated neural interconnections, and backpropagation has to be halted while the feedforward is computed (and vice-versa) [1].

Information theory [29] underlies much research on learning theory [8, 30, 31]. The Information Bottleneck (IB) principle [7] generalizes the notion of minimal sufficient statistics, expressing a tradeoff in the hidden representation between the information needed for predicting the output, and the information retained about the input. The IB objective is formally written as:

$$\min_{p_{T_i|X},\, p_{Y|T_i}} I(X; T_i) - \beta I(T_i; Y), \tag{1}$$

where X,Y are the input and label random variable respectively, T_i represents the hidden representation at layer i. Note that information characterized by T_i relating to Y is extracted from X. Intuitively, the IB principal preserves the information of the hidden representations about the label while compressing information about the input data.

In practice, the IB is hard to compute for several reasons. If the network inputs are regarded as continuous, the mutual information $I(X, T_i)$ is infinite unless noise is added to the network. Many algorithms are based on binning, which does not scale to high-dimensions and yields different results with different choices of bin size. The distinction between discrete and continuous data, and between discrete and differential entropy, presents additional considerations [6, 32]. In this paper, we replace the Mutual Information terms in the Information Bottleneck objective with HSIC. In contrast to mutual information estimates, HSIC provides a robust computation with a time complexity $O(m^2)$ where m is the number of data points.

The Hilbert-Schmidt Independence Criterion (HSIC) [33] is the Hilbert-Schmidt norm of the cross-covariance operator between the distributions in Reproducing Kernel Hilbert Space (RKHS):

$$
\begin{aligned}
HSIC(\mathbb{P}_{XY}, \mathcal{H}, \mathcal{G}) &= \|C_{xy}\|^2 \\
&= \mathbb{E}_{XYX'Y'}[k_x(x, x')k_y(y, y')] \\
&+ \mathbb{E}_{XX'}[k_x(x, x')]\mathbb{E}_{YY'}[k_y(y, y')] \\
&- 2\mathbb{E}_{XY}[\mathbb{E}_{X'}[k_x(x, x')]\mathbb{E}_{Y'}[k_y(y, y')]],
\end{aligned}
\tag{2}
$$

where k_x and k_y are kernel functions, where $\mathcal{H}$ and $\mathcal{G}$ are the Hilbert spaces, and where $\mathbb{E}_{XY}$ is the expectation over X an Y.

Equation (2) leads to the following empirical expression [33]:

$$
HSIC(\mathbb{P}_{xy}, \mathcal{H}, \mathcal{G}) = (m - 1)^{-1} \operatorname{tr}(\mathbf{K}_X \mathbf{H} \mathbf{K}_Y \mathbf{H})
\tag{3}
$$

where m is the number of samples, $\mathbf{K}_X \in R^{m \times m}$ and $\mathbf{K}_Y \in R^{m \times m}$ with entries are $\mathbf{K}_{Xij} = k(x_i, x_j)$ and $\mathbf{K}_{Yij} = k(y_i, y_j)$, and $\mathbf{H} \in R^{m \times m}$ is the centering matrix $H = I_m - \frac{1}{m}\mathbf{1}_m\mathbf{1}_m^T$.

With an appropriate kernel choice such as the Gaussian $k(x, y) \sim \exp(-\frac{1}{2}\|x - y\|^2/\sigma^2)$ (used in all our experiments), HSIC is zero if and only if the given two random variables X, Y are independent $P_{xy} = P_x P_y$ [34]. An intuition for the HSIC approach here is provided by the fact that the series expansion of the exponential contains a weighted sum of all moments of the data, and two distributions are equal if and only if their moments are identical. Considering the expression (3), the i element of the Equation 3 is

$$
\begin{bmatrix} k_{xi,1}, \ k_{xi,2}, \ \cdots, k_{xi,n} \end{bmatrix} \cdot \begin{bmatrix} k_{y_i,1}, \ k_{y_i,2}, \ \cdots, k_{y_i,n} \end{bmatrix}
$$

where $k_{xi,j} \equiv (\mathbf{K}_X\mathbf{H})_{i,j}$ and similarly $k_{y_i,j} \equiv (\mathbf{K}_Y\mathbf{H})_{i,j}$. This inner product will be large when the relation between each point i of X and all other points of X is similar to the relation between the corresponding point i of Y and all other points of Y, summed over all i, and where similarity is measured through the kernel $k(x_i, x_j)$ that (appropriately chosen) captures all statistical moments of the data.

Unlike mutual information, HSIC does not have an interpretation in terms of information theoretic quantities (bits or nats). On the other hand, HSIC does not require density estimation and is simple and reliable to compute. Kernel distribution embedding approaches such as HSIC can also be resistant to outliers, as can be seen by considering the effect of outliers under the Gaussian kernel. The empirical estimate converges to the population HSIC value at the rate $1/\sqrt{n}$ independent of the dimensionality of the data [33], meaning that it partially circumvents the curse of dimensionality.

While in principle HSIC can discover arbitrary dependencies between variables, in practice and with finite data the choice of σ parameter in the HSIC kernel emphasizes relationships at some scales more than others. Intuitively, two data points x, y are not well distinguished when their difference is sufficiently small or large, such that they lie on the small-slope portions of the Gaussian. This is typically handled by choosing the kernel σ based on median distances among the data [35, 36], or by a parameter search (such as grid-search [37] or random-search [38]).

3 Proposed Method

In this section we introduce the proposed HSIC-trained network. Training a deep network without backpropagation using the HSIC-bottleneck objective will be termed HSIC-bottleneck training or pre-training. The output of the bottleneck-trained network contains information necessary for classification, but not in the right form. We evaluate

two approaches to produce usable classifications from the HSIC-bottleneck trained network. First, if the outputs are one-hot, they can simply be permuted to align with the training labels. In the second scheme we append a single layer and softmax output to the frozen bottleneck-trained network, and train the appended layer using SGD without backpropagation. This step is termed <u>post-training.</u>

3.1 HSIC-Bottleneck

Suppose we have a network composed of m hidden layers $T_i(\cdot) : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, resulting in hidden representations[1] $\mathbf{Z}_i \in \mathbb{R}^{m \times d_i}$, where $i \in \{1, ..., L\}$, and m denotes batch size. Implementing the Information Bottleneck principle, we replace the original Mutual Information terms with HSIC as the learning objective:

$$\mathbf{Z}_i^* = \arg\min_{\mathbf{Z}_i} \ \text{HSIC}(\mathbf{Z}_i, \mathbf{X}) - \lambda \, \text{HSIC}(\mathbf{Z}_i, \mathbf{Y}), \tag{4}$$

where $\mathbf{X} \in \mathbb{R}^{m \times d_x}$ is the input, $\mathbf{Y} \in \mathbb{R}^{m \times d_y}$ is the label, where d_x and d_y are the dimensionalities of the input and output variable respectively. Since we concentrate on classification in our experiments, d_y is the number of classes. However, our approach can be generalized to other tasks. The λ is the Lagrange multiplier expressing the balance of IB objectives. Following (3), the HSIC of each term is:

$$\text{HSIC}(\mathbf{Z}_i, \mathbf{X}) = (m-1)^{-1} \, \text{tr}(\mathbf{K}_{Z_i} \mathbf{H} \mathbf{K}_X \mathbf{H})$$
$$\text{HSIC}(\mathbf{Z}_i, \mathbf{Y}) = (m-1)^{-1} \, \text{tr}(\mathbf{K}_{Z_i} \mathbf{H} \mathbf{K}_Y \mathbf{H}) \tag{5}$$

The formulation (4), (5) suggests that the optimal hidden representation $\mathbf{Z}_i$ finds a balance between independence from unnecessary details of the input and dependence with the output. Ideally, the information needed to predict the label is retained when (4) converges, while unnecessary information that would permit overfitting is removed.

3.2 HSIC-trained network

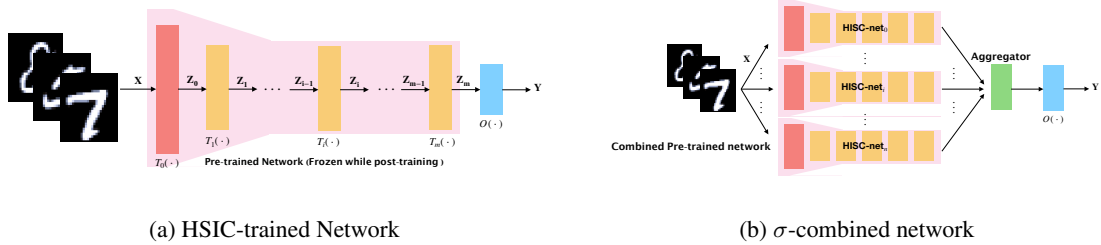(a) HSIC-trained Network   (b) σ-combined network

Figure 1: The HSIC-trained network (Figure 1a) is a standard feedforward network trained using the HSIC IB objective, resulting in hidden representations at the last layer that can be trained rapidly. Figure 1b) shows the σ-combined network, where each branch of the network HSIC-net$_j$ is trained with a specific σ. Thus, each hidden representation from the HSIC-trained network may contain different information obtained by optimizing the HSIC bottleneck objective at a particular scale. The Aggregator sums the hidden representations to form an output representation.

Our HSIC-trained network is a feedforward network (see Figure 1a,

$$L_{\text{HSIC}}(\mathbf{Z}, \mathbf{X}, \mathbf{Y}) = \sum_i^L \text{HSIC}(\mathbf{Z_i}, \mathbf{X}) - \lambda \text{HSIC}(\mathbf{Z_i}, \mathbf{Y}), \tag{6}$$

where $\mathbf{Z} = \{\mathbf{Z}_i\}$, $i \in \{0, ..., L\}$ and L is the number of hidden layers. The HSIC-trained network can be understood as resulting in the optimized encoding $P_{\theta_T}(Z_i|X)$ and the decoding $P_{\theta_T}(Y|Z_i)$ where $i \in \{0, ..., L\}$ that provides the hidden representation that contains the information required for performing the task at hand.

As shown in Section 4.1, HSIC-bottleneck training tends to produce one-hot outputs in the experiments we tried. This inspired us to use the HSIC-bottleneck objective to directly solve the classification problem. This is done by setting the dimensionality of the last layer Z_m to match the number of classes (e.g., $d_m = d_y$). Since the activated entry is typically permuted with respect to the labels (e.g., images of the digit zero activate the third output layer entry), we simply pick the output with the highest activation across the inputs of a particular class as the correct output for that class.

---

[1]This can be extended to the activations produced from convolutional layers, as each activation is flattened and stacked in array Z_i.

3.3 Post-trained network

For our experiments using a standard supervised feed-forward network for classification, we simply append a single output layer O, taking the optimized hidden representation $P_{\theta_T}(Z_L|X)$ as its input. The new layer is trained using SGD, however backpropagation is not used in any experiment other than as a baseline for comparison.

3.4 σ-network

In principle HSIC is a powerful measure of statistical independence, but the results do depend somewhat on the chosen σ parameter. To cope with this, we combine HSIC-trained networks with different sigma, and use a single aggregator to assemble all the hidden representations in single layer. The aim of the σ-combined network is that it can provide all information obtained characterized at different scales σ to the post training.

The σ-combined network architecture is illustrated in Figure 1b. Thus the total objective of the combined network is:

$$L_{\text{Comb}}(\mathbf{Z}, \mathbf{X}, \mathbf{Y}) = \sum_{j}^{n} \sum_{i}^{L} L_{\text{HSIC}_{\sigma_j}}(\mathbf{Z}_{ji}, \mathbf{X}, \mathbf{Y}), \tag{7}$$

where j and i refer to the i-th hidden representation of j-th HSIC-trained network. The performance of the σ-network is presented in 4.3.

3.5 Computation

One of the main benefits of HSIC-bottleneck is computationally efficiency. As comparison, note that the gradient of cost function with respect to the weight W at layer l is $\frac{\delta C}{\delta W_l} = \sigma^l a^{l-1^T} = W_{l+1}^T \sigma^{l+1} \odot f'(z^l) \cdot a^{l-1^T}$, where a^l is the activation $a^l = f(z^l) = f(W^l a^{l-1} + b^l)$ at layer l. Thus, the computational complexity for calculating the gradient $\frac{\delta C}{\delta W_l}$ is $O(D^3)$, where we assume that each layer has D neurons; thus, the complexity is $O(LD^3)$ for the entire network. The HSIC objective of (3) has a computational complexity of $O(M^2)$ for M data samples [33]. Therefore, the HSIC-bottleneck Equation (4) is $O(M^2)$ for a particular layer, and $O(LM^2)$ for the entire network.

Further, note that the computation of the HSIC bottleneck cost function (6) sums over the layers of the network. While the layers can be optimized sequentially, we can also perform the task in parallel, with each layer potentially computed on a separate processor.

4 Experiments

In this section, we reported several experiments that explored and validated the HSIC-trained network concept. First, to motivate our work, we plot the HSIC-bottleneck values and activation distributions of a simple model during training. Second, we provide a comparison of the performance of our approach versus standard backpropagation on a feedforward network with the same number of parameters. Next, we discuss several experiments on HSIC-trained networks with different capacities and consider the effect of the HSIC sigma hyperparameter. We briefly discuss the application of this approach to other network architectures such as ResNet.

For the experiments, we used standard feedforward networks on the MNIST/Fashion MNIST/CIFAR10 dataset for the HSIC-bottleneck analysis solving with classic classification problem (Section 4.1), HSIC sufficient statistics (Section 4.2), network capacity (Section 4.3) experiments, and tested on the architecture ResNet (Section 4.4). All experiments including standard backpropagation, pre-training, and post-training stages were using a simple SGD optimizer. The learning rates of the HSIC-trained and post-trained networks were 0.001 and 0.1 unless mentioned otherwise. The coefficient λ of the HSIC-bottleneck was empirically set to 100, which balances the compression and keeps the relevant information available to the post-training. For comparisons, we used conventional training in combination with with cross-entropy loss and backpropagation weight updates.

4.1 Pure HSIC-bottleneck training of deep networks

Figure 2 illustrates the efficacy of the HSIC-bottleneck training objective in the context of a deep neural network. Monitoring the HSIC between hidden activations and the input and output of a simple network trained using backprop shows that $\text{HSIC}(Y, Z)$ rapidly increased during early training as representations are formed, while $\text{HSIC}(X, Z)$ rapidly drops. The dependency $\text{HSIC}(Y, Z)$ varies with network depth (Figure (2e)) and depends on the choice of activation (Figure (2b)). Furthermore, it clearly parallels the increase in training accuracy (Figures (2c), (2f)).
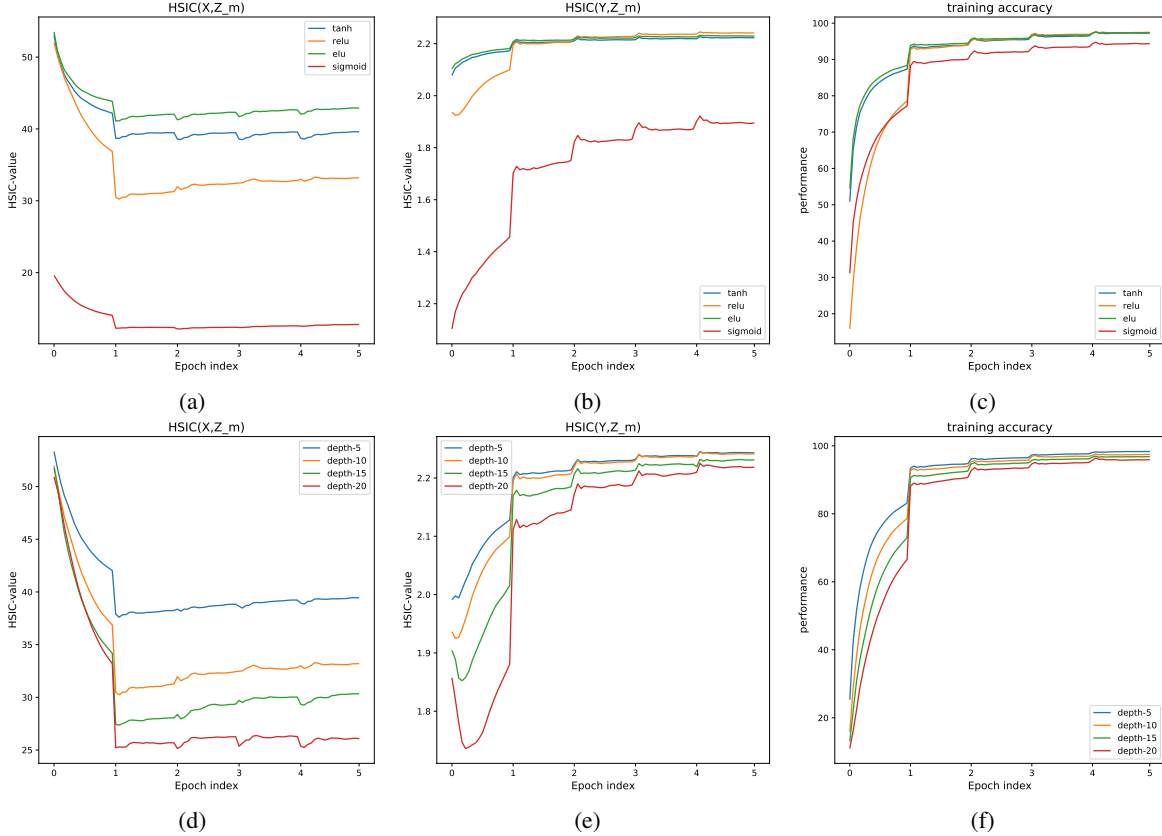
(a)   (b)   (c)

(d)   (e)   (f)

Figure 2: These experiments show the evolution of the HSIC-bottleneck quantities $\mathrm{HSIC}(\mathbf{X}, \mathbf{Z}_L)$ and $\mathrm{HSIC}(\mathbf{Y}, \mathbf{Z}_L)$ as well as training accuracy monitored during conventional backpropagation training as the network activation function (2a)-(2c) and depth (2d)-(2f) are varied. Figures (2d)-(2f) use the same non-linear activation function (ReLU) with different network depths. Conversely Figures (2a)-(2c) vary only the activation type. The figures show that across a range of different networks the training naturally increases the mutual dependency $\mathrm{HSIC}(\mathbf{Y}, \mathbf{Z}_L)$ of the last layer's representations and the label, while the dependency with the input $\mathrm{HSIC}(\mathbf{X}, \mathbf{Z}_L)$ rapidly drops early in training.

Next, Figures 3a and 3b visualize the per-class activations of the last hidden layer of the simple feedforward network. In this experiment we were interested in whether the proposed HSIC-trained network can form distinct hidden representations across classes. To explore this, we trained a 784-64-32-16-8-4-2-1-10 architecture with HSIC-bottleneck objective training (Figure 3a) and conventional training Figure (3b) on the MNIST dataset using a tanh activation function. It is interesting to see that the HSIC bottleneck better separates the hidden signals in the single neuron's representation compared to backpropagation. This suggests the HSIC-bottleneck objective can help to make the activation distribution more independent and easier to associate with its label.

In Figure 3 we found the HSIC-bottleneck objective had the effect of separating signals according to the maximum activation value. Inspired by this finding, we trained a network of size 128-128-128-128-10 using the HSIC-bottleneck. Remarkably, in experiments on CIFAR10/FashionMNIST/MNIST the network output has non-overlapping one-hot activations, as seen in Figure 4. This allows classification to be performed by simply using the highest activation value to select the class. For example in Figure 4, the activity of images of the digit 'zero' has highest density at entry seven.

Next, we showed that our approach can produce results competitive with standard training. Figure 5 illustrates the result of backpropagation and proposed HSIC-bottleneck training. At the first epoch our approach outperforms the standard training, reaching 43%, 85%, and 95% on Cifar10, FashionMNIST, and MNIST respectively. At the end of training the accuracies are generally similar to those of standard backpropagation training of the same architectures.

In standard backpropagation, the computational complexity for training is high. All layers are computed sequentially based on the preceding signals. In contrast, the proposed HSIC-bottleneck can train each layer individually, allowing each processor to optimize a layer of the network. As noted earlier in section 3.1 this facilitates parallel implementation.
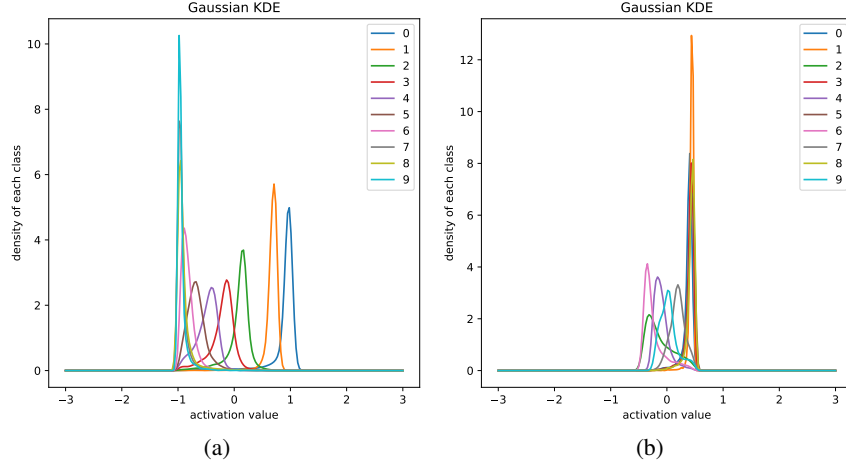
Figure 3: The tanh activation distribution of the last hidden layer (consisting of a single neuron) of a network sized 784-64-32-16-8-4-2-**1**, trained with the HSIC-bottleneck objective (3a) and backpropagation (3b). The class-conditional activations in (a) have less overlap than those in (b). The curves are calculated with Gaussian kernel density estimation. It's worth to note that due to the small gradient at two sides of tanh, the activation in backprop (3b) is hard to reach the extreme values. Contrary to backprop, the proposed HSIC-bottleneck (3a) can separate out classed signals better.

We believed these experiments support the idea that training encodes the input variables in a form from which the desired output can be easily discovered. With these motivations, we used the HSIC-bottleneck as an training objective to train deep representations without backpropagation. The resulting HSIC-trained network then provides good representations for the output layer during post-training.

4.2 Using the HSIC-bottleneck as sufficient statistics for post-training

An interesting question regarding deep neural networks is how effectively these stacked layers learn the information from the input and the label. To explore this, we fixed all the hyper-parameters of HSIC-trained network except the training time (number of epochs). We expect that training the HSIC-trained network for more epochs will result in a hidden representation that better represents the information needed to predict the label, ending up with higher accuracy in post-training. Figure 6 shows the result of this experiment. The figure shows the accuracy and loss of a single epoch of post-training on a 15-layer HSIC-trained network trained for 5, 25, and 100 epochs. From Figure 6 it is evident that the HSIC-trained network can boost accuracy even at the very beginning of SGD post-training. Additionally, as the HSIC-trained network trains longer, the post-training yields higher accuracy.

4.3 The effect of network capacity

The five experiments in Figure 7 show the effect of different widths in the HSIC-trained network, followed by a standard post-training step. Since the HSIC-trained networks have different width, the network has converged with respect to the objective (6).

Examining Figure 7a and Figure 7b indicates that larger networks (say width-128 compared to width-8) lead to faster post-training. This suggests the HSIC bottleneck objective (6) works effectively on large networks that provide more relevant information to the post-training, whereas small pre-networks may contain less information to train with.

As mentioned in Section 3.2 the HSIC results do depend on the chosen σ. In our experiments, making σ a learnable parameter did not produce significant performance improvement. Consequently, in this experiment we aggregate several HSIC-trained networks with different σ together in parallel to better capture dependencies at multiple scales. Our experiment setup trains three parallel HSIC networks having the same five-layer configuration but with different kernel widths $\sigma = 5$, $\sigma = 10$, and $\sigma = 15$ as shown in Figure 7c. The results show post-training on the σ-combined network outperforms other experiments, suggesting that it is providing additional information relating to the corresponding scale to the post-training. It also indicates that a single σ is not sufficient to capture all dependencies in these networks, as anticipated.
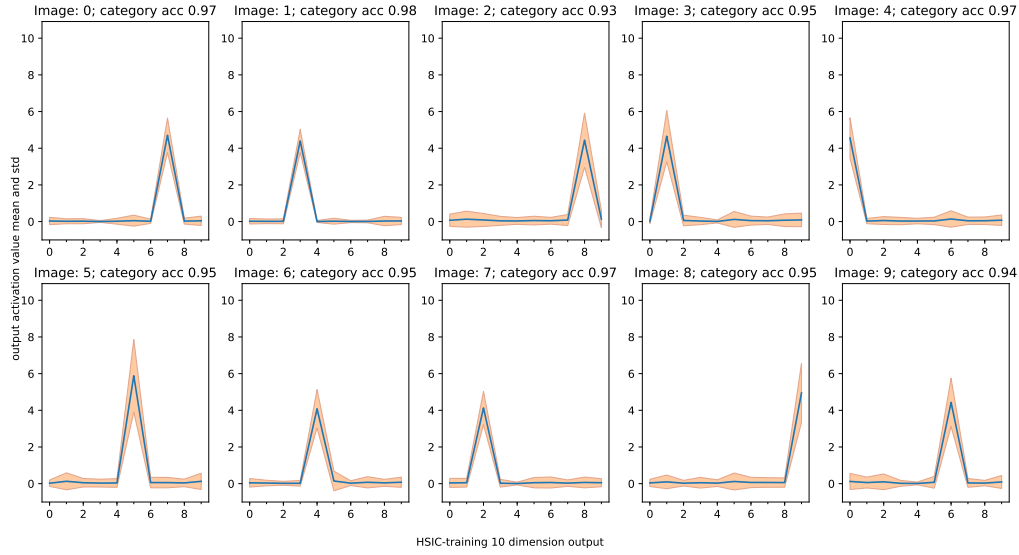
Figure 4: The MNIST output category distribution from a shallow dense network. Each of the sub-figure is the output from specific category as labeled in title of figure. The one-hot effect from the HSIC-bottleneck training inspired us to make permutation on the label based on ordering the entries by activation. The activation value indicates how much the image class activates the output entry (e.g., most of the 'zero' input digits activate the 7th output, as seen in the top-left plot). The category accuracy labelled at each sub-figure title gives the proportion of class data belongs to this entry.

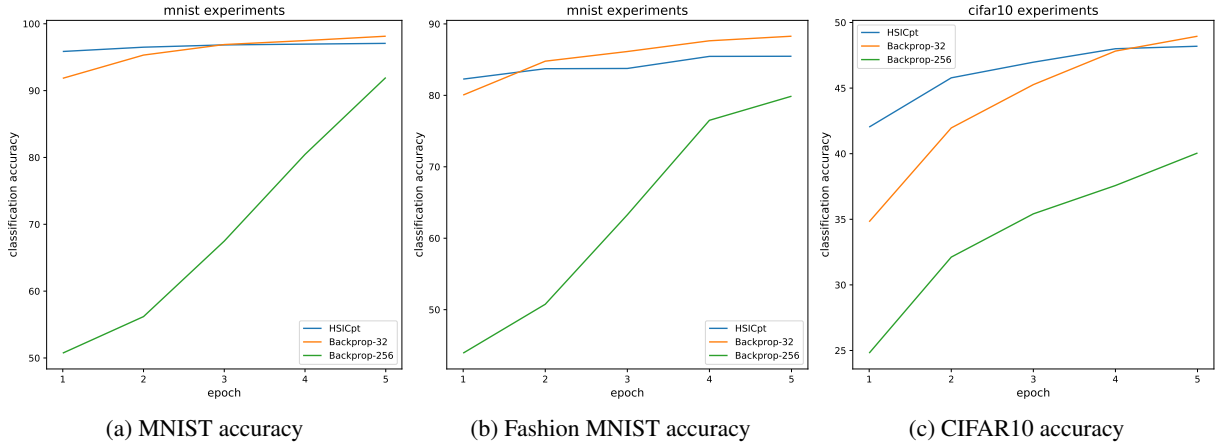| (a) MNIST accuracy | (b) Fashion MNIST accuracy | (c) CIFAR10 accuracy |

Figure 5: The accuracy of proposed pure HSIC-bottleneck training on standard classification problems. The experiments use the same training configuration except for batch size (32 for BackProp-32 and 256 for BackProp-256 and HSICpt). Note that the HSICpt is using Equation (4) only, without the crossentropy objective and backpropagation, and it is outperforming the same configuration of BackProp-256 dramatically.

4.4 Experiments on ResNet

Most of our results aimed at demonstrating the training efficacy of the new paradigm are based on basic feedforward fully connected networks. To show that the paradigm is also effective for other architectures, we also train a ResNet on MNIST/CIFAR10 dataset. For this special case of ResNet, we added the loss (4) to the output of each residual block.

In Figure 8, we show the results for a HSIC-trained network with five convolutional residual blocks on several datasets. Each experiment includes 50 HSIC-trained epochs followed by post-training with a one-layer classifier network, and compared with its standard backpropagation-trained counterpart. Our results show that post-training can rapidly converge to high accuracy performance by making use of the distinct representations from the HSIC-trained network.
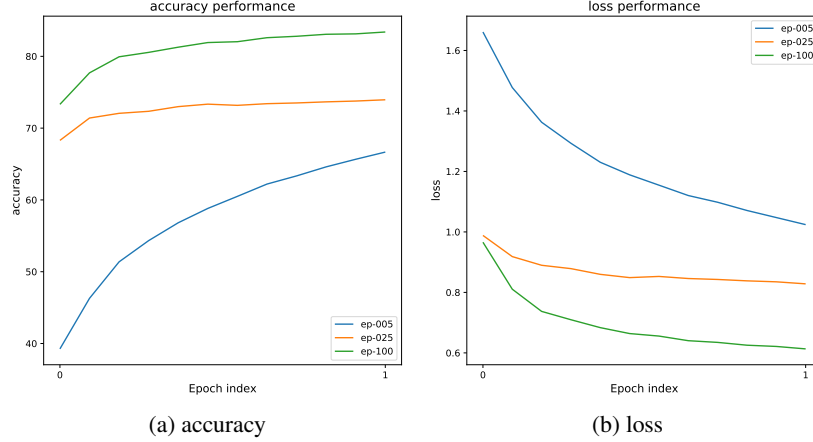
(a) accuracy

(b) loss

Figure 6: The first-epoch post-training performance using a pre-trained network trained for 5, 25, and 100 epochs (labeled as ep-005, ep-025, and ep-100 respectively). Pre-training to convergence provides better post-training performance in this experiment.
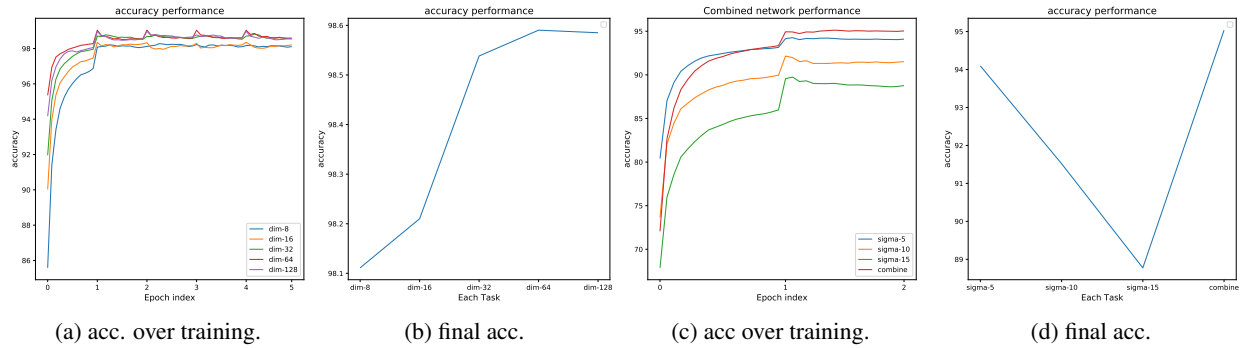

(a) acc. over training.

(b) final acc.

(c) acc over training.

(d) final acc.

Figure 7: Comparison of post-training with different sized HSIC-trained networks (Figures 7a, 7b) and a σ-combined network (Figures 7c, 7d). The network capacity in post-training experiments is labeled in Figure 7a. This figure suggests that larger capacity provides more relevant information to the post-training. The σ-combined network experiment demonstrates that using several σ improves performance.

The final accuracy at the end of the first epoch is as follows: Figure 8a (MNIST), Figure 8b (Fashion MNIST), Figure 8c (CIFAR10) show the final accuracy of the HSIC-trained (95.9%, 87.8%, 47.4%) and backpropagation-trained networks (92.9%, 80.6%, 38.6%), respectively.

5   Conclusion

We presented a new approach to training deep neural networks without the use of backpropagation. The method is inspired by the information bottleneck and can be seen as an approximation there-of. HSIC-bottleneck training of several standard classification problems results in one-hot outputs that can be directly permuted to perform classification, with accuracy approaching that of standard backpropagation training of the same architectures. Performance is further improved by using the outputs as representations for a second post-hoc training stage, in which a single layer (and softmax) is appended and trained with conventional SGD, but without backpropagation.

The HSIC bottleneck trained network provides good hidden representations by removing irrelevant information and retaining information that is important for the task at hand. As a result, the performance converges significantly faster than the standard backpropagation method, both in training time and an in number of epochs required. It is likely that scheduling of the HSIC-bottleneck coefficient can lead to further improvements in convergence rate.

HSIC bottleneck training has several benefits over backpropagation:

- It has significantly faster convergence rates;

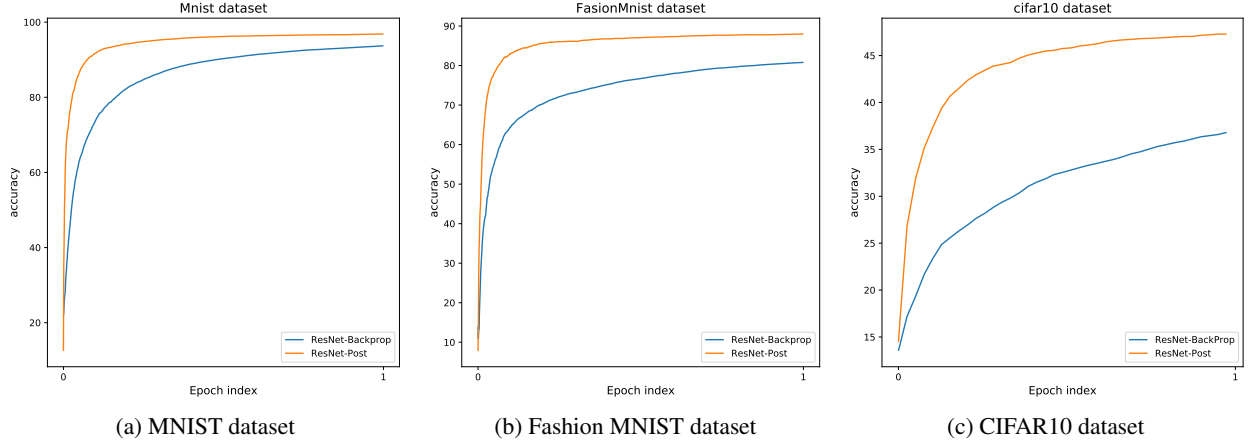(a) MNIST dataset   (b) Fashion MNIST dataset   (c) CIFAR10 dataset

Figure 8: ResNet post-training performance on several datasets. In this experiment, we evaluate our work versus conventional backpropagation training on a small ResNet with five residual blocks. Our results (labeled as ResNet-Post) outperforms standard backpropagation training (labeled ResNet-Backprop) across the initial batch updates through the full first epoch. The CIFAR10 result is well below state-of-the-art performance, because we are not using a state-of-the-art architecture. Nevertheless, the HSIC-bottleneck network provides a significant boost in performance.

- it removes the vanishing and exploding gradient issues found in backpropagation, since it solves the problem layer-by-layer without the use of the chain rule;
- it removes the need for backward sweeps;
- it potentially allows layers to be trained in parallel, using layerwise block coordinate descent.

Our work is an initial exploration of backpropagation-free learning using the HSIC bottleneck, and, in common with other explorations of new training methods for deep learning, e.g., [14]) does not attempt to achieve state-of-the-art performance. Future work might consider careful tuning of the HSIC σ to improve performance, and evaluate the HSIC-bottleneck approach on a broader range of architectures.

Acknowledgments

References

[1] Y. Bengio, D. Lee, J. Bornschein, and Z. Lin, "Towards biologically plausible deep learning," CoRR, vol. abs/1502.04156, 2015.

[2] I. Pozzi, S. Boht'e, and P. R. Roelfsema, "A biologically plausible learning rule for deep learning in the brain," CoRR, vol. abs/1811.01768, 2018.

[3] S. Bartunov, A. Santoro, B. A. Richards, G. E. Hinton, and T. P. Lillicrap, "Assessing the scalability of biologically-motivated deep learning algorithms and architectures," CoRR, vol. abs/1807.04587, 2018.

[4] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[5] R. Shwartz-Ziv and N. Tishby, "Opening the black box of deep neural networks via information," CoRR, vol. abs/1703.00810, 2017.

[6] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," in International Conference on Learning Representations, 2018.

[7] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in CoRR, 1999.

[8] I. Belghazi, S. Rajeswar, A. Baratin, R. D. Hjelm, and A. C. Courville, "MINE: mutual information neural estimation," CoRR, vol. abs/1801.04062, 2018.

[9] P. J. Werbos, "Backpropagation through time: what it does and how to do it," Proceedings of the IEEE, vol. 78, pp. 1550–1560, Oct 1990.

[10] D. Balduzzi, H. Vanchinathan, and J. Buhmann, "Kickback cuts backprop's red-tape: Biologically plausible credit assignment in neural networks," in Proc. Conference on Artificial Intelligence (AAAI), 2015.

[11] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random feedback weights support learning in deep neural networks," arXiv preprint arXiv:1411.0247, 2014.

[12] T. H. Moskovitz, A. Litwin-Kumar, and L. F. Abbott, "Feedback alignment in deep convolutional networks," CoRR, vol. abs/1812.06488, 2018.

[13] A. A. Kohan, E. A. Rietman, and H. T. Siegelmann, "Error forward-propagation: Reusing feedforward connections to propagate errors in deep learning," CoRR, vol. abs/1808.03357, 2018.

[14] A. Choromanska, B. Cowen, S. Kumaravel, R. Luss, M. Rigotti, I. Rish, P. Diachille, V. Gurev, B. Kingsbury, R. Tejwani, and D. Bouneffouf, "Beyond backprop: Online alternating minimization with auxiliary variables," in Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA, pp. 1193–1202, 2019.

[15] E. Cambria, G.-B. Huang, L. Kasun, H. Zhou, C.-M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li, V. Leung, L. Feng, Y. Ong, M.-H. Lim, A. Akusok, A. Lendasse, F. Corona, R. Nian, Y. Miche, and J. Liu, "Extreme learning machines," IEEE Intelligent Systems, vol. 28, pp. 30–59, 11 2013.

[16] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 42, pp. 513–529, April 2012.

[17] B. Widrow, A. Greenblatt, Y. Kim, and D. Park, "The no-prop algorithm: A new learning algorithm for multilayer neural networks," Neural Networks, vol. 37, pp. 182–188, 2013.

[18] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," CoRR, vol. abs/1411.1792, 2014.

[19] J. M. Kleinberg, "Two algorithms for nearest-neighbor search in high dimensions," in Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing, STOC '97, (New York, NY, USA), pp. 599–608, ACM, 1997.

[20] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning," in Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11, (USA), pp. 1089–1096, Omnipress, 2011.

[21] A. Rosenfeld and J. K. Tsotsos, "Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing," CoRR, vol. abs/1802.00844, 2018.

[22] K. He, Y. Wang, and J. E. Hopcroft, "A powerful generative model using random weights for the deep image representation," in NIPS, pp. 631–639, 2016.

[23] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," CoRR, vol. abs/1310.1531, 2013.

[24] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," CoRR, vol. abs/1403.6382, 2014.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Commun. ACM, vol. 60, pp. 84–90, may 2017.

[26] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," Nature Communications, vol. 7, 2016.

[27] W. Xiao, H. Chen, Q. Liao, and T. A. Poggio, "Biologically-plausible learning algorithms can scale to large datasets," CoRR, vol. abs/1811.03567, 2018.

[28] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," Cognitive Science, vol. 11, no. 1, pp. 23–63, 1987.

[29] T. M. Cover and J. A. Thomas, Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). New York, NY, USA: Wiley-Interscience, 2006.

[30] N. K. and, "Input feature selection by mutual information based on Parzen window," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, pp. 1667–1671, Dec 2002.

[31] P. Brakel and Y. Bengio, "Learning independent features with adversarial nets for non-linear ICA," 2018.

[32] Z. Goldfeld, E. van den Berg, K. H. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating information flow in neural networks," CoRR, vol. abs/1810.05728, 2018.

[33] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in Proceedings of the 16th International Conference on Algorithmic Learning Theory, ALT'05, (Berlin, Heidelberg), pp. 63–77, Springer-Verlag, 2005.

[34] B. K. Sriperumbudur, K. Fukumizu, and G. Lanckriet, "On the relation between universality, characteristic kernels and RKHS embedding of measures," in International Conference on Artificial Intelligence and Statistics, pp. 773–780, 2010.

[35] D. Sejdinovic, A. Gretton, B. K. Sriperumbudur, and K. Fukumizu, "Hypothesis testing using pairwise distances and associated kernels (with appendix)," CoRR, vol. abs/1205.0411, 2012.

[36] M. Sugiyama and M. Yamada, "On kernel parameter selection in hilbert-schmidt independence criterion," IEICE Transactions on Information and Systems, vol. E95D, 10 2012.

[37] G. E. Hinton, A Practical Guide to Training Restricted Boltzmann Machines, pp. 599–619. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[38] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," J. Mach. Learn. Res., vol. 13, pp. 281–305, Feb. 2012.