frontiers in Computational Neuroscience

ORIGINAL RESEARCH
published: 27 December 2016
doi: 10.3389/fncom.2016.00131

Representational Distance Learning for Deep Neural Networks

Patrick McClure and Nikolaus Kriegeskorte*

MRC Cognition and Brain Sciences Unit, Cambridge, UK

Deep neural networks (DNNs) provide useful models of visual representational transformations. We present a method that enables a DNN (student) to learn from the internal representational spaces of a reference model (teacher), which could be another DNN or, in the future, a biological brain. Representational spaces of the student and the teacher are characterized by representational distance matrices (RDMs). We propose representational distance learning (RDL), a stochastic gradient descent method that drives the RDMs of the student to approximate the RDMs of the teacher. We demonstrate that RDL is competitive with other transfer learning techniques for two publicly available benchmark computer vision datasets (MNIST and CIFAR-100), while allowing for architectural differences between student and teacher. By pulling the student's RDMs toward those of the teacher, RDL significantly improved visual classification performance when compared to baseline networks that did not use transfer learning. In the future, RDL may enable combined supervised training of deep neural networks using task constraints (e.g., images and category labels) and constraints from brain-activity measurements, so as to build models that replicate the internal representational spaces of biological brains.

Keywords: neural networks, transfer learning, distance matrices, visual perception, computational neuroscience

OPEN ACCESS

Edited by:
Marcel Van Gerven,
Radboud University Nijmegen, Netherlands

Reviewed by:
Michael Hanke,
Otto-von-Guericke University Magdeburg, Germany
Iris I. A. Groen,
National Institutes of Health (NIH), USA

****Correspondence:***
Patrick McClure
patrick.mcclure@mrc-cbu.cam.ac.uk

Received: *21 July 2016*
Accepted: *29 November 2016*
Published: *27 December 2016*

Citation:
McClure P and Kriegeskorte N (2016) Representational Distance Learning for Deep Neural Networks. Front. Comput. Neurosci. 10:131. doi: 10.3389/fncom.2016.00131

1. INTRODUCTION

Deep neural networks (DNNs) have recently been highly successful for machine perception, particularly in the areas of computer vision using convolutional neural networks (CNNs) (Krizhevsky et al., 2012) and speech recognition using recurrent neural networks (RNNs) (Deng et al., 2013). The success of these methods depends on their ability to learn good, hierarchical representations for these tasks (Bengio, 2012). DNNs have not only been useful in achieving engineering goals, but also as models of computations in biological brains. Several studies have shown that DNNs trained only to perform object recognition learn representations that are similar to those found in the human ventral stream (Khaligh-Razavi and Kriegeskorte, 2014; Yamins et al., 2014; Güçlü and van Gerven, 2015). The models benefit from task training, which helps determine the large number of parameters and bring the domain knowledge required for feats of intelligence such as object recognition into the models. This is in contrast to the earlier approach in visual computational neuroscience of using nonlinear systems identification techniques to set the parameters exclusively on the basis of measured neural responses to large sets of stimuli (Naselaris et al., 2011). The latter approach is challenging for deep neural networks, because the high cost of brain-activity measurement limits the amount of data that can be acquired (Yamins and DiCarlo, 2016). Ultimately, task-based constraints will have to be combined with constraints from brain-activity measurements to model information processing in biological brains.

Here we propose a method that enables the training of DNNs with combined constraints on the desired outputs and the internal representations. We demonstrate the method by using another neural net model as the reference system whose internal representations the DNN is to emulate. One method for doing so would be to have a layer in a DNN linearly predict individual measured responses (e.g., fMRI voxels or neurons), and backpropagate the error derivatives from the linear measured-response predictors into the DNN. However, the linear measurement prediction model has a large number of parameters ($n_{units} \times n_{responses}$). An alternative approach is to constrain the DNN to replicate the representational distance matrices (RDMs) estimated from brain responses. In this paper, we take a step in that direction by considering the problem of training a DNN (student) to model the sequence of representational transformations in another artificial system (teacher), a CNN trained on different data.

Our technique falls in the class of transfer learning methods. In the deep learning literature, several such techniques have been proposed both for pulling a DNN's internal representations toward the task target and for transferring knowledge from a teacher DNN to a student DNN. We begin by briefly considering the previous approaches used to accomplish these goals.

1.1. Auxiliary Classifiers: Pulling Internal Representations Toward the Desired Output

Recently, it has been investigated how the error signal reaching an internal layer through backpropagation can be complemented by auxiliary error functions. These more directly constrain internal representations using auxiliary optimization goals. A variety of methods using auxiliary error functions to pull representations toward the desired output have been proposed.

Weston et al. (2012) proposed semi-supervised embeddings to augment the error from the output layer. A reference embedding of the inputs was used to guide representational learning. The embedding constraint was implemented in different ways: inside the network as a layer, as part of the output layer, or as an auxiliary error function that directly affected a particular hidden layer. Weston et al. discussed a variety of embedding methods that could be used, including multidimensional scaling (MDS) (Kruskal, 1964) and Laplacian Eigenmaps (Belkin and Niyogi, 2003). The addition of these semi-supervised error functions led to increased accuracy compared to DNNs trained using output layer backpropagation alone.

Lee et al. (2014) also showed that auxiliary error functions improve DNN representational learning. Instead of using semi-supervised methods, they performed classification with a softmax or L2SVM readout at a given intermediate hidden layer. The softmax layer allowed the output of a network to be treated as a probability distribution by performing normalized exponentiation on the previous layer's activations ($y_i = e^{x_i} / \sum_j e^{x_j}$). The error of the intermediate-level readout was then backpropagated to earlier layers to drive intermediate layers directly toward the target output. The gradients from these classifiers were linearly combined with the gradients from the output layer classifier. This technique resulted in improved accuracies for several datasets.

A challenge in training very deep networks is the problem of vanishing gradients. Layers far from the output may receive only a weak learning signal via conventional backpropagation. Auxiliary error functions were successfully applied to these very deep networks by Szegedy et al. (2015) to inject a complementary learning signal at internal layers by constraining representations to better discriminate between classes. This was implemented in a very large CNN which won the ILSVRC14 classification competition (Russakovsky et al., 2014). In this DNN, two auxiliary networks were used to directly backpropagate from two intermediate layers back through the main network. Similar to the method used in Lee et al. (2014), the parameters for the layers in the main network directly connected to auxiliary networks were updated using a linear combination of the backpropagated gradients from later layers and the auxiliary network.

Wang et al. (2015) investigated the effectiveness of auxiliary error functions in very large CNNs and their optimal placement. They selected where to place these auxiliary functions by measuring the average magnitude of the conventional backpropagation error signal at each layer. Auxiliary networks, similar to those used in Szegedy et al. (2015), were placed after layers with vanishing gradients. These networks consisted of a convolutional layer followed by three fully connected layers and a softmax classifier. As in Lee et al. (2014) and Szegedy et al. (2015), the auxiliary gradients were linearly combined to update the model parameters. Adding these supervised auxiliary error functions led to an improved accuracy for two very large datasets, ILSVRC12 (Russakovsky et al., 2014) and MIT Places (Zhou et al., 2014).

1.2. Transfer Learning: Pulling the Representations of a Student Toward Those of a Teacher

Enabling a student network to learn from a teacher is useful for a number of tasks, for instance model compression (also known as knowledge distillation) and transfer learning (Bengio, 2012). The goal in either case is to use the representational knowledge learned by a teacher neural network to improve the performance of a student network (Bucilua et al., 2006; Ba and Caruana, 2014; Hinton et al., 2015). For model compression, the teacher is a larger or more complex network with higher performance than the student. For knowledge transfer, the representations learned by the teacher network are used to improve the training of a student network on a different tasks or using different data. Several techniques have been proposed for performing these methods.

One technique for model compression is to have the student learn the output representation of the teacher for a given training input. For classification, the neurons before the softmax layer can be constrained to have the same values as the teacher using mean squared error (MSE) as done in Bucilua et al. (2006); Ba and Caruana (2014). Alternatively, the output of the softmax layer can be constrained to represent the same, or similar, output distribution as the teacher. This can be done by minimizing the

cross-entropy between the output distributions of the teacher and student networks for the training inputs (Hinton et al., 2015). However, these techniques assume that the student is learning the same task as the teacher.

Knowledge from different networks can also be transferred at internal layers. Romero et al. (2014) proposed a method for transferring the knowledge of a wide and shallow teacher to a thin and deep student, called FitNet. Pre-trained a network by constraining an intermediate layer of the student network to have representations that could linearly predict "hints" from the teacher network (i.e., activation patterns at a corresponding layer in the teacher network). After this, the network was fine-tuined using the technique proposed in Hinton et al. (2015). The FitNet method was shown to improve the students classification accuracy.

Another prominent technique for performing transfer learning is to initialize the weights of the student network to those of the teacher. The network is then trained on a different task or using different data. This can lead to improved network performance (Yosinski et al., 2014). However, this requires that the teacher and student have the same, or very similar, architectures, which may not be desirable, especially if the teacher is a biological neural network.

In this paper, we introduce an auxiliary error function that enables a student network to learn from the internal representational spaces of a teacher that has a similar or different architecture. The method constrains the student's representational distances in a set of layers to approximate those of the teacher. The student can thus learn the computational transformations discovered by the teacher, leading to improved representational learning during training.

2. METHODS

Our method, representational distance learning (RDL), enables DNNs to learn from the representations of other models to improve performance. As in Lee et al. (2014), Szegedy et al. (2015), and Wang et al. (2015), we utilize auxiliary error functions to train internal layers directly in conjunction with the error from the output layer found via backpropagation. We propose an error function that maximizes the similarity between the representational spaces of a student DNN and that of a teacher model.

2.1. Representational Distance Matrices

In order to compare the representational spaces of models, a method must be used to describe them. As discussed in Weston et al. (2012), a representational space can be characterized by the pairwise distances between representations. This idea has been used in several methods such as MDS, which seeks to reduce the dimensionality of data while minimizing the error between the pairwise distance matrix of the original data and the reduced dimensionality data (Kruskal, 1964).

Kriegeskorte et al. (2008) proposed using the matrix of pairwise dissimilarities between representations of different inputs, which they called representational distance, or dissimilarity, matrices (RDMs), to compare computational models and neurological data. More recently, Khaligh-Razavi and Kriegeskorte (2014) used this technique to analyze several computer vision models, including the CNN proposed in Krizhevsky et al. (2012), and neurological data. Any distance function could be used to compute the pairwise dissimilarities, for instance the Euclidean or correlation distances. An RDM for a DNN can be defined by:

$$RDM(X; f_m)_{i,j} = d(f_m(x_i; W_m), f_m(x_j; W_m)) \quad (1)$$

where X is a set of n inputs (e.g., a mini-batch or a subset of a mini-batch), f_m is the neuron activations at layer m, x_i, and x_j are single inputs, W_m is the weights of the neural network up to layer m, and some distance, or dissimilarity, measure d.

In addition to characterizing the information present in a particular layer of a DNN, RDMs can be used to visualize the representational space of a layer in a DNN (**Figure 1**). information captured by internal layers in a DNN is challenging. Zeiler and Fergus (2014) proposed a method for visualizing the input features which active internal neurons at varying layers using deconvolutional neural networks. Yosinski et al. (2015) also proposed methods for visualizing the activations of a DNNs for a given input. However, these methods do not show the categorical information of each representational layer. Visualizing the similarity of labeled inputs at layers of interest, via an RDM, allow clusters inherent to the learned representational transformations to be viewed.

2.2. Representational Distance Learning

RDL uses an auxiliary error functions that maximizes the similarity between the RDMs of a student and the RDMs of a teacher at several layers. This is motivated by the idea that RDMs, or distance matrices in general, can characterize the representational space of a model. DNNs seek to learn a set of hierarchical representations. For classification, this culminates in finding a representational space where different classes are separable. RDL allows a DNN to learn from the representations of a different, potentially better, model by maximizing the similarity between the RDMs of the DNN being trained and the target model at several layers. Unlike in Bucilua et al. (2006), Ba and Caruana (2014), and Hinton et al. (2015). RDL not only directly trains the output representation, but also the representations of hidden layers. As discussed in Bengio (2012), however, large datasets can prohibit the use of pairwise techniques, since the number of comparisons grows quadratically with dataset size. To address this, our technique only uses a random subset of all pairwise distances for each parameter update. This allows the speed of our method to be constrained by the subset size and not the overall number of training examples, which is usually several orders of magnitude larger.

In order to maximize the similarity between the RDM of a DNN layer being trained and a target RDM, we propose minimizing the mean squared error between the two RDMs. This corresponds to making all possible pairwise distances as similar as possible:

$$E_{aux}(X; f_m; T_m) = \frac{2}{n(n-1)} \sum_{(i,j)|i<j} (RDM(X; f_m)_{i,j} - T_{m,i,j})^2 \quad (2)$$

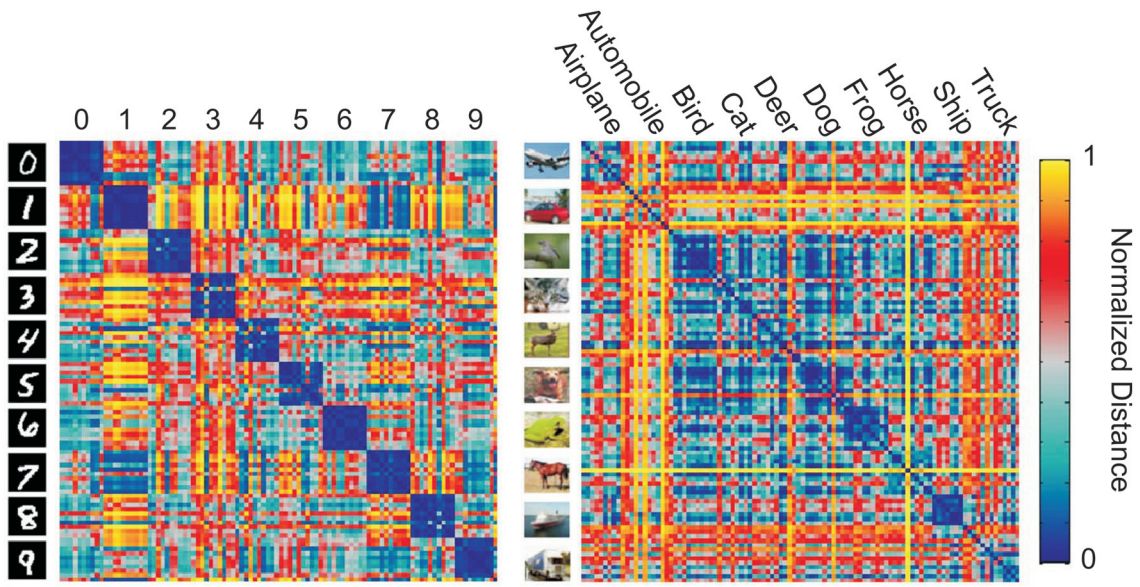

FIGURE 1 | Example representational distance matrices (RDMs) of the output layer of convolutional neural networks (CNNs) for 10 random images of each class from MNIST (left) and CIFAR-10 **(right)** made using the RSA toolbox (Nili et al., 2014).

TABLE 1 | The convolutional neural network (CNN) architecture used for MNIST.

Layer	Kernel size	Number of features	Stride	Non-linearity	Other
Conv-1	5 × 5	32	1	ReLU	–
MaxPool-1	3 × 3	32	3	Max	–
Conv-2	5 × 5	64	1	ReLU	–
MaxPool-2	2 × 2	64	2	Max	–
FC	1500	200	–	ReLU	Dropout ($p = 0.5$)
Linear	200	10	–	–	–

TABLE 2 | The McNemar exact test p-values for the tested CNNs trained on MNIST.

	Baseline	Teacher	Finetuning	Deep supervision	Hints	RDL
Baseline	–	0.38	0.00 ↑	0.11	0.34	0.01 ↑
Teacher	0.38	–	0.01 ↑	0.66	0.89	0.20
Finetuning	0.00 ←	0.01 ←	–	0.14	0.04 ←	0.63
Deep Supervision	0.11	0.66	0.14	–	0.64	0.39
Hints	0.34	0.89	0.04 ↑	0.64	–	0.17
RDL	0.01 ←	0.20	0.63	0.39	0.17	–

Arrows indicate a significant difference ($p < 0.05$, uncorr.) and point to the better model.

where X is a set of n inputs (e.g., a mini-batch or a subset of a mini-batch), f_m is the neuron activations at layer m, and $T_{m,i,j}$ is the distance between the teacher's representations of input x_i and input x_j at layer m. The function d used to calculate the RDMs (Equation 1) could be any dissimilarity or distance function, but we chose to use the mean squared error (MSE). This results in the average auxiliary error with respect to neuron k of f_m, $f_{m,k}$, for input x_i and the weights of the neural network up to layer m, W_m, being defined as:

$$\frac{\partial E_{aux}(x_i; X; f_m; T_m)}{\partial f_{m,k}} = \frac{8}{n(n-1)} \sum_{j|j \neq i} (RDM(X; f_m)_{i,j} - T_{m,i,j})(f_{m,k}|_{x_j}^{x_i}) \quad (3)$$

where $f_{m,k}|_{x_j}^{x_i} = f_{m,k}(x_i; W_m) - f_{m,k}(x_j; W_m)$.

However, calculating the error for every pairwise distance can be computational expensive, so we estimate the error using a random subset, P, of the pairwise distances for each update of a network's parameters. This leads to the auxiliary error gradient being approximated by:

$$\frac{\partial E_{aux}(x_i; X; f_m; T_m)}{\partial f_{m,k}} \approx \frac{8}{|X_P||P_{x_i}|} \sum_{(i,j) \in P_{x_i}} (RDM(X; f_m)_{i,j} - T_{m,i,j})(f_{m,k}|_{x_j}^{x_i}) \quad (4)$$

where X_P is the set of all images contained in P, P_{x_i} is the set of all pairs, (i, j), in P that include input x_i and another input, x_j. If an image is not sampled, its auxiliary error is zero.

The total error of $f_{m,k}$ for input x_i is calculated by taking a linear combination of the auxiliary error at layer m and the error from backpropagation of the output error function and any later auxiliary functions. These terms are combined using weighting hyper parameter α, similar to the method discussed in Lee et al. (2014), Szegedy et al. (2015), and Wang et al. (2015). In RDL,

α is the weight of the RDL error in the overall error function. Subsequently, the error gradient at a layer with an auxiliary error function is defined as:

$$\frac{\partial E_{total}(x_i; y_i; X; f_m; T_m)}{\partial f_{m,k}} = \frac{\partial E_{backprop}(x_i; y_i; f_m)}{\partial f_{m,k}} + \alpha \frac{\partial E_{aux}(x_i; X; f_m; T_m)}{\partial f_{m,k}} \quad (5)$$

This error is then used to calculate the error of earlier layers in the DNN using backpropagation. As discussed by Lee et al. (2014) and Wang et al. (2015), the value of α was decayed as training progressed. Throughout training, α was updated following $\alpha_{t+1} = \alpha_0{}^*(1 - t/t_{max})$ where t is the epoch number and t_{max} is the total number of epochs. By using this decay rule, the auxiliary error function initially helps drive the parameters to good values while allowing the DNN to converge predominantly using the output error by the end of training.

3. RESULTS

To evaluate the effectiveness of RDL, we perform two experiments using four different datasets, MNIST, InfiMNIST, CIFAR-10, and CIFAR-100. For each experiment, we transferred the knowledge of a teacher network trained on a separate dataset to a student network with the a similar architecture using: (1) finetuning after directly copying the weights of the teacher, (2) pre-training an internal layer of the student to linearly predict a corresponding layer in the teacher using "hints," and (3) using RDL. We compared the results to two non-transfer learning networks, a network only constrained at the output layer using the target labels and a deeply supervised network, which constrained both the output layer and internal layers using the target labels. We implemented all of these methods using Torch (Collobert et al., 2011). These experiments show that the knowledge stored in the weights of a teacher network can be transferred to a student network using the representational distances learned by a teacher trained on a related task.

3.1. MNIST

MNIST is a dataset of 28 × 28 images of handwritten digits from 10 classes, 0 through 9 (LeCun et al., 1998). The dataset contains 60,000 training images and 10,000 test images. A 10,000 image subset of the training data was used as a validation set for hyper-parameter tuning. No pre-processing or data augmentation was applied. InfiMNIST is a dataset that extends the MNIST dataset using pseudo-random deformations and translations (Loosli et al., 2007). The first 10,000 non-MNIST InfiMNIST examples were used as a validation set and the next 120,000 examples were used as a training set for the teacher network. Each tested network had the same architecture (**Table 1**), excluding any auxiliary error functions. The deeply supervised network had linear auxiliary softmax classifiers placed after the max pooling layers and α was decayed using $\alpha_{t+1} = \alpha_t^*0.1^*(1 - t/t_{max})$, as proposed in Lee et al. (2014). For the finetuning network, the weights

TABLE 3 | Test errors for MNIST trained convolutional neural networks (CNNs) and the CIFAR-100 trained "Network in Network" (NiN) models.

Method	Error (%)
MNIST	
Baseline CNN	0.63
Teacher	0.56
Teacher with finetuning	0.48
Student with deep supervision	0.55
Student with hints	0.56
Student with RDL	0.49
CIFAR-100	
Baseline NiN	30.68
Teacher with finetuning	38.75
Student with deep supervision	29.46
Student with hints	29.37
Student with RDL	28.77

The performance of the teacher for the CIFAR-100 classification is not shown, since it was trained on CIFAR-10 and, therefore, predicted across 10 not 100 classes, making it unable to perform the CIFAR-100 task.

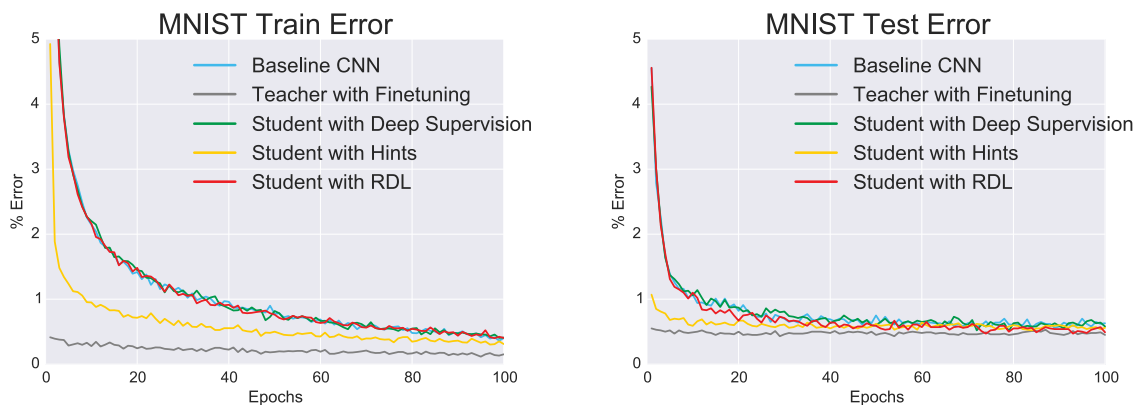

FIGURE 2 | The change in the train and test errors through time as the tested convolutional neural networks (CNNs) are trained on MNIST.

were initialized as the weights of the teacher network instead of being randomly initialized. After this, the network was trained normally. The RDL network had auxiliary error functions after both max pooling layers and the fully connected layer. 5% (500) of the image pairs per mini-batch were used to calculate the RDL auxiliary errors. A momentum of 0.9 and a mini-batch size of 100 were used for all networks trained on MNIST and InfiMNIST.

In addition to the classification error (**Figure 2** and **Table 3**), we used the McNemar exact test (Edwards, 1948) to evaluate whether a network was significantly more accurate in classifying

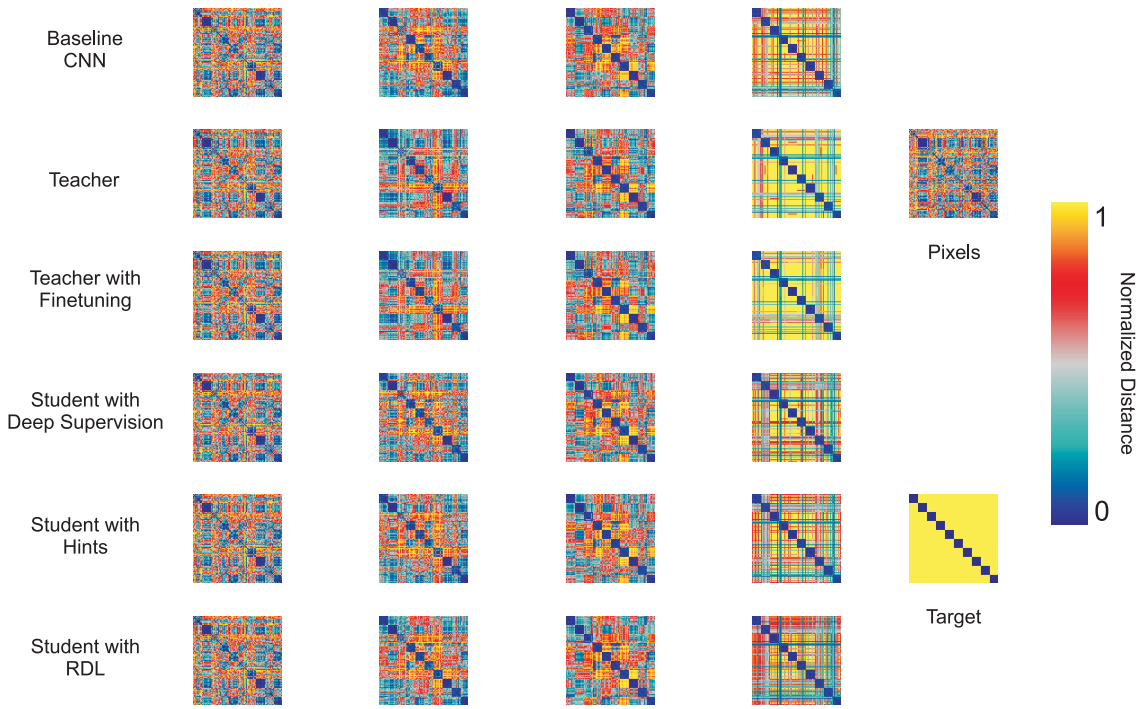

FIGURE 3 | Representational distance matrices (RDMs) using the Euclidean distance for the first and second convolutional layers as well as the fully connected (FC) and softmax layers of the CNN tested methods, the raw pixel data, and the target labels for 10 random class exemplars from MNIST.

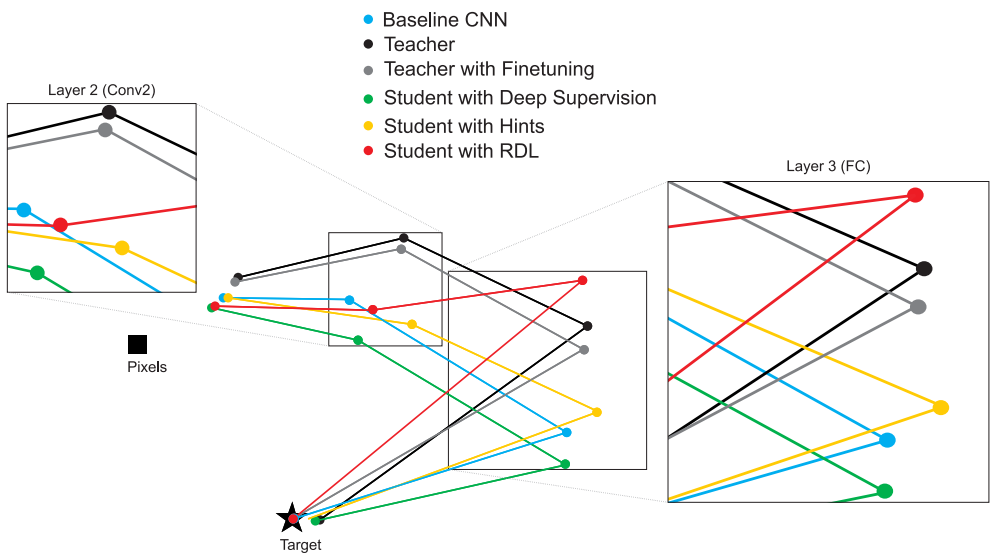

FIGURE 4 | 2-D multi-dimensional scaling (MDS) visualization of the distances between the representational distance matrices (RDMs) for selected layers of the MNIST trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the correlation distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot.

a random image from the distribution from which the images in the training and test sets were drawn. The results (**Table 2**) show that the finetuning and RDL methods both signifantly improve accuracy compared to the baseline CNN. They are, however, not significantly different, showing the ability of RDL to indirectly transfer the knowledge of the teacher network. The finetuned network is also significantly better than the teacher and the "hint" network, unlike RDL. This is because RDL actively constrains the student network to imitate the teacher, while finetuning only affects initialization.

In order to further compare the trained networks, RDMs were generated for each fully trained model. **Figure 3** shows RDMs for 100 random test images, 10 from each class. This visualization emphasizes the class clustering as inputs are transformed from pixel space to label space. Some classes are already clustered in pixel space. For instance, 1, 7, and 9 s each have large blocks along the diagonal portion of the pixel RDM. However, by looking at the rows and columns we can see that these classes are difficult to separate from one another. After the first convolutional layer, class clustering increases, especially for the baseline CNN. After the second convolutional layer, class clustering increases for every model and other class relationships become apparent. For instance, 3 and 5 s are becoming increasingly different from other classes, but are still similar to each other. Also, 1s remain similar to many other classes. The fully connected (FC) layer leads to stronger, but not perfect, class cluster. As expected, the softmax layer leads to extremely strong class distinction. However, most of the models still view 1s as similar to other classes, as seen by the large horizontal and vertical gray stripes. The notable exception is the finetuned CNN, which had the lowest testing error.

While viewing the RDMs directly can make certain facts about the transformations performed by the models evident, it can be hard to compare RDMs to each other by visual inspection. To better understand the relationships between the representations of the different models, we calculate the correlation distance between each pair of RDMs and use MDS to create a 2-D plot showing the relative position in representational space of the transformations learned by the various trained networks (**Figure 4**). This allows for drawing several qualitative conclusions. As expected, the RDMs of the networks start close to the pixel-based RDM and become more similar to the target RDM the deeper the layer. The differences between the evaluated techniques can most clearly be seen at the 2nd (Conv2) and 3rd (FC) layers. As expected: (1) the network initialized with the weights of the teacher and then finetuned has the most similar RDMs to the teacher, (2) deep supervision pulls the RDMs of the student toward the target, (3) RDL pulls the RDMs of the student toward and the RDMs of the teacher, especially at 3rd layer.

TABLE 4 | The "Network in Network" (NiN) architecture with batch-normalization (BN) (Ioffe and Szegedy, 2015) used for CIFAR-100.

Layer	Kernel size	Number of features	Stride	Non-linearity	Other
Conv-1	5 × 5	192	1	ReLU	BN
MLPConv-1-1	1 × 1	160	1	ReLU	BN
MLPConv-1-2	1 × 1	96	1	ReLU	BN
MaxPool	3 × 3	96	2	Max	–
Conv-2	5 × 5	192	1	ReLU	BN, Dropout ($p = 0.5$)
MLPConv-2-1	1 × 1	192	1	ReLU	BN
MLPConv-2-2	1 × 1	192	1	ReLU	BN
AveragePool-1	3 × 3	192	2	–	–
Conv-3	5 × 5	192	1	ReLU	BN, Dropout ($p = 0.5$)
MLPConv-3-1	1 × 1	192	1	ReLU	BN
MLPConv-3-2	1 × 1	100	1	ReLU	BN
AveragePool-2	8 × 8	100	–	–	–

3.2. CIFAR-100

In order to test RDL on a more interesting problem, we performed transfer learning from CIFAR-10 to CIFAR-100. This experiment consists of transferring knowledge learned in an easier task to a harder one, something that is useful in many instances. CIFAR-100 is a dataset of 32 × 32 color images each containing one of 100 objects. The dataset contains 50,000 training images and 10,000 test images. A 10,000 image subset of the training data was used as a validation set for hyper-parameter

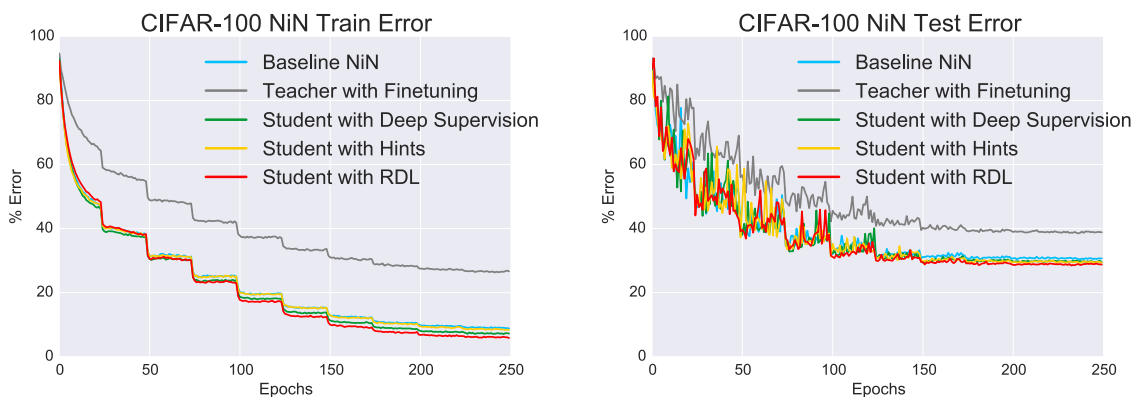

FIGURE 5 | The change in the train and test errors through time for the "Network in Network" (NiN) models trained on CIFAR-100.

tuning. CIFAR-10 is also a dataset of 32 × 32 color images, but containing only 10 distinct classes instead of 100. CIFAR-10 also contains 50,000 training images and 10,000 test images. For both datasets, the data were pre-processed using global contrast normalization. During training, random horizontal flips of the images were performed and the learning rate was halved every 25 epochs.

To evaluate using RDL with a more complex network, we used a "Network in Network" (NiN) architecture (Lin et al., 2013), which use MLPConv layers, convolutional layers that use multi-layered perception (MLP) filters instead of linear filters (**Table 4**).

TABLE 5 | The McNemar exact test *p*-values for the tested "Network in Network" (NiN) models trained on CIFAR-100.

	Baseline	Finetuning	Deep supervision	Hints	RDL
Baseline	—	0.00 ←	0.00 ↑	0.00 ↑	0.00 ↑
Finetuning	0.00 ↑	—	0.00 ↑	0.00 ↑	0.00 ↑
Deep supervision	0.00 ←	0.00 ←	—	0.86	0.08
Hints	0.00 ←	0.00 ←	0.86	—	0.05
RDL	0.00 ←	0.00 ←	0.08	0.05	—

Arrows indicate a significant difference ($p < 0.05$,uncorr.) and point to the better model.

The CIFAR-10 trained teacher network had the same architecture as the baseline CIFAR-100 NiN (**Table 4**) except with a 10-class output layer and had a testing error of 8.0%. The DSN had linear auxiliary softmax classifiers after the first and second pooling layers and α was decayed as proposed in Lee et al. (2014). The finetuning network's weights were initialized using those of the CIFAR-10 teacher network and a linear readout was added. The RDL network had the same architecture as the baseline CIFAR-100 network with randomly initialized weights and the addition of auxiliary error functions that used the RDMs from the CIFAR-10 teacher. For RDL, an additional linear readout was added after the last MLPConv layer since RDL does not specify that each neuron in a representation corresponds to an output class. For RDL, 2.5% (406) of the image pairs per mini-batch of 128 images were used to calculate the RDL auxiliary errors.

As in the previous experiment, the performances of the networks (**Figure 5** and **Table 3**) were statistically compared using the McNemar test. The results are shown in **Table 5**. Unlike in the MNIST experiment, the fine tuned network performed statistically worse than all tested methods. This is likely a combination of the weights being overspecialized for CIFAR-10 classification and the last MLPConv layer having less units. The networks that were trained with deep supervision, hints, and RDL all significantly improved upon the baseline NiN and the

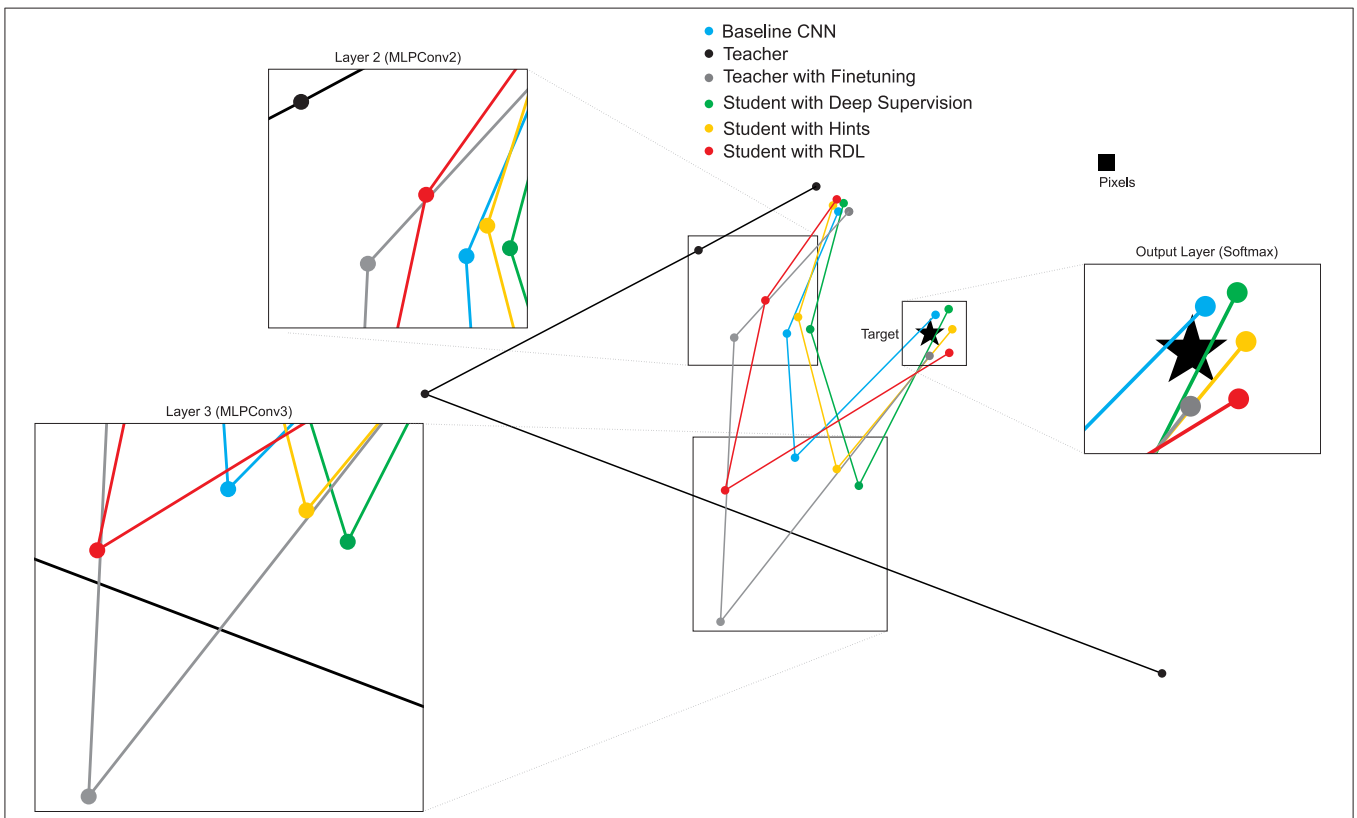

FIGURE 6 | 2-D multi-dimensional scaling (MDS) visualization of the distances between the representational distance matrices (RDMs) for selected layers of the CIFAR-100 trained networks. RDMs were generated for each model using 20 bootstrapped samples of 100 images from the test set. For each sampled image set, the normalized Euclidean distance between the RDMs of the different networks were calculated. These values were then averaged to generate the MDS plot.

finetuned network. These results show that learning from RDMs can extract meaningful information from a teacher network, which leads to improved classification performance.

To investigate the relationships between the representations of the different NiN models, we calculate the correlation between each pair of RDMs and use MDS to create a 2-D plot showing the relative position in representational space of the transformations learned by the various trained networks (**Figure 6**). The MDS plots shows that: (1) the layer 2 and layer 3 RDMs of the network initialized with the weights of the teacher and then finetuned are further from the target than the other non-teacher networks, (2) deep supervision pulls the RDMs of the student toward the target, (3) despite learning a series of transformations that do not map directly to the target, the teacher contains useful information to the students' task, and (4) RDL pulls the RDMs of the student toward and the RDMs of the teacher. This shows the ability of RDL to incorporate both the representational information from the teacher as well as from the classification task.

4. DISCUSSION

In this paper, we proposed RDL, a technique for transferring knowledge from a teacher model to a student DNN. The representational space of the student is pulled toward that of a teacher model during training using stochastic gradient descent. This was performed by minimizing the difference between the pairwise distances between representations of two models at selected layers using auxiliary error functions. Training with RDL was shown to improve classification performance by extracting knowledge from another model trained on a similar task, while allowing architectural differences between the student and teacher. This suggests that RDL can transfer the relationships between class examples learned by the teacher. This information is not present when only constraining internal layers using class labels, as done in the deeply supervised method, since the target vectors for each class are orthogonal. In particular, RDL allows a student network to learn similar sequential transformations to those learned by a teacher network. This could be of potential use in learning transformations similar to those performed in the human visual ventral stream. Such a model might be able to generate brain-like RDMs for novel stimuli. In the future, we plan to train such a model by constraining large DNNs using fMRI-based RDMs from the human visual ventral stream. By learning from brain-activity patterns, RDL has the potential to help build more realistic models of computations in biological brains.

AUTHOR CONTRIBUTIONS

NK and PM conceived of RDL. PM and NK developed the method. PM implemented RDL and performed the training and validation. PM and NK wrote the paper.

ACKNOWLEDGMENTS

This research was funded by the Cambridge Commonwealth, European & International Trust, the UK Medical Research Council (Program MC-A060-5PR20), and a European Research Council Starting Grant (ERC-2010-StG 261352).

REFERENCES

Ba, J., and Caruana, R. (2014). "Do deep nets really need to be deep?" in *Advances in Neural Information Processing Systems* (Montréal, QC), 2654–2662.

Belkin, M., and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* 15, 1373–1396. doi: 10.1162/089976603321780317

Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. *Unsupervised Transfer Learn. Challeng. Machine Learn.* 27, 17–36.

Bucilua, C., Caruana, R., and Niculescu-Mizil, A. (2006). "Model compression," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Philadelphia, PA), 535–541.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). "Torch7: a matlab-like environment for machine learning," in *BigLearn, NIPS Workshop* (Granada).

Deng, L., Hinton, G., and Kingsbury, B. (2013). "New types of deep neural network learning for speech recognition and related applications: an overview," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Vancouver, BC), 8599–8603.

Edwards, A. L. (1948). Note on the correction for continuity in testing the significance of the difference between correlated proportions. *Psychometrika* 13, 185–187.

Güçlü, U., and van Gerven, M. A. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *J. Neurosci.* 35, 10005–10014. doi: 10.1523/JNEUROSCI.5023-14.2015

Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. arXiv:1503.02531.

Ioffe, S. and Szegedy, C. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of The 32nd International Conference on Machine Learning* (Lille), 448–456.

Khaligh-Razavi, S.-M., and Kriegeskorte, N. (2014). Deep supervised, but not unsupervised, models may explain it cortical representation. *PLoS Computat. Biol.* 10:e1003915. doi: 10.1371/journal.pcbi.1003915

Kriegeskorte, N., Mur, M., and Bandettini, P. A. (2008). Representational similarity analysis-connecting the branches of systems neuroscience. *Front. Sys. Neurosci.* 2:4. doi: 10.3389/neuro.06.004.2008

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (Lake Tahoe, NV), 1097–1105.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika* 29, 1–27.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 2278–2324.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2014). Deeply-supervised nets. arXiv:1409.5185.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. arXiv:1312.4400.

Loosli, G., Canu, S., and Bottou, L. (2007). "Training invariant support vector machines using selective sampling," in *Large Scale Kernel Machines*, eds L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (Cambridge, MA: MIT Press), 301–320.

Naselaris, T., Kay, K. N., Nishimoto, S., and Gallant, J. L. (2011). Encoding and decoding in fmri. *Neuroimage* 56, 400–410. doi: 10.1016/j.neuroimage.2010.07.073

Nili, H., Wingfield, C., Walther, A., Su, L., Marslen-Wilson, W., and Kriegeskorte, N. (2014). A toolbox for representational similarity analysis. *PLoS Computat. Biol.* 10:e1003553. doi: 10.1371/journal.pcbi.1003553

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., and Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv:1412.6550.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2014). Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision* 115, 211–252. doi: 10.1007/s11263-015-0816-y

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (Boston, MA), 1–9.

Wang, L., Lee, C.-Y., Tu, Z., and Lazebnik, S. (2015). Training deeper convolutional networks with deep supervision. arXiv:1505.02496.

Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, eds G. Montavon, G. B. Orr, and K.-R. Müller (Heidelberg: Springer), 639–655.

Yamins, D. L. and DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nat. Neurosci.* 19, 356–365. doi: 10.1038/nn.4244

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proc. Natl. Acad. Sci. U.S.A.* 111, 8619–8624. doi: 10.1073/pnas.1403112111

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems* (Montréal, QC), 3320–3328.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv:1506.06579.

Zeiler, M. D., and Fergus, R. (2014). "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*, eds D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Cham: Springer), 818–833.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems* (Montréal, QC), 487–495.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.