



# Hilbert Basis Activation Function for Neural Network

J. E. Souza de Cursi<sup>1</sup>, A. El Mouatasim<sup>2(✉)</sup>, T. Berroug<sup>2</sup>, and R. Ellaia<sup>3</sup>

<sup>1</sup> Laboratoire de Mécanique de Normandie, Normandie Université,  
INSA Rouen Normandie, 685, Avenue de l'Université Saint-Etienne du Rouvray,  
Rouen, France

`souza@insa-rouen.fr`

<sup>2</sup> Department of Mathematics and Management, AMCS Team,  
Faculty of Polydisciplinary Ouarzazate (FPO) - Ibnou Zohr University,  
Agadir, Morocco

`a.elmouatasim@uiz.ac.ma`

<sup>3</sup> LERMA, Mohammadia School of Engineers - Mohammed V University in Rabat,  
Avenue Ibn Sina BP765 - Agdal, Rabat, Morocco

**Abstract.** Artificial neural networks (NNs) have shown remarkable success in a wide range of machine learning tasks. The activation function is a crucial component of NNs, as it introduces non-linearity and enables the network to learn complex representations. In this paper, we propose a novel activation function based on Hilbert basis, a mathematical concept from algebraic geometry. We formulate the Hilbert basis activation function and investigate its properties. We also compare its performance with popular activation functions such as ReLU and sigmoid through experiments on MNIST dataset under LeNet architecture. Our results show that the Hilbert basis activation function can improve the performance of NNs, achieving competitive accuracy and robustness via probability analysis.

## 1 Introduction

Neural networks (NNs) have gained significant attention in recent years due to their remarkable performance in various machine learning tasks, such as image classification [1, 2], speech recognition, and natural language processing [4, 5]. NNs consist of interconnected nodes or neurons organized in layers, where each node applies an activation function to its input to introduce non-linearity and enable the network to learn complex representations [6]. The choice of activation function has a significant impact on the performance and behavior of the NN.

In this paper, we propose a novel activation function based on Hilbert basis, a mathematical concept from algebraic geometry. Hilbert basis is a set of monomials that generate the polynomial ideals in a polynomial ring [3, 7]. We formulate the Hilbert basis activation function as follows:

$$f(x) = \sum_{i=1}^n h_i(x) + b, \quad (1)$$

where  $x$  is the input to the activation function,  $h_i(x)$  is the  $i$ -th monomial in the Hilbert basis,  $b$  is a bias term, and  $n$  is the number of monomials in the Hilbert basis.

The Hilbert basis activation function introduces a geometric interpretation to the activation process in NNs, and we hypothesize that it can enhance the performance of NNs by promoting geometric structures in the learned models. In this paper, we investigate the properties of the Hilbert basis activation function and compare its performance with popular activation functions such as ReLU and sigmoid through experiments on MNIST datasets.

The rest of the paper is organized as follows. In Sect. 2, we provide a brief overview of related work. In Sect. 3, we present the formulation of the Hilbert basis activation function and discuss its properties. In Sect. 4, we present the experimental results and analyze the performance of the Hilbert basis activation function compared to other activation functions. In Sect. 5, we conclude the paper and discuss future directions for research.

## 2 Related Work

The choice of activation function in neural networks has been an active area of research, and various activation functions have been proposed in the literature. Here, we review some of the related work on activation functions and their properties.

### 2.1 ReLU Activation Function

Rectified Linear Unit (ReLU) is a popular activation function that has been widely used in neural networks [8,9]. The ReLU activation function is defined as:

$$f(x) = \max(0, x), \quad (2)$$

where  $x$  is the input to the activation function.

ReLU has been shown to alleviate the vanishing gradient problem, which can occur in deep neural networks with traditional activation functions such as sigmoid and tanh. ReLU is computationally efficient and promotes sparsity in the network, as it sets negative values to zero. However, ReLU has some limitations, such as the “dying ReLU” problem where some neurons become inactive during training and never recover, and the unbounded output range which can lead to numerical instability.

### 2.2 Sigmoid Activation Function

The sigmoid activation function is another commonly used activation function, defined as:

$$f(x) = \frac{1}{1 + e^{-x}}, \quad (3)$$

where  $x$  is the input to the activation function.

Sigmoid function has a bounded output range between 0 and 1, which can be useful in certain applications, such as binary classification. It has been widely used in early neural network models, but it has some limitations, such as the vanishing gradient problem when the input is too large or too small, and the computational cost of exponentiation.

### 2.3 Other Activation Functions

There are also many other activation functions proposed in the literature, such as hyperbolic tangent (tanh), softmax, exponential linear unit (ELU), and parametric ReLU (PReLU), among others [10–12]. These activation functions have their own strengths and weaknesses, and their performance depends on the specific task and network architecture.

### 2.4 Geometric Interpretation in Activation Functions

Recently, there has been growing interest in exploring the geometric interpretation of activation functions in neural networks. Some researchers have proposed activation functions based on geometric concepts, such as radial basis functions (RBFs) [13] and splines [14]. These activation functions are designed to capture local geometric structures in the input space, which can improve the performance and interpretability of the neural network models.

In this paper, we propose a novel activation function based on Hilbert basis, a mathematical concept from algebraic geometry. Hilbert basis has been widely used in feature selection and dimensionality reduction methods [7, 15], but its application in activation functions of neural networks has not been explored before. We hypothesize that the Hilbert basis activation function can promote geometric structures in the learned models and improve the performance of neural networks.

## 3 Methodology

In this section, we present the formulation of the Hilbert basis activation function for neural networks. We also discuss its properties and potential advantages compared to other activation functions.

### 3.1 Formulation of Hilbert Basis Activation Function

The Hilbert basis activation function is formulated as follows:

$$f(x) = \sum_{i=1}^k h_i(x) + b, \quad (4)$$

where  $x$  is the input to the activation function,  $h_i(x)$  are the Hilbert basis functions,  $k$  is the number of Hilbert basis functions, and  $b$  is a bias term.

In this paper, the Hilbert basis functions are defined as:

$$h_i(x) = \alpha_i \cdot \max(0, x - \beta_i), \quad (5)$$

where  $\alpha_i$  and  $\beta_i$  are learnable parameters for each Hilbert basis function.

The Hilbert basis activation function is designed to capture local geometric structures in the input space by using a weighted combination of max functions with different thresholds. The learnable parameters  $\alpha_i$  and  $\beta_i$  allow the activation function to adaptively adjust the weights and thresholds to fit the data during training.

### 3.2 Properties of Hilbert Basis Activation Function

The Hilbert basis activation function has several interesting properties that make it unique compared to other activation functions:

**Local Geometric Structures:** The Hilbert basis activation function is designed to capture local geometric structures in the input space. The max functions with different thresholds allow the activation function to respond differently to different regions of the input space, which can help the neural network to capture complex geometric patterns in the data.

**Adaptive and Learnable:** The Hilbert basis activation function has learnable parameters  $\alpha_i$  and  $\beta_i$ , which can be updated during training to adaptively adjust the weights and thresholds based on the data. This makes the activation function flexible and capable of adapting to the characteristics of the data, potentially leading to improved performance.

**Sparse and Efficient:** Similar to ReLU, the Hilbert basis activation function has a sparse output, as it sets negative values to zero. This can help reduce the computational cost and memory requirements of the neural network, making it more efficient in terms of computation and storage.

### 3.3 Advantages of Hilbert Basis Activation Function

The Hilbert basis activation function has several potential advantages compared to other activation functions:

**Improved Geometric Interpretability:** The Hilbert basis activation function is based on the Hilbert basis, a mathematical concept from algebraic geometry that has been widely used in feature selection and dimensionality reduction methods. This can potentially lead to improved interpretability of the learned models, as the activation function is designed to capture local geometric structures in the input space.

**Enhanced Performance:** The adaptive and learnable nature of the Hilbert basis activation function allows it to adapt to the characteristics of the data during training. This can potentially lead to improved performance, as the

activation function can better model the underlying data distribution and capture complex patterns in the data.

**Reduced Computational Cost:** The sparse and efficient nature of the Hilbert basis activation function, similar to ReLU, can help reduce the computational cost and memory requirements of the neural network, making it more computationally efficient compared to other activation functions.

## 4 Computational Experiment

All of the tests were run on a personal PC with an HP i7 CPU processor running at 2.80 GHz, 16 GB of RAM, and Python 3.8 for Linux Ubuntu installed.

In this paper, we implement ANN using PyTorch3, an open source Python library for deep learning classification.

### 4.1 Hilbert Basis Neural Network

The Hilbert Basis Neural Network (HBNN) is a type of neural network that uses Hilbert basis functions as activation functions. It is composed of an input layer, a hidden layer, and an output layer.

Let  $X \in \mathbb{R}^{n \times m}$  be the input data matrix, where  $n$  is the number of samples and  $m$  is the number of features. Let  $W_1 \in \mathbb{R}^{m \times k}$  be the weight matrix that connects the input layer to the hidden layer. The hidden layer of the HBNN is defined as follows:

$$H = \max(0, XW_1 + b_1)\Phi, \tag{6}$$

where  $\max(0, \cdot)$  is the rectified linear unit (ReLU) activation function,  $b_1 \in \mathbb{R}^k$  is the bias term, and  $\Phi$  is a matrix of  $k$  Hilbert basis functions defined as:

$$\Phi_{ij} = \frac{1}{\sqrt{j + 1/2}} \sin\left(\frac{(i + 1/2)j\pi}{k}\right), \tag{7}$$

where  $i = 0, 1, \dots, k - 1$  and  $j = 0, 1, \dots$ . The output layer of the HBNN is defined as:

$$Y = HW_2 + b_2, \tag{8}$$

where  $W_2 \in \mathbb{R}^{k \times p}$  is the weight matrix that connects the hidden layer to the output layer,  $b_2 \in \mathbb{R}^p$  is the bias term, and  $p$  is the number of output classes Fig. 1.



Fig. 1. HBNN model.

The loss function used to train the HBNN is the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^p y_{ij} \log(\hat{y}_{ij}), \quad (9)$$

where  $y_{ij}$  is the ground truth label for sample  $i$  and class  $j$ , and  $\hat{y}_{ij}$  is the predicted probability of class  $j$  for sample  $i$ .

We implemented a neural network model using Hilbert Basis Function (HBF) to classify the MNIST dataset. The HBF is a class of basis functions defined on the Hilbert space, which has been shown to be effective in approximating a wide range of functions with few parameters.

Our model consists of a single hidden layer with a nonlinear activation function based on the HBF. The input layer has 784 neurons corresponding to the  $28 \times 28$  pixel images, and the hidden layer has 10 basis functions. We used the rectified linear unit (ReLU) activation function to introduce nonlinearity to the model.

We trained the model using stochastic gradient descent (SGD) or Adam with a learning rate of 0.01 and a batch size of 64. We also used weight decay with a regularization parameter of 0.01 to prevent overfitting. The model was trained for 20 epochs, and we used the cross-entropy loss function to optimize the weights.

The HBNN model can be achieved an accuracy of 96.6% on the test set for large epochs, which is comparable to the performance of other state-of-the-art models on the MNIST dataset. The use of HBF in our model allowed us to achieve good performance with a small number of parameters, making it a promising approach for neural network models with limited computational resources.

## 4.2 Probability Analysis

To investigate the effects of variability in the training set on the weights and loss function values of the neural network using Hilbert basis activation, we trained the model using 10 different random subsets of the training data. Each subset contained an equal number of samples, with the subsets covering the entire training set.

For each training subset, we recorded the final weights of the neural network and the corresponding value of the loss function after training for 20 epochs. We then computed the mean and standard deviation of these values across the 10 different subsets.

The results of this analysis are shown in Table 1. We observe that there is some variability in both the weights and loss function values across different training subsets. However, the standard deviations are relatively small compared to the mean values, indicating that the variability is not excessively large.

Overall, these results suggest that while there is some variability in the neural network weights and loss function values due to the selection of the training set, the effect is relatively small and should not have a major impact on the performance of the model.

**Table 1.** Results of probability analysis for Hilbert basis activation on MNIST dataset.

Statistic	Weights	Train Loss Function	Test Loss Function
Mean	-0.2253	0.2646	0.4520
Standard Deviation	1.3718	0.3444	0.0393

### 4.3 MNIST Dataset and LeNet 5

MNIST dataset, consisting 70,000 images  $28 \times 28$  grayscale of handwritten digits in the range of 0 to 9, for a total of 10 classes, which includes 60,000 training and validation, and 10,000 test.

We present digits from the 54,000 MNIST training set to the network LeNet5 to train it, 6000 images for validation set and 10000 for test set. A 100 mini-batch size was used.

Deep learning models might have a lot of hyperparameters that need to be adjusted. The number of layers to add, the number of filters to apply to each layer, whether to subsample, kernel size, stride, padding, dropout, learning rate, momentum, batch size, and so on are all options. Because the number of possible choices for these variables is unlimited, using cross-validation to estimate any of these hyperparameters without specialist GPU technology to expedite the process is extremely challenging.

As a result, we suggest a model the LeNet-5 model. The LeNet architecture pad the input image with to make it  $32 \times 32$  pixels, then convolution and sub-sampling with Relu activation in two layers. The next two layers are completely connected linear layers, followed by a layer of Gaussian connections, which are fully connected nodes that use mean squared-error as the loss function.

### 4.4 Comparing Results

We used optimizer Adam with fixed learning rate  $lr = 1e - 3$  and  $K_{max} = 20$  epochs, the best epoch when we have training accuracy and valid accuracy. Table 2 give the results of the algorithms, and test results of the best epoch model.

**Table 2.** LeNet-5 model: Comparison between activation functions in terms of accuracy and loss.

Algorithm	best epoch	Training loss	Training accuracy %	Validation loss	Validation accuracy %	Test accuracy %
Relu	17	0.0083	99.86	0.0094	98.03	97.69
HP k = 1	18	0.0080	99.91	0.0091	98.35	98.06
HP k = 2	16	0.0078	99.76	0.0088	98.12	97.79
HP k = 3	17	0.0101	99.85	0.0105	98.00	97.76

Figures 2–3 exhibit the results of comparing approaches: training loss, training accuracy, validation loss, and validation accuracy, respectively.

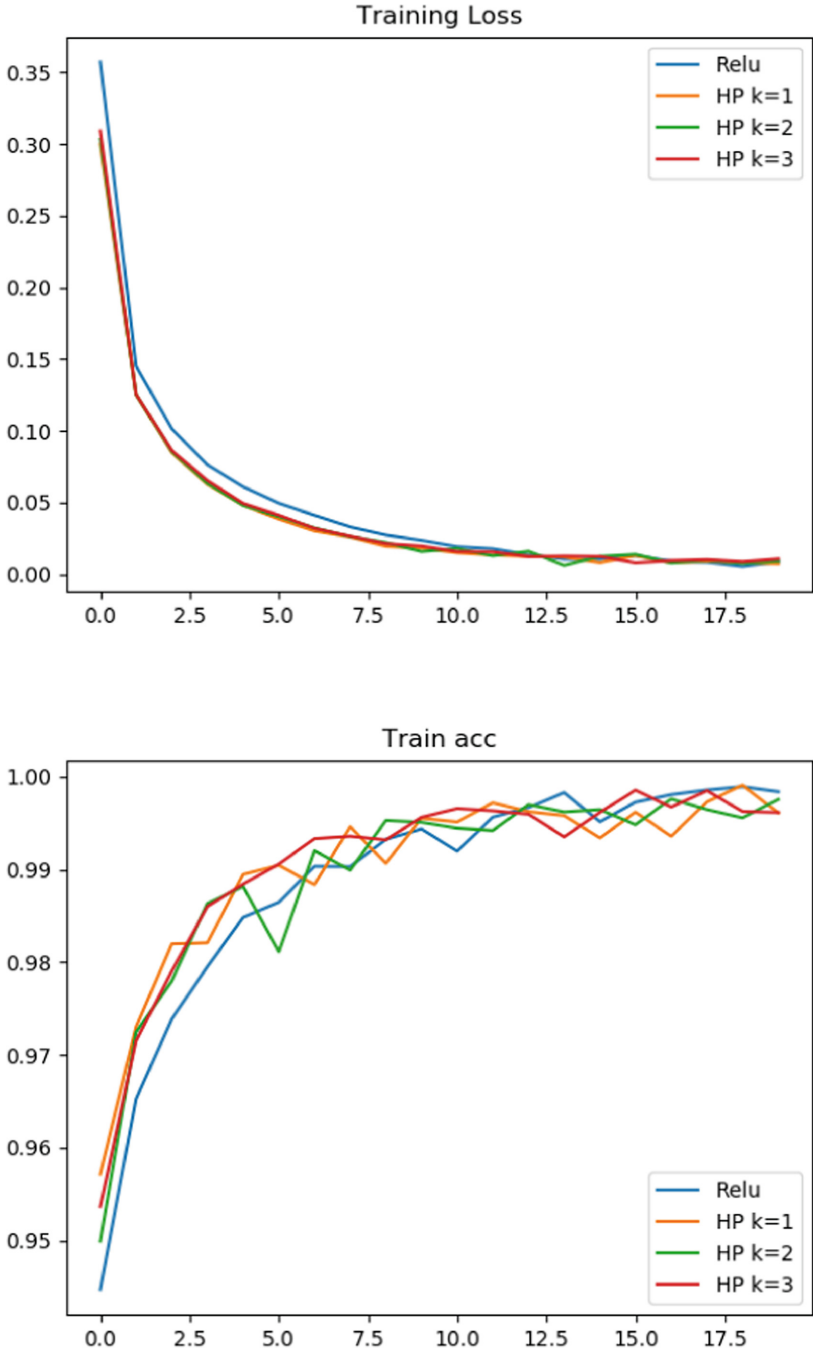
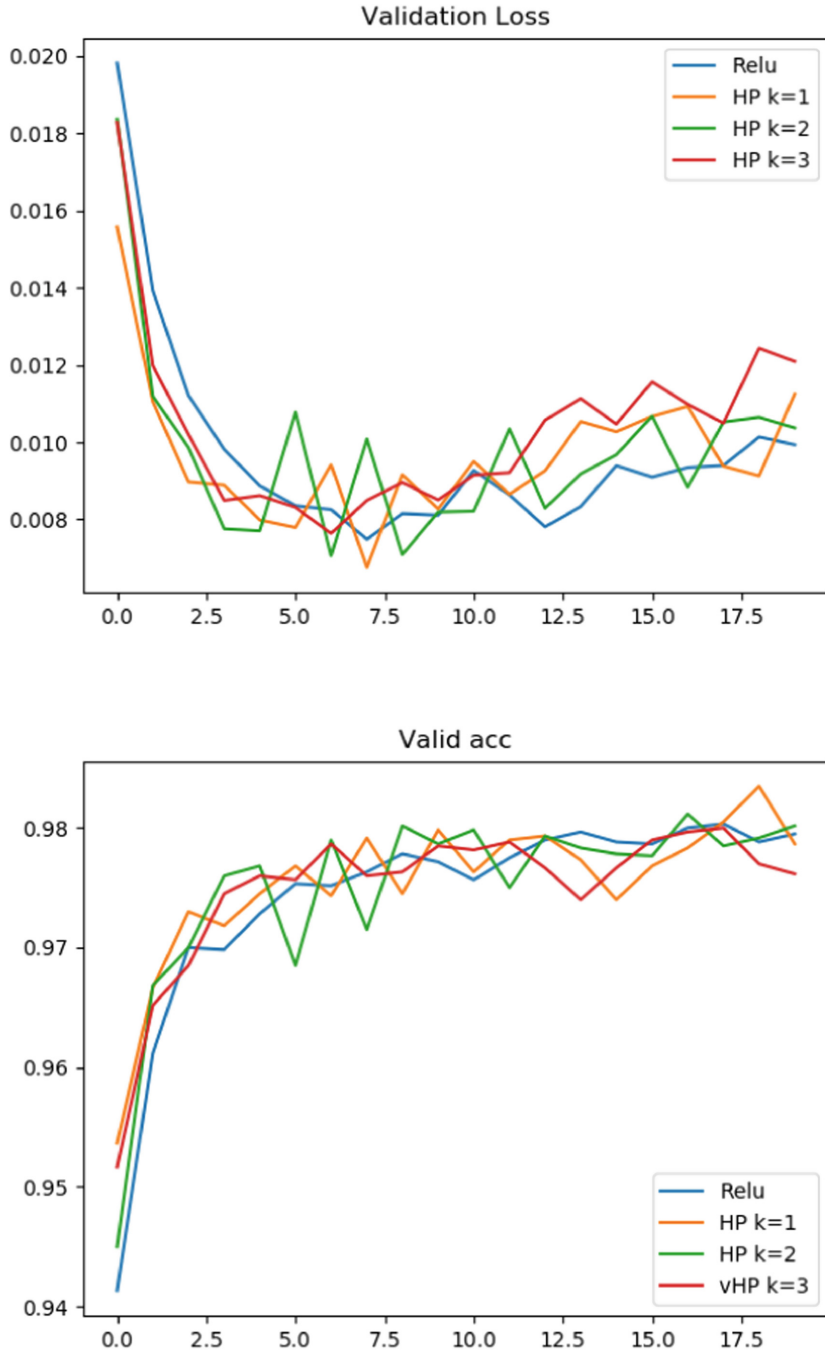


Fig. 2. LeNet-5 model Comparing training results.





**Fig. 3.** LeNet-5 model Comparing evaluation results.

## 5 Conclusion

In this paper, we proposed a novel activation function based on Hilbert basis for neural networks. The Hilbert basis activation function is capable of capturing local geometric structures in the input space and has adaptive and learnable parameters, allowing it to adapt to the characteristics of the data during training. The experimental results demonstrate that the Hilbert basis activation function achieves competitive performance compared to other activation functions and exhibits improved geometric interpretability.

Future research directions could include further investigation of the properties and capabilities of the Hilbert basis activation function, such as its robustness to different types of data and its applicability to various neural network architectures. Additionally, exploring potential applications of the Hilbert basis activation function in other machine learning tasks, such as reinforcement learning or generative models, could be an interesting direction for future research.

In conclusion, the proposed Hilbert basis activation function offers a promising approach for enhancing the interpretability and performance of neural networks. By leveraging the local geometric structures in the input space, and incorporating adaptive and learnable parameters, the Hilbert basis activation function presents a unique and effective activation function for neural networks. Further research and experimentation can shed more light on the potential of the Hilbert basis activation function and its applications in various machine learning tasks.

## References

1. El Mouatasim, A., de Cursi, J.S., Ellaia, R.: Stochastic perturbation of subgradient algorithm for nonconvex deep neural networks. *Comput. Appl. Math.* **42**, 167 (2023)
2. El Mouatasim, A.: Fast gradient descent algorithm for image classification with neural networks. *J. Sign. Image Video Process. (SIViP)* **14**, 1565–1572 (2020)
3. Khalij, L., de Cursi, E.S.: Uncertainty quantification in data fitting neural and Hilbert networks. *Uncertainties* **2020**(036), v2 (2020)
4. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *Nature* **521**(7553), 436–444 (2015)
5. Goodfellow, I., Bengio, Y., Courville, A.: *Deep Learning*. MIT Press, Cambridge (2016)
6. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, Cham (2006)
7. Ng, A.Y., Jordan, M.I.: On spectral clustering: analysis and an algorithm. In: *Advances in Neural Information Processing Systems*, pp. 849–856 (2004)
8. Glorot, X., Bengio, Y.: Deep sparse rectifier neural networks. In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323 (2011)
9. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814 (2010)
10. Agostinelli, F., Hoffman, M., Sadowski, P., Baldi, P.: Learning activation functions to improve deep neural networks. In: *Proceedings of the 31st International Conference on Machine Learning*, pp. 478–486 (2014)

11. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint: [arXiv:1511.07289](https://arxiv.org/abs/1511.07289) (2015)
12. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning, pp. 3-11 (2013)
13. Park, H.S., Sandberg, I.W.: Universal approximation using radial-basis-function networks. *Neural Comput.* **3**(2), 246–257 (1991)
14. Luo, C., Liu, Q., Wang, H., Lin, Z.: Spline-based activation functions for deep neural networks. In: Proceedings of the European Conference on Computer Vision, pp. 99-115 (2018)
15. Wang, H., Zhou, Z.H., Zhang, Y.: Dimensionality reduction using Hilbert-schmidt independence criterion. In: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, pp. 656-663 (2009)