IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 8, 2020, accepted July 1, 2020, date of publication July 6, 2020, date of current version July 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007337

Radial Basis Function Networks for Convolutional Neural Networks to Learn Similarity Distance Metric and Improve Interpretability

MOHAMMADREZA AMIRIAN[1,2] **AND FRIEDHELM SCHWENKER**[2]**, (Member, IEEE)**
[1]Institute of Applied Information Technology, Zurich University of Applied Sciences, 8400 Winterthur, Switzerland
[2]Institute of Neural Information Processing, Ulm University, 89081 Ulm, Germany

Corresponding author: Mohammadreza Amirian (amir@zhaw.ch)

ABSTRACT Radial basis function neural networks (RBFs) are prime candidates for pattern classification and regression and have been used extensively in classical machine learning applications. However, RBFs have not been integrated into contemporary deep learning research and computer vision using conventional convolutional neural networks (CNNs) due to their lack of adaptability with modern architectures. In this paper, we adapt RBF networks as a classifier on top of CNNs by modifying the training process and introducing a new activation function to train modern vision architectures end-to-end for image classification. The specific architecture of RBFs enables the learning of a similarity distance metric to compare and find similar and dissimilar images. Furthermore, we demonstrate that using an RBF classifier on top of any CNN architecture provides new human-interpretable insights about the decision-making process of the models. Finally, we successfully apply RBFs to a range of CNN architectures and evaluate the results on benchmark computer vision datasets.

INDEX TERMS Radial basis function neural networks (RBFs), convolutional neural networks (CNNs), CNN-RBFs, supervised learning, unsupervised learning, similarity distance metric.

I. INTRODUCTION

Inspired by the locally tuned response of biological neurons, Broomhead and Lowe introduced radial basis function neural networks (RBFs) in 1988 [1]. The modeling concept behind RBFs is a combination of unsupervised and supervised learning for pattern classification and regression. However, RBFs have not been integrated into the contemporary approaches to computer vision using convolutional neural networks (CNNs) so far due to structural deficiencies. This paper presents developments in a new area of research and lays the foundation for using RBFs in deep learning and computer vision by modifying their architecture and learning process as well as providing open-source code[1]. The results demonstrate that integrating RBFs into CNN models for computer vision provides both a similarity distance metric as well as an interpretable decision-making process.

This research is motivated by the unique opportunities the RBF architectures introduce when used with CNN models. A new training process introduced for RBFs in this paper provides the opportunity of using labeled and unlabeled data by optimizing two loss functions combining supervised and unsupervised learning. The training process of RBF architectures employs a distance metric optimization that we propose to use as a similarity distance metric to find similar and dissimilar images. Additionally, this research proposes visualization techniques to illustrate the clusters and activations with training and test images to gain more insight into the reasoning behind the decisions made by the networks, thus improving interpretability. The contributions of this paper to computer vision literature can be summarized as follows:

- Combining supervised and unsupervised learning.
- Learning a similarity distance metric to find similar images.
- Improving the interpretability of decision-making.

Despite the advantages of combining RBFs to modern CNN architectures, there are two factors in the architecture and training process of RBFs hindering their integration into CNNs. First, the nonlinear activations and computational graphs of RBFs used in the literature prevent efficient gradient flow. Secondly, RBFs assume that the training features are fixed and the cluster centers are initialized accordingly.

The associate editor coordinating the review of this manuscript and approving it for publication was Bin Xu.

[1]https://github.com/moamirian/CNN_RBFs.git

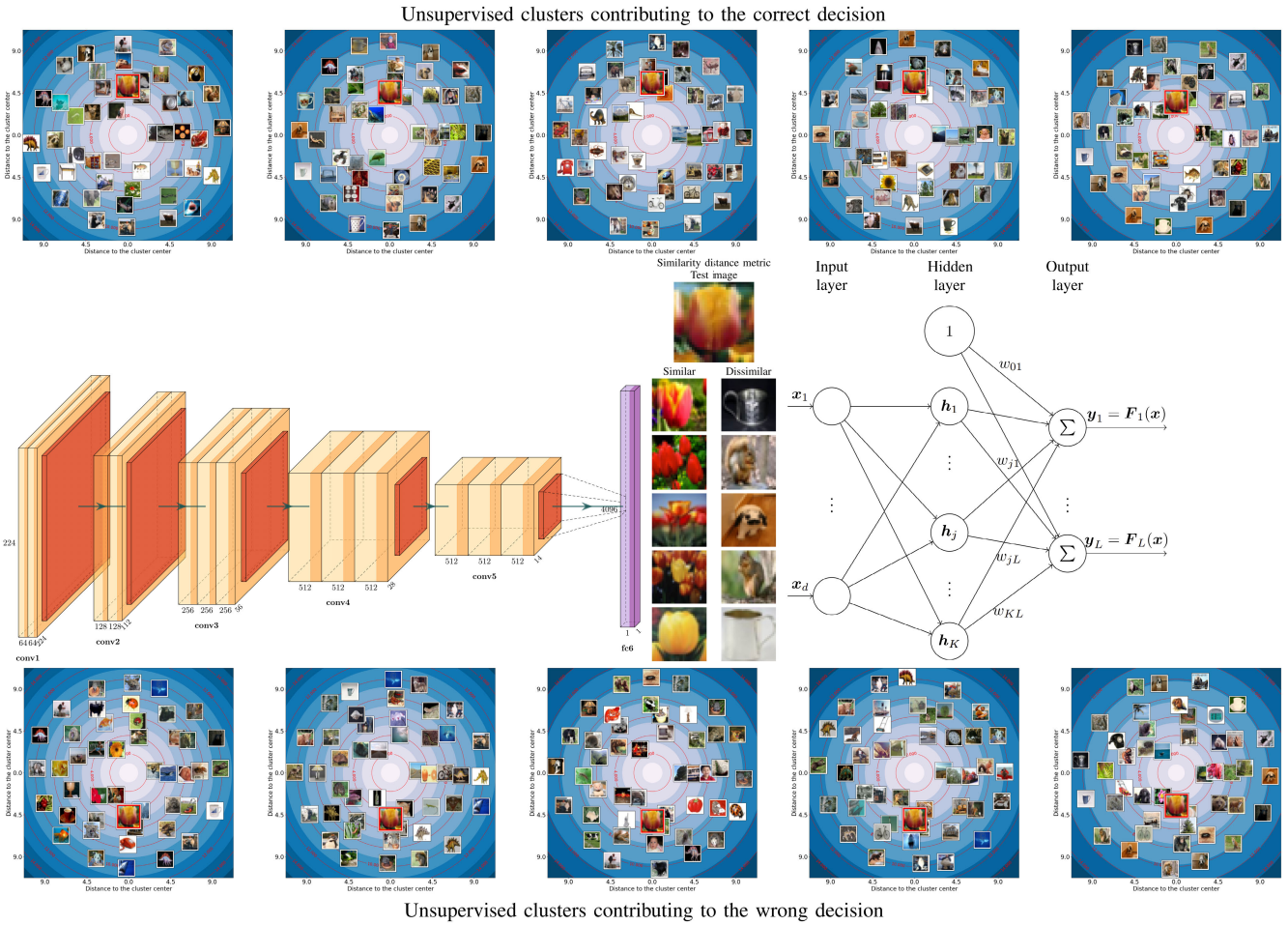

FIGURE 1. A visual summary of the paper: Figures on the top and bottom rows visualize the position of a test image in the unsupervised clusters. The output of a CNN backbone is connected to the input of an RBF through a fully connected layer. The input features of the RBFs are referred to as embeddings in this research work. The embeddings of each image are compared to cluster centers with a trainable similarity distance metric. The same distance metric can be used to find images similar to a test sample amongst training images (visualized in the table in the middle row). The RBFs apply an activation function to the distance of the training images from the cluster centers to compute activation values. The output layer of the RBF is optimized for classification based on these activation values. We train the entire CNN-RBF architecture end-to-end.

Nonetheless, CNN architecture learns the embeddings, which are used as features of RBFs. This paper tackles the limitations of the original RBFs and presents the following contributions to RBF literature:

- Introducing a quadratic activation function and a linear computational graph for end-to-end learning.
- Adding an unsupervised loss term to update the cluster centers in the training process with the learned embeddings.
- Applying the RBFs to computer vision in a first attempt using deep CNN architectures.

The remainder of the paper covers the related works in Section II followed by the theoretical background of RBFs in Section III. We then present our original research and contributions in Section IV with our proposed modifications to RBFs followed by a visual explanation of the new proposed training and decision-making process in Section V. The experimental results of applying the proposed RBF-CNN architectures using a range of CNN backbones on benchmark datasets are presented in Section VI. The potential contributions of our proposed similarity distance metric on the field of computer vision to enhance the transparency of the decision making process is demonstrated in Section VII. We finally present our conclusions in VIII.

II. RELATED WORKS

Research into optimizing RBF architectures has followed two approaches. The first appraoch concentrates on the training process and initialization of the networks while the second approach aims to find better optimized activation functions. This paper presents improvements to both approaches and integrates the RBFs into contemporary vision models using CNNs.

RBFs were originally introduced as a supervised approach for classification and regression tasks. Broomhead and Lowe proposed to draw the cluster centers either from a uniform distribution or randomly from the training samples and then optimizing the output weights using a pseudo-inverse analytic solution [1]. Initializing the cluster centers randomly and only training the output weights is a one-phase training process for

RBFs. Two-phase training for RBFs uses various methods to initialize the cluster centers as well as optimizing the output weights. Research since 1988 has used supervised and unsupervised methods to initialize the centers. Moody and Darken proposed an unsupervised algorithm to initialize these cluster centers [2] while Schwenker *et al.* proposed supervised vector quantization [3]. Decision trees were used to find centers independently by [4] and [5] before training the output weights. Finally, Schwenker *et al.* proposed a third phase to optimize the entire RBF network end-to-end including output weights, cluster center, and parameters of activation functions using gradient descent [6].

All of these methods for cluster center initialization assume a fixed feature space for the input layer. However, CNNs learn the embeddings automatically and develop the feature space of the images during the training process. Therefore, we suggest optimizing an unsupervised learning loss during the training to cope with this change in the feature space. This work differs from previous research as it combines supervised and unsupervised learning by optimizing two separate losses using gradient descent.

Various applications and implementations have motivated several activation functions presented in the literature of RBFs [7]. The Gaussian function is the kernel encouraged by modeling the data through a multivariate Gaussian distribution [1]. Other functions adopted in the RBF architecture include linear kernels, thin-plate splines, logistic functions, and multiquadratic functions [8]–[11]. Hardy's multiquadratic functions motivate an activation function for RBFs used by Karimi *et al.* [12] and Zhao *et al.* [13]. Du *et al.* proposed a kernel for digital signal processing (DSP) units [7]. In this paper, we suggest a quadratic kernel to build a linear computational graph for efficient gradient flow and integrate RBFs for end-to-end training with CNN architectures.

Besides the mature fundamental research, RBFs have been applied to a broad range of applications for pattern classification and regression in recent years. Nicodemou *et al.* used RBF networks for 3D hand pose estimation [14], Dehghan and Mohammadi estimated a numerical solution for Fokker-Planck differential equations with RBFs [15], Li *et al.* used sparse multiscale RBFs for seizure detection in EEG signals [16], and Zhao *et al.* predicted interfacial interactions by training RBFs [13]. Furthermore, Geng *et al.* introduced deep RBF networks and applied the method to food safety inspection data. RBFs are used to train models for classification and regression in discrete and continuous pain quantification [17].

RBFs have been used for computer vision tasks and image classification as well. Friedhelm *et al.* used raw images as feature vectors to classify hand-written digits [6]. Er *et al.* extracted the features from facial images using principle component analysis (PCA) and processed these features using Fisher's linear discriminant (FLD) technique before classifying the patterns using RBFs [18]. However, the successful rise of the modern conventional neural networks, such as LeNet-5 [19] and AlexNet [20], led to a paradigm shift from using hand-crafted features to automated deep convolutional feature and representation learning using CNNs. In recent years, most computer vision tasks, like facial recognition [21], are dominated by modern CNN architectures as they present superior performance compared to classical methods for image processing. To the best of our knowledge, this paper presents the first attempt to integrate RBFs into modern CNN architectures for computer vision.

This research work relates to literature focusing on deep metric learning since RBFs optimize a similarity distance metric automatically during training based on their architecture. Euclidean distance, Mahalanobis distance and cosine similarity have been used to evaluate similarity between the embeddings (extracted features from CNNs) of two images in the literature [22]–[24]. Researchers have applied different strategies and loss functions to optimize these similarity metrics for same-class images while also maximizing the distance of different-class images. The research in this area concentrates on the training process and the design of a loss function which brings similar images closer in the embedding space based on a similarity measure. Hu *et al.* proposed to minimize the inter-class scores and maximize the intra-class scores based on Euclidian distances [25]. Hoffer and Ailon suggested optimizing a similarity-based loss function defined for selected triplets of images [22]. Song *et al.* used the pairwise distances between images of an entire batch and proposed a structured loss function for metric learning [26]. Similar research work aimed at optimizing angular distance, cosine distance and large-margin Euclidean distance of similar and dissimilar samples [24], [27], [28].

This paper presents a method to retrieve a ranked list of similar and dissimilar images, which leads to visually appealing results for similarity metric learning. However, the proposed similarity metric learned by the RBFs does not require any complicated triplet sample section or loss design. The presented results are obtained using a typical supervised loss function for classification (softmax cross-entropy). Furthermore, RBFs can not only optimize for Euclidean and Mahalanobis distances, but also for the entire covariance matrix.

III. RADIAL BASIS FUNCTION NETWORKS

In this section, we briefly review and explain the theoretical foundation of radial basis function networks. RBFs are presented in the literature as a global approximation method for learning a mapping $\boldsymbol{F}$ from a given feature space with dimensionality d to label a space with dimensionality K ($\boldsymbol{F} : I\!R^d \rightarrow I\!R^K$) [1]. In this paper, the function $\boldsymbol{F}$ of features $\boldsymbol{x}$ approximates the one-hot encoded labels $\boldsymbol{y}$. The features used in this work to train the RBFs are the embeddings of deep (CNNs) to predict the class labels using end-to-end optimization. However, we use a fully connected layer between CNN architectures and RBFs to provide compatibility between the two architectures and prevent overfitting. The architecture of the RBF consists of an input layer, a single

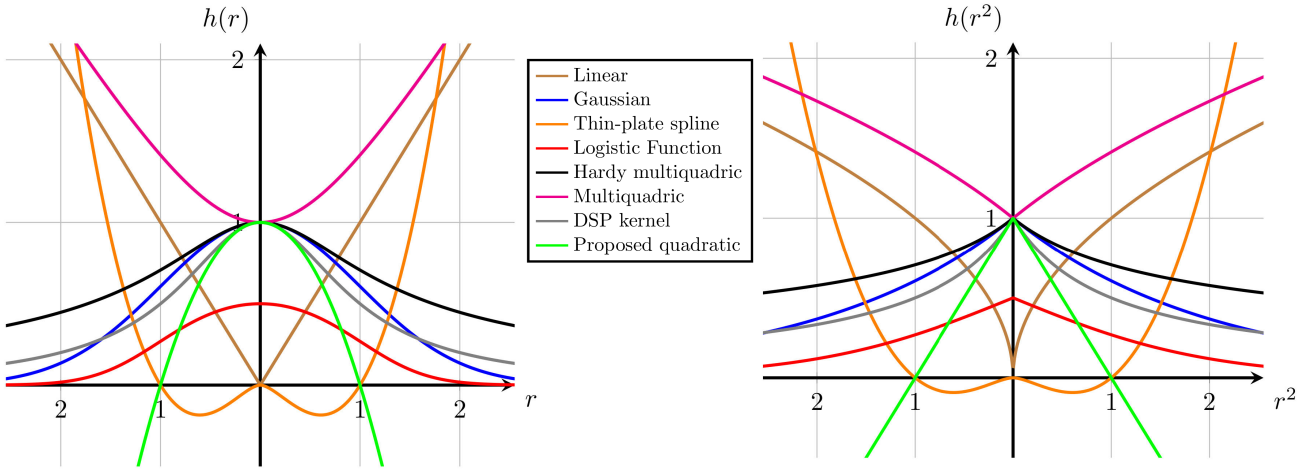

FIGURE 2. Activation functions for RBF networks. We used the following parameters for all of the kernels: $\sigma = 1$, $\alpha = 1/2$, and $\beta = 1/2$. The proposed quadratic activation kernel is linear based on the r^2. Consequently, the CNN goes through a completely linear forward path and thus computes the gradient during backpropagation efficiently.

trainable hidden layer with C cluster centers ($\boldsymbol{c}_j$), and an output layer.

During the evaluation, also termed inference in the deep learning architecture, the RBF computes a distance between embeddings of deep CNNs and the cluster centers and applies an activation function to this distance. The network outputs are then computed by multiplying the weights of the output layer with the activation values. This evaluation process formally defined as:

$$r^2 = \| \boldsymbol{x}^\mu - \boldsymbol{c}_j \|^2_{\boldsymbol{R}_j} = (\boldsymbol{x} - \boldsymbol{c}_j)^T \boldsymbol{R}_j (\boldsymbol{x} - \boldsymbol{c}_j) \tag{1}$$

$$\boldsymbol{y}_k = \boldsymbol{F}_k(\boldsymbol{x}) = \sum_{j=1}^{C} w_{jk} h(\| \boldsymbol{x}^\mu - \boldsymbol{c}_j \|^2_{\boldsymbol{R}_j}) + w_{0k} \tag{2}$$

where r represents the distance, R_j is the positive definite covariance matrix (trainable distance), T denotes the matrix transposition, w_{jk} shows the weights of the output layers, h is the activation function, and w_{0k} are the biases. In these equations, μ, j, and k enumerate the number of samples, cluster centers, and classes, respectively. Trainable parameters in Equation 1 and 2 are the output weights, cluster centers, and covariance matrix.

Optimizing the RBF networks with an identity covariance matrix results in training in Euclidean space. It is possible to optimize a Mahalanobis distance [29] by training the main diagonal on the covariance matrix. Any arbitrary distance metric can be trained by optimizing the entire covariance matrix while projecting the matrix to the space of positive definite matrices. The distance r computed in Equation 1 is not only a measure of the proximity of an image to a cluster center but can also be used to compare images and find similar and dissimilar images in the embeddings space.

The linear and nonlinear activation functions used in RBFs are as follows [9]–[11]:

$$\text{Linear :} \quad h(r) = r \tag{3}$$

$$\text{Gaussian :} \quad h(r) = e^{-r^2/2\sigma^2} \tag{4}$$

$$\text{Thin-plate spline :} \quad h(r) = r^2 \ln r \tag{5}$$

$$\text{Logistic function :} \quad h(r) = \frac{1}{1 + e^{(r^2 - r_0^2)/\sigma^2}} \tag{6}$$

$$h(r) = \frac{1}{(r^2 + \sigma^2)^\alpha}, \quad \alpha > 0 \tag{7}$$

$$h(r) = (r^2 + \sigma^2)^\beta, \quad 0 < \beta < 1 \tag{8}$$

$$h(r) = \frac{1}{1 + r^2/\sigma^2} \tag{9}$$

In addition to the standard machine learning activation kernels in equations 3-6, the kernel presented in Equation 7 is derived from the generalized Hardy's multiquadratic function [8] and results in the same function for $\alpha = 1/2$. Du *et al.* [7] proposed the kernel in Equation 9 because of its convenience for implementation on DSP units. Various activation functions for RBF are depicted in Figure 2.

The complete process of training RBFs was introduced by Schwenker *et al.* [6] as three phase process:

Unsupervised Learning: This step is aimed at finding cluster centers that are representative of the data. The k-means [30] clustering algorithm is widely used for this purpose. k-means iteratively finds a set of cluster centers and minimizes the overall distance between cluster centers and members over the entire dataset. The target of the k-means algorithm can be written in the following form:

$$\text{Loss}_{unsupervised} = \sum_{j=1}^{K} \sum_{\boldsymbol{x}^\mu \in \vartheta_j} \| \boldsymbol{x}^\mu - \boldsymbol{c}_j \|^2 \tag{10}$$

where $\boldsymbol{x}^\mu \in \vartheta_j$ denotes the members of the j^{th} cluster shown by ϑ_j.

Computing Weights: The output weights of an RBF network can be computed using a closed-form solution. The matrix of activation of the samples is defined from the training set (H) as follows:

$$\boldsymbol{H} = h(\| \boldsymbol{x}^\mu - \boldsymbol{c}_j \|^2_{\boldsymbol{R}_j})_{\mu=1,\ldots,M, j=1,\ldots,C} \tag{11}$$

Based on Equation 2, the matrix of output weights (W), which estimates the matrix of labels (Y), is computed by the

following equation:

$$\boldsymbol{Y} \approx \boldsymbol{M}\boldsymbol{W} \Rightarrow \boldsymbol{W} \approx \boldsymbol{M}^{\dagger}\boldsymbol{Y} \tag{12}$$

where $\boldsymbol{M}^{\dagger}$ is the Moore–Penrose pseudo-inverse matrix [31] of $\boldsymbol{H}$ and is computed as:

$$\boldsymbol{H}^{\dagger} = \lim_{\alpha \to 0^{+}} (\boldsymbol{H}^{T}\boldsymbol{H} + \alpha \boldsymbol{I})^{-1}\boldsymbol{H}^{T} \tag{13}$$

End-to-End Optimization: After initializing the RBF weights and cluster centers with a clustering algorithms such as k-means, it is possible to optimize the network end-to-end via backpropagation and gradient descent. Schwenker *et al.* computed the gradients of the loss function for a Gaussian activation function in [6].

IV. ADAPTING RBFs FOR CNNs

In this section, we propose using RBF classifiers on top of CNNs as depicted in Figure 1. The deep embeddings are computed using standard convolutional layers and inception blocks. The deep embedding of the CNNs are flattened and fed to an RBF after using a fully connected layer in the architecture. The network ends with an output layer with softmax activation and is optimized end-to-end. Integrating the RBFs into deep structures and using them in conjunction with CNNs presents three challenges:

Initialization: Training the RBFs from scratch with randomly initialized weights using gradient descent is quite inefficient due to inappropriate initial cluster centers. The large initial distances in high dimensional spaces leads to small activation values and the gradients attenuate considerably after the RBF hidden layer during backpropagation. Therefore, we use the k-means algorithm to initialize the cluster centers before starting the training. Furthermore, computing the weights from Equation 12 is not feasible at the scale of computer vision problems such as ImageNet [32], which has over 14 million images and 1000 classes. Hence, we randomly initialize the output layer and optimize it using gradient descent.

Dynamic Input Features: The input features of classical RBFs are fixed, but this assumption is not valid with respect to CNNs. As the embeddings of CNNs develop during the training process, the cluster centers initialized by the k-means algorithm are no longer optimal after a few epochs of training. We propose to optimize the k-means algorithm target with the unsupervised loss defined in Equation 10 during the training process.

Activation: The nonlinear computational graph drawn by computing the distance in Equation 1 and applying the activations in equations 3-9 leads to inefficient gradient flow. We target this challenge by introducing a new activation function.

A. INTRODUCING UNSUPERVISED LEARNING LOSS

In this section, we explain two modifications to classical RBFs to make them suitable for deep CNNs. First, we introduce an additional loss term to the RBF hidden layer. This term is based on the target function of the k-means algorithm defined in Equation 10 and continues in the unsupervised learning process during the development of embeddings. Secondly, we introduce a new quadratic kernel to build a linear computational graph for efficient optimization using backpropagation.

The embeddings of CNNs change during the training process and necessitates updating the cluster centers with an unsupervised loss. We introduce an additional term to the networks loss function based on the unsupervised target in Section III to optimize the cluster centers during training using the k-means loss in Equation 10.

The final loss of a CNN with RBFs as the classifier (CNN-RBFs) is computed as

$$\text{Loss}_{\text{rbf}} = \text{Loss}_{\text{supervised}} + \lambda \times \text{Loss}_{\text{unsupervised}} \tag{14}$$

where the classification loss $\text{Loss}_{\text{classification}}$ is any arbitrary loss function, for instance categorical cross entropy.

It is conventional to use clustering algorithms such as the k-means or expectation-maximization (EM) algorithms to initialize the cluster centers. The loss function in Equation 14 is optimized using gradient descent by minimizing the distance of the embeddings for each sample from its nearest cluster center regardless of the class labels. The distance from the nearest cluster center is computed from the distance metric $\boldsymbol{R_j}$ defined in Equation 1.

B. QUADRATIC KERNEL

The kernels used for classical RBFs are nonlinear, increasing model complexity. The architectures proposed in Figure 1 profit from utilizing the state-of-the-art in representation learning, i.e. CNNs, as a backbone. Therefore, CNN-RBF architectures can be trained with simpler linear models to improve the gradient flow during backpropagation. The proposed quadratic activation function is linear in the space of r^2 and is defined as follows:

$$h(r) = 1 - r^2/\sigma^2 \tag{15}$$

where σ is the parameter that determines the width of the kernel. The proposed kernel is depicted in Figure 2 along with the conventional activation kernels. Our proposed quadratic kernel reduces the nonlinearity of the CNN-RBF computational graph for backpropagation. The squares of the distances between cluster centers and samples are computed by linear matrix multiplication in Equation 1 and applying the proposed activation, which is linear for r^2. Thus, the gradients of the deep embeddings propagate backwards through a distance computation with matrix multiplication and linear activations.

V. VISUALIZATION OF TRAINING PROCESS

In this section, we visualize the performance of the RBFs on top of CNNs on a simple dataset. The experiments are conducted on the modified national institute of standards and technology dataset (MNIST) [34], which is a dataset of hand-written digits including 10 classes. Learning the dataset

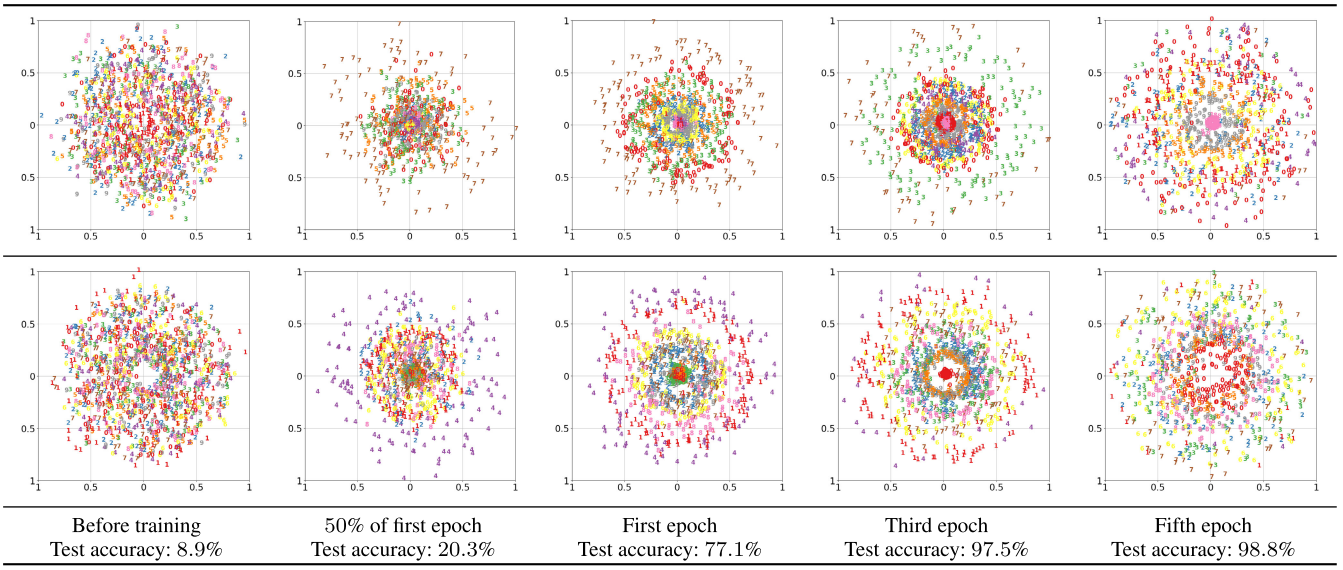

FIGURE 3. The location of data samples compared to the cluster centers during the training process. The centers of the clusters are in the middle of the figures. The training samples are located by a random angle and their distance from the center of the cluster. The vertical and horizontal axes show the normalized distances.

is considered a simple task in computer vision. The simplicity of the dataset and learning task provides us with the opportunity to visualize the training process at a fine level of detail. We chose the same number of cluster centers as classes (10) in the dataset to depict the training process of a CNN-RBF. The network architecture in this section consists of a four-layer CNN and the output of these layers is connected to the RBF after a global average pooling layer and a fully connected layer.

Figure 3 demonstrates the evolution of a representation around the cluster center during the training process. The data samples in this figure are placed according to their distance from the center and at a random angle. The samples are shown with a number denoting their class and are additionally color code accordingly in Figure 3. To reduce the overlap of similar samples, we add random uniformly distributed noise at an amplitude of 0.1 to the distance of the samples from the cluster centers.

Minimizing the unsupervised loss in Equation 10 reduces the distance of the data samples from the cluster centers. Furthermore, the supervised loss enforces the samples of the same class to maintain the same distance from cluster centers, as the activations are the only information for the final decision of the network. The circles with samples of the same class around the cluster centers demonstrate the effect of supervised loss in training. It should be noted that the clusters presented in Figure 3 are selected to optimally illustrate the concepts underlying training CNN-RBFs. Figure 4 illustrates the two-dimensional mapping of the CNN embeddings (top row) and RBF activations (bottom row) using t-SNE [33]. The effect of both supervised and unsupervised loss from Equation 14 is also visible in this figure. The data samples split into clusters regardless of their class labels in the embedding space of CNN due to the unsupervised loss (top row in Figure 4). The activation values divide into clusters corresponding to the class labels, a process encouraged by the supervised loss.

VI. EXPERIMENTAL RESULTS

This section presents the experimental results that reinforce the applicability of RBFs to CNNs. We use several standard computer vision benchmark datasets and investigate the effect of tweaking various hyperparameters of the CNN-RBF architectures in the training phase and generalization to test data. We used three convolutional backbones: those of EfficientNet-B0 [35], InceptionV2 [36], and ResNet50 [37]. A list of the benchmark computer vision datasets is presented in Table 1.

TABLE 1. Computer vision benchmark datasets used to evaluate the performance of CNN-RBFs.

Dataset	Train Size	Test Size	# Classes
CIFAR-10 [38]	50 000	10 000	10
CIFAR-100 [38]	50 000	10 000	100
Oxford-IIIT Pets [39]	3 680	3 369	37
Oxford Flowers [40]	1 020	6 140	102
FGVC Aircraft [41]	6 667	3 333	100
Caltech Birds [42]	5 996	5 794	200

Figure 5 shows the hyperparameter search results for object classification on two benchmark computer vision datasets: CIFAR-10 and CIFAR-100. The backbone CNN model in this experiment is EfficientNet-B0 with RBFs on top of this backbone for classification. The image preprocessing pipeline, called AutoAugment [43], consists of a set of optimal and automatically searched augmentation policies for the ImageNet [32] dataset. The CNN-RBF architecture demonstrated

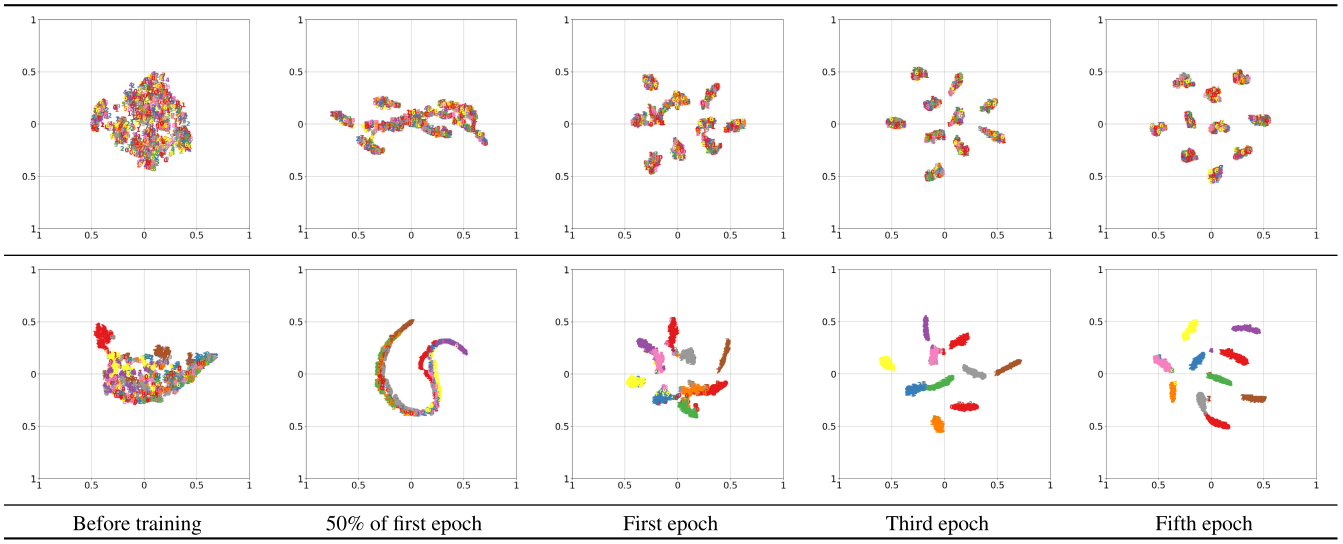

FIGURE 4. The two-dimensional representation of the training process. The figure presents the embeddings of the convolutional backbone (top row) and the activations of the RBFs (bottom row) and then maps them to a two-dimensional space using t-SNE [33]. The vertical and horizontal axes depict the normalized values; however, the unit value is the same in all sub-figures.

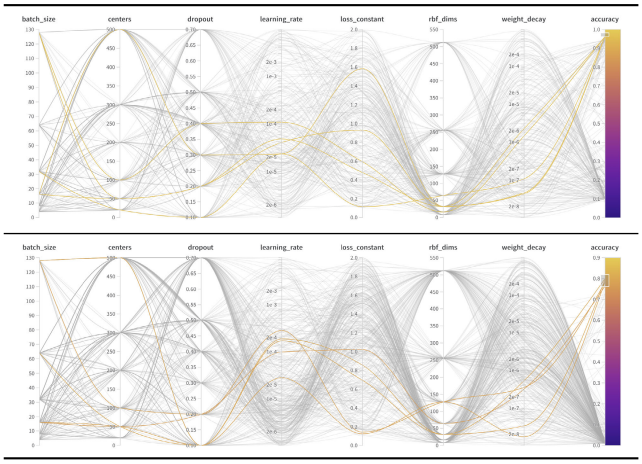

FIGURE 5. Hyperparameter search results from CIFAR-10 (top) and CIFAR-100 (bottom). The top five performing sets of hyperparameters for each dataset are highlighted in yellow.

in Figure 1 has two further hyperparameters: the number of cluster centers and the input dimensions of the RBF network. The models are optimized using an AdamW [44] optimizer with learning rate and weight decay as hyperparameters. The loss constant λ from Equation 14, dropout rate, and batch size are the other hyperparameters.

The hyperparameter searches in Figure 5 are conducted using the hyperband [45] algorithm with 4 agents running in parallel on two *Quadro T2000* graphic processing units (GPUs) for approximately 10 days. It should be noted that dropout is only applied after the CNN backbone and before the fully connected layer in Figure 1. The output of the fully connected layer, without any activation function, is used as the input feature of the RBFs. The results in Figure 5 shows that training CNN-RBF architectures leads to high performances with a wide range of hyperparameters. However, achieving good test performance with a high dropout rate and a large input dimension is challenging. CNN-RBF architectures show a better performance without dropout and rectified linear unit (ReLU) activations in the input layer of RBFs. Thus, we neglect the dropout for further hyperparameter searches conducted on the datasets in Table 1. The list of optimal hyperparameters for all datasets is presented in Table 2.

We also applied CNN-RBFs to several other computer vision datasets with various backbone architectures. The experimental results of applying the CNN-RBFs to the computer vision benchmark datasets with the standard train and test splits are presented in Table 3. CNN-RBFs show the capacity to learn the entire dataset in all of the cases. There is, however, a small gap between the best reported performances in computer vision literature and CNN-RBF architectures. Using dropout with CNN-RBFs for regularization does not lead to desirable results and reducing the number of parameters of the RBFs, while limiting the input size, is the best regularization strategy that we found for RBFs besides data augmentation. Developing regularization methods for RBFs to improve generalization is an open research topic for reducing the gap between our current results and those of state-of-the-art computer vision models.

VII. SIMILARITY METRIC LEARNING AND INTERPRETABILITY OF CNN-RBFs

Using a different approximation strategy compared with fully connected layers provides CNN-RBFs with the chance to probe the decision-making process based on these visual clues:

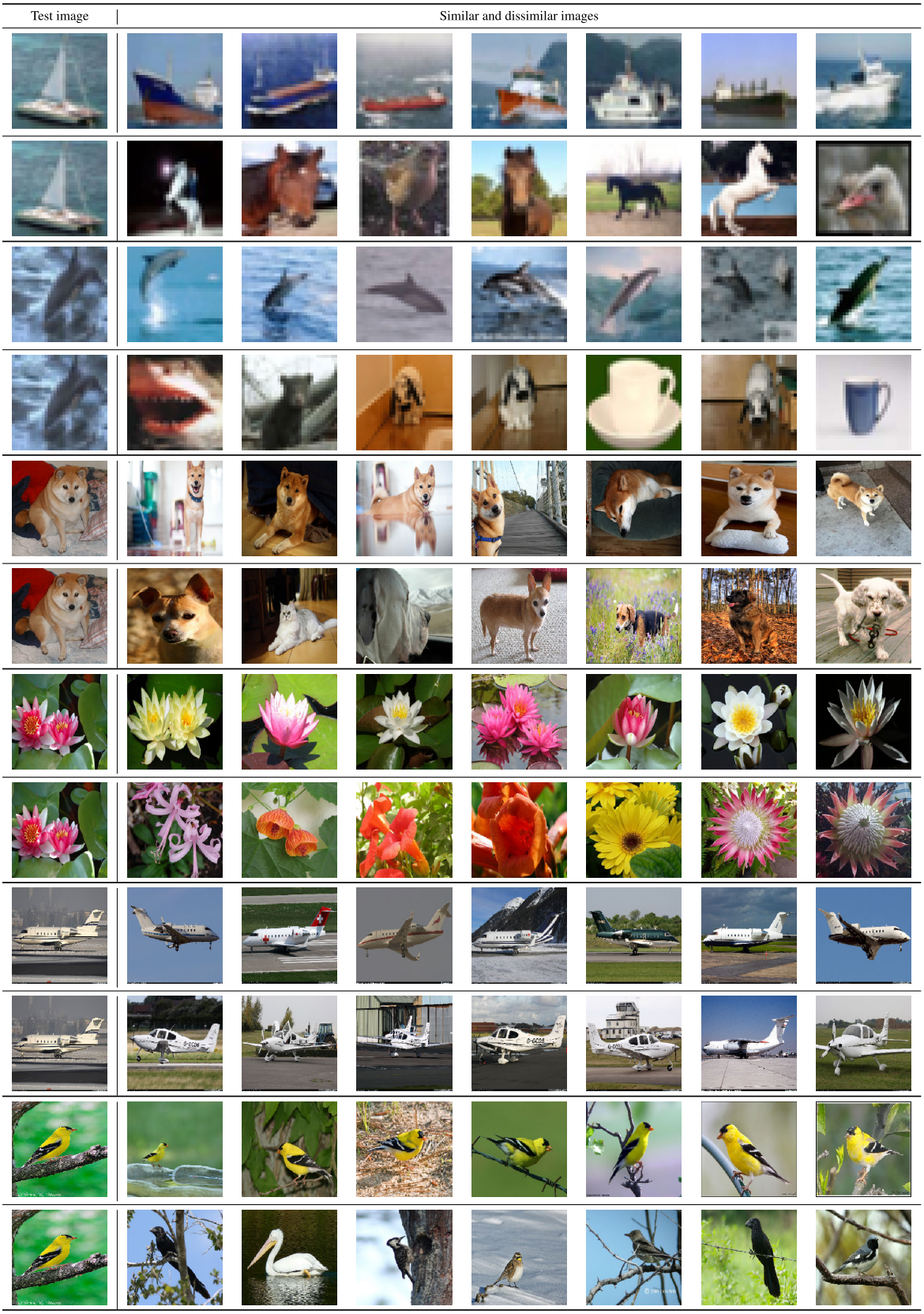

FIGURE 6. Similar and dissimilar training images for given test images based on the similarity metric compute in Equation 16 and 17. The Figure depicts the top 7 most similar and dissimilar training images for a given test image in every two rows. The images shown in every two consecutive rows belong to one of the datasets in Table 1 with the same order.

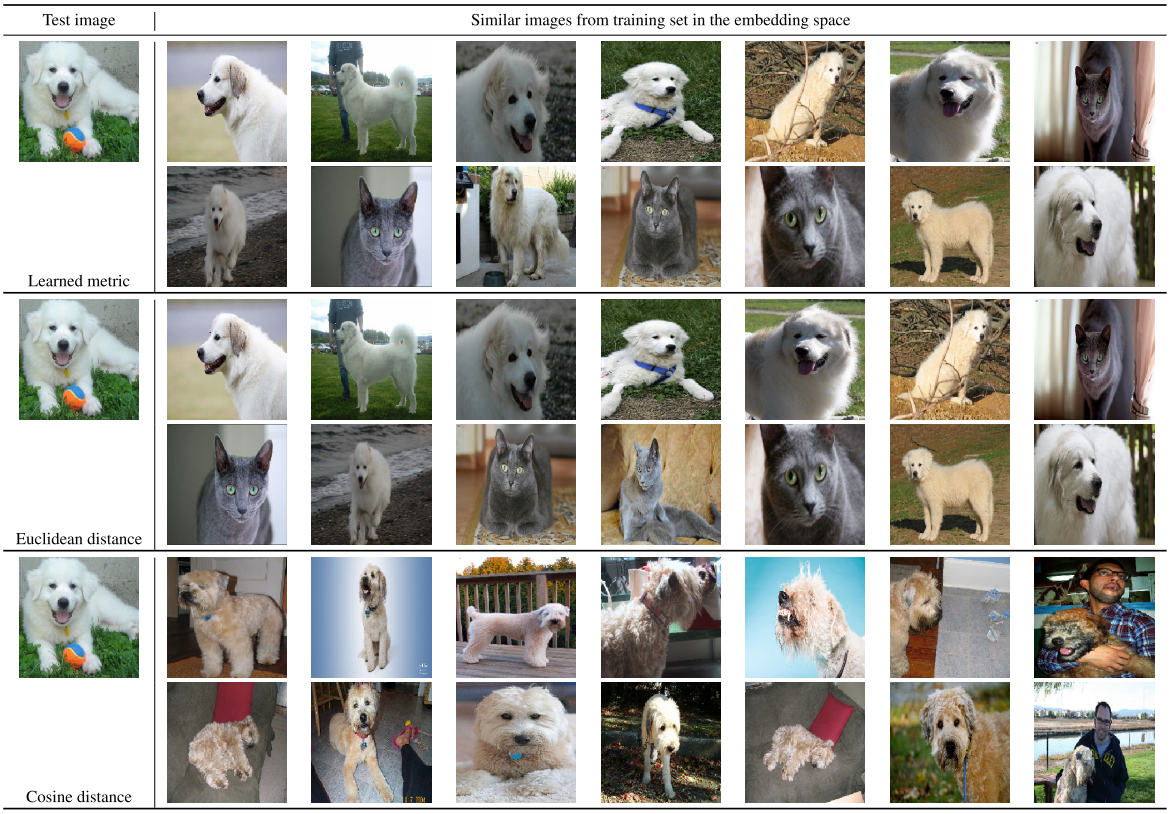

FIGURE 7. Comparing the top 14 images selected using different distance metrics in the embedding space for a given test image.

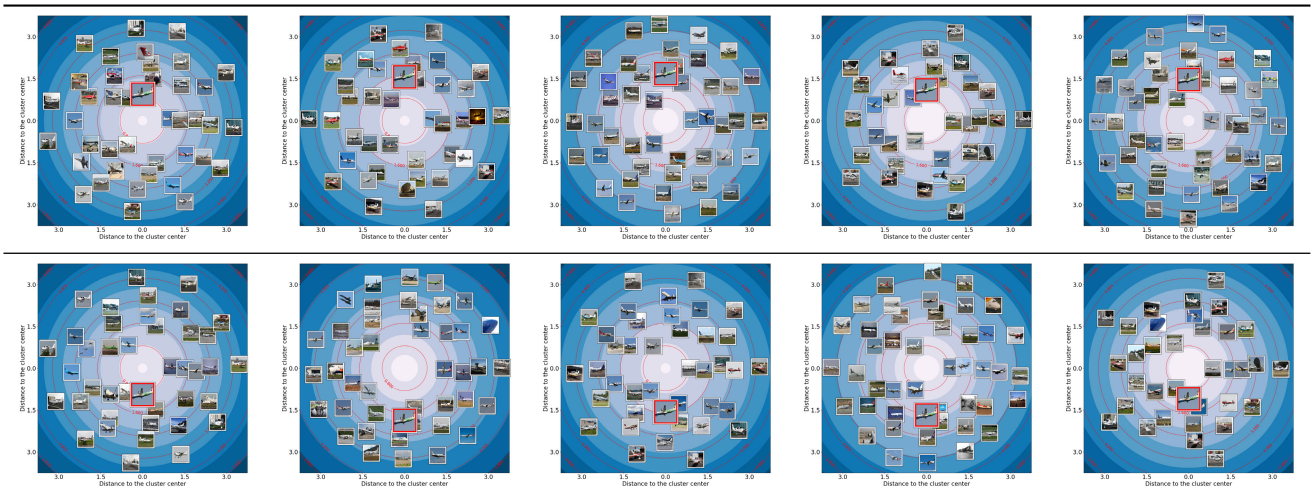

FIGURE 8. The Figure illustrates the clusters contributing to the correct class (top row) and the wrong class (bottom row) of the CNN-RBF network. The test sample is the larger image with red boarded in each cluster representation. Red circles show the distance of the samples to the cluster center, and the background is proportional to the activation values of the cluster. The brighter the activation value, the larger it is and the maximum activation at the cluster center is equal to one.

- Similar images as measured by the similarity distance metric of RBFs trained on CNN embeddings
- Visualizing the clusters with higher contribution to the network's decision and distance of the samples from the centers of these clusters

The embeddings of CNN are evaluated by their learned distance metric from cluster centers in RBFs. The same distance can be used to measure the distance between a test image and similar images from training data. Figure 6 shows the similar images found in the training dataset for a given test sample by the similarity distance metric in Equation 1. The most similar and dissimilar images are computed using the following criteria:

$$\boldsymbol{x}_{similar} = \underset{\boldsymbol{x}_{train}}{\operatorname{argmin}} \parallel \boldsymbol{x}^{\mu}_{train} - \boldsymbol{x}_{test} \parallel^2_{\boldsymbol{R}} \tag{16}$$

$$\boldsymbol{x}_{dissimilar} = \underset{\boldsymbol{x}_{train}}{\operatorname{argmax}} \parallel \boldsymbol{x}^{\mu}_{train} - \boldsymbol{x}_{test} \parallel^2_{\boldsymbol{R}} \tag{17}$$

TABLE 2. List of the final hyperparameters used for each computer vision benchmark dataset to achieve the performance of CNN-RBF architectures.

Dataset	Loss constant	Learning rate	Embeddings dimensions	Batch size	Number of centers	Weight decay
CIFAR-10	0.1141	2.355e-5	64	32	20	1.090e-7
CIFAR-100	0.8557	1.873e-4	32	64	50	5.369e-7
Oxford-IIIT Pets	1.067	7.487e-5	64	16	50	1.150e-7
Oxford Flowers	1.562	1.076e-4	16	64	100	3.843e-6
FGVC Aircraft	0.5471	1.103e-4	8	8	50	1.222e-6
Caltech Birds	0.5156	2.603e-4	32	32	50	1.416e-8

TABLE 3. Comparing the performance of various CNN-RBF architectures with pretraining and augmentation on benchmark computer vision datasets. The best results column is the top performance of the current state-of-the-art architecture on the benchmark dataset.

Dataset	Backbone	CNN-RBFs EfficientNet-B0	CNN-RBFs InceptionV2	CNN-RBFs ResNet50	Best result
CIFAR-10	No-Augment	0.966	0.963	0.969	0.993
	Auto-Augment	0.975	**0.977**	0.942	
CIFAR-100	No-Augment	0.797	0.752	0.693	0.936
	Auto-Augment	**0.822**	0.805	0.778	
Oxford-IIIT Pets	No-Augment	0.840	0.804	0.622	0.967
	Auto-Augment	**0.887**	0.820	0.829	
Oxford Flowers	No-Augment	0.609	0.659	0.595	0.997
	Auto-Augment	**0.828**	0.757	0.667	
FGVC Aircraft	No-Augment	0.723	0.717	0.665	0.945
	Auto-Augment	0.842	**0.843**	0.828	
Caltech Birds	No-Augment	0.613	0.428	0.281	0.904
	Auto-Augment	**0.618**	0.587	0.503	

where x_{test} presents the input of RBFs for a given test image, x^{μ}_{train} shows the input vector for training samples, and μ enumerates the training samples from 1 to N. $\boldsymbol{x}_{simliar}$ and $\boldsymbol{x}_{dissimliar}$ represent the most similar and dissimilar images to the given test image (x_{test}) respectively. We can use the same similarity metrics in Equation 16 and 17 to create a ranked list of similar and dissimilar images for a given test sample.

Figure 7 compares the performance of the similar sample selection for a given test images. The figure suggests that the learned metric and Euclidean distances outperform the cosine distance for similar sample selection. Furthermore the learned metric slightly outperforms the Euclidean distance in this specific case.

The active clusters for every sample provide the reasoning behind the final decision of a CNN-RBF. The clusters can be described by the distance of images from their centers. Figure 8 depicts training samples and their distances from the cluster centers against a test sample. The product of activations and output weights determines the final decision of an RBF. Thus, the importance of clusters for a decision can be determined by sorting the product of activations and class weights. Figure 8 depicts the clusters with the highest contributions to the correct class (ground truth) and the wrong class based on this multiplication product. The wrong class here refers to the class with the second-highest level of confidence.

VIII. CONCLUSION

The research work presents fundamental architectural modifications that are applied to RBFs to integrate them with CNNs for computer vision. The experimental results indicate that the integration of RBFs on top of CNNs achieves competitive performances in benchmark computer vision datasets by combining supervised and unsupervised learning. The proposed activation and training process is compatible with any arbitrary state-of-the-art CNN architecture, including inception blocks and residual connections. The small gap between the CNN-RBFs performance and best CNN models is a subject for future research to find optimal regularization methods for RBF networks. Using RBF architecture with CNNs introduces two unique and network-specific opportunities for learning a similarity distance metric and interpreting the decision-making process in more detail. Similar and dissimilar images found using a similarity distance metric trained by RBFs are interpretable by humans. The cluster representations are currently only used to trace the decision making process as in the current research, the distribution of images around clusters are not visually conclusive due to being optimized in an unsupervised manner regardless of ground-truth labels.

ACKNOWLEDGMENT

The authors would like to thank Thilo Stadelmann, Yvan Putra Satyawan, and Simon Milligan for their suggestions, and grammatical and stylistic edits. This article was made possible by the open source Tensorflow [46] and Keras [47] deep learning libraries, the Weights & Biases toolbox [48][2] for experiment logging, and PlotNeuralNet[3] for neural network visualization. They would also like to thank the significant effort invested into developing these tools.

REFERENCES

[1] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks, complex systems," Roy. Signals Radar Establishment, Malvern, U.K., Tech. Rep., 1988, vol. 2.
[2] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Comput.*, vol. 1, no. 2, pp. 281–294, Jun. 1989.
[3] F. Schwenker, H. A. Kestler, G. Palm, and M. Hoher, "Similarities of LVQ and RBF learning–a survey of learning rules and the application to the classification of signals from high-resolution electrocardiography," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, vol. 1, Oct. 1994, pp. 646–651.
[4] M. Kubat, "Decision trees can initialize radial-basis function networks," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 813–821, Sep. 1998.
[5] F. Schwenker and C. Dietrich, "Initialisation of radial basis function networks using classification trees," *Neural Netw. World*, vol. 10, no. 3, pp. 473–482, 2000.
[6] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, nos. 4–5, pp. 439–458, May 2001.
[7] K.-L. Du, K. K. M. Cheng, and M. N. S. Swamy, "A fast neural beamformer for antenna arrays," in *Proc. IEEE Int. Conf. Commun.*, vol. 1, Apr./May 2002, pp. 139–144.
[8] R. Franke, "A critical comparison of some methods for interpolation of scattered data," Naval Postgraduate School, Monterey CA, USA, Tech. Rep. NPS-53-79-003, 1979.
[9] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proc. IEEE*, vol. 78, no. 9, pp. 1481–1497, Sep. 1990.
[10] Y. Liao, S.-C. Fang, and H. L. W. Nuttle, "Relaxed conditions for radial-basis function networks to be universal approximators," *Neural Netw.*, vol. 16, no. 7, pp. 1019–1028, Sep. 2003.
[11] S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant, "Practical identification of NARMAX models using radial basis functions," *Int. J. Control*, vol. 52, no. 6, pp. 1327–1350, Dec. 1990.

[2]https://www.wandb.com/
[3]https://github.com/HarisIqbal88/PlotNeuralNet

[12] N. Karimi, S. Kazem, D. Ahmadian, H. Adibi, and L. V. Ballestra, "On a generalized Gaussian radial basis function: Analysis and applications," *Eng. Anal. with Boundary Elements*, vol. 112, pp. 46–57, Mar. 2020.
[13] Z. Zhao, Y. Lou, Y. Chen, H. Lin, R. Li, and G. Yu, "Prediction of interfacial interactions related with membrane fouling in a membrane bioreactor based on radial basis function artificial neural network (ANN)," *Bioresource Technol.*, vol. 282, pp. 262–268, Jun. 2019.
[14] V. C. Nicodemou, I. Oikonomidis, and A. Argyros, "Single-shot 3D hand pose estimation using radial basis function networks trained on synthetic data," *Pattern Anal. Appl.*, vol. 23, no. 1, pp. 415–428, Feb. 2020.
[15] M. Dehghan and V. Mohammadi, "The numerical solution of Fokker–Planck equation with radial basis functions (RBFs) based on the meshless technique of Kansa's approach and Galerkin method," *Eng. Anal. Boundary Elements*, vol. 47, pp. 38–63, Oct. 2014.
[16] Y. Li, W.-G. Cui, H. Huang, Y.-Z. Guo, K. Li, and T. Tan, "Epileptic seizure detection in EEG signals using sparse multiscale radial basis function networks and the Fisher vector approach," *Knowl.-Based Syst.*, vol. 164, pp. 96–106, Jan. 2019.
[17] M. Amirian, M. Kächele, and F. Schwenker, "Using radial basis function neural networks for continuous and discrete pain estimation from biophysiological signals," in *Proc. IAPR Workshop Artif. Neural Netw. Pattern Recognit.* Ulm, Germany: Springer, 2016, pp. 269–284.
[18] M. J. Er, S. Wu, J. Lu, and H. Lye Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 697–710, May 2002.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[21] I. Masi, Y. Wu, T. Hassner, and P. Natarajan, "Deep face recognition: A survey," in *Proc. 31st SIBGRAPI Conf. Graph., Patterns Images (SIBGRAPI)*, Oct. 2018, pp. 471–478.
[22] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *Proc. Int. Workshop Similarity-Based Pattern Recognit.* Copenhagen, Denmark: Springer, 2015, pp. 84–92.
[23] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 521–528.
[24] N. Wojke and A. Bewley, "Deep cosine metric learning for person re-identification," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 748–756.
[25] J. Hu, J. Lu, and Y.-P. Tan, "Deep transfer metric learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 325–333.
[26] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4004–4012.
[27] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2593–2601.
[28] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When deep learning meets metric learning: Remote sensing image scene classification via learning discriminative CNNs," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 5, pp. 2811–2821, May 2018.
[29] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The Mahalanobis distance," *Chemometrics Intell. Lab. Syst.*, vol. 50, no. 1, pp. 1–18, 2000.
[30] M. R. Anderberg, *Cluster Analysis for Applications*, vol. 19. New York, NY, USA: Academic, 2014.
[31] R. Penrose, "A generalized inverse for matrices," *Math. Proc. Cambridge Phil. Soc.*, vol. 51, no. 3, pp. 406–413, 1955.
[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[33] L. van der Maaten and G. Hinton, "Visualizing non-metric similarities in multiple maps," *Mach. Learn.*, vol. 87, no. 1, pp. 33–55, Apr. 2012.
[34] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[35] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.
[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 001, 2009. [Online]. Available: https://arxiv.org/pdf/1207.0580.pdf
[39] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3498–3505.
[40] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proc. 6th Indian Conf. Comput. Vis., Graph. Image Process.*, Dec. 2008, pp. 722–729.
[41] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," 2013, *arXiv:1306.5151*. [Online]. Available: http://arxiv.org/abs/1306.5151
[42] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD birds 200," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2010-001, 2010.
[43] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.
[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–18.
[45] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6765–6816, 2017.
[46] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, and S. Ghemawat. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[47] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras
[48] L. Biewald. (2020). *Experiment Tracking With Weights and Biases*. [Online]. Available: https://www.wandb.com/

MOHAMMADREZA AMIRIAN received the M.Sc. degree in communications technology from Ulm University, Ulm, Germany, in 2017, where he is currently pursuing the Ph.D. degree. He is currently working as a Researcher with the Institute of Applied Information Technology (InIT), Zurich University of Applied Sciences (ZHAW), Winterthur, Switzerland. His research interests include biophysiological signal processing for person-centered medical and affective pattern recognition. His current research interests include interpretable deep-learning algorithms for industrial applications and automated deep learning.

FRIEDHELM SCHWENKER (Member, IEEE) received the Diploma and Ph.D. degrees in mathematics and computer science from the University of Osnabrück. He is currently a Privatdozent at the Institute of Neural Information Processing, Ulm University. He has (co-) edited 20 special issues and workshop proceedings published in international journals and publishing companies. He has published more than 200 papers at international conferences and journals. His research interests include artificial neural networks, machine learning, statistical learning theory, data mining, pattern recognition, information fusion and affective computing. He has served as the (Co-) Chair of the IAPR TC3 on Neural Networks and Computational Intelligence. Since 2016, he has been the Chair of the new IAPR TC9 on Pattern Recognition in Human–Computer Interaction.