IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 16, 2019, accepted February 9, 2019, date of publication February 18, 2019, date of current version March 7, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2899901*

# Interpretation of Deep CNN Based on Learning Feature Reconstruction With Feedback Weights

**NAEEM UL ISLAM AND SUKHAN LEE, (Fellow, IEEE)**
College of Information and Communication Engineering, Intelligent Systems Research Institute, Sungkyunkwan University, Suwon 440-746, South Korea
Corresponding author: Sukhan Lee (lsh1@skku.edu)

This work was supported in part by the "3D Recognition Project" of the Korea Evaluation Institute of Industrial Technology (KEIT) under Grant 10060160, and in part by the "Robot Industry Fusion Core Technology Development Project" of KEIT under Grant 10048320, sponsored by the Korea Ministry of Trade, Industry and Energy (MOTIE).

**ABSTRACT** Interpreting a deep Convolutional Neural Network (CNN) involves identifying the features in a hierarchy of layers that contribute to recognition. Although the current approaches serve as methods to interpret a deep CNN, further advancement is required for a more accurate and efficient way of understanding how a hierarchy of features formed by a deep CNN contributes to recognition. In this paper, we propose attaching a feedback CNN to a pretrained feedforward CNN as a means of learning how recognition is performed by the feedforward CNN. In other words, the features reconstructed in a hierarchy of the feedback CNN represent those learned by the feedforward CNN. By analyzing how clusters are formed in the layers of feature spaces in the feedback CNN, we can interpret which features critically contribute to recognition. It also helps to evaluate whether or not recognition is done successfully. In order to show this, we experimentally verify the capabilities of the proposed approach in terms of 1) accurately recovering the ground truth input under data corruption; 2) generating novel input data corresponding to an untrained feature vector without input data iterations; and 3) identifying incorrectly recognized input data by pinpointing the source of the error in feature spaces. The experiments conducted on the ModelNet datasets indicate that the proposed approach offers an extended capability of interpreting a deep CNN as described above with higher accuracy than the conventional approaches.

**INDEX TERMS** Convolutional neural network, feature interpolation, receptive field, response field.

## I. INTRODUCTION

Deep neural networks have achieved impressive performance in numerous vision-related tasks, including object classification [1], [2], detection [3], [4], and image captioning [5]. The success of these deep neural networks has been made possible by three facts: (i) the availability of a large amount of labeled training data, (ii) high computational capabilities in terms of GPUs, and (iii) open source libraries.

Although deep neural networks offer tremendous benefits when these resources are available to them, this end-to-end training process with highly nonlinear functions of deep networks treats them as black boxes which lack proper information about the internal representation of the data. The activities of neurons toward the representation of the internal structure of the data as well as their behavior in terms of collaboration with each other in such complex models are obscure, and the model learning is based on trial-and-error [6]. This is a substantial limitation of deep networks in understanding the classification applications, as it hinders the human experts in carefully verifying the classification decision. Understanding the activities and contributions of neurons in the deep network can be useful for many applications such as scene understanding [7], image segmentation [8], [9], and image style transfer [10]–[12]. The clear semantic in the convolution layer of the deep network can make the prediction more understandable in terms of the contributions of specific features for prediction [13], since simple yes and no answers can be of limited value in certain applications. A more elaborate and clear answer would involve where something occurs or how it is structured, as compared to a binary or real-valued one-dimensional assessment of the mere presence or absence of certain objects at the top layer [14].

In summary, the unreasonable properties of assessing the model based on the binary or real-valued one-dimensional answer at the decision layer make it hard to interpret

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

the activities of neurons at different layers. Furthermore, the black-box nature of deep networks makes it nearly impossible for one to know about the collaboration of neurons or fix problems when errors occur while performing different tasks. Therefore, meaningful interpretation of a deep neural network is required which allows the user to learn the behavior of the network and trust its ability in terms of interacting with the system using deep networks in different applications.

In this paper, our contributions regarding the interpretability of deep neural networks are as follows:

i) Modified Feature Extraction and Reconstruction Network: which learns the local as well as global features in a hierarchical manner, hence making the interpretability of neurons at the different spatial locations and different layers more meaningful.

ii) Response Field-based Reconstruction Algorithm: In order to evaluate the activities of each neuron in the deep convolutional neural network at different layers, we take advantage of the response field-based reconstruction algorithm. The nature of the convolution operation forces the neurons in the response field to collaborate with each other. Therefore, in order to find the activities of a particular neuron at a specific layer, the consideration of the neurons in collaboration with that particular neuron needs to be known. The proposed response field-based reconstruction method finds those collaborations and visualizes the effect in the input space by reconstructing from the specified response field.

iii) Clustering-based Analysis: In order to analyze the deep neural network from the perspective of recognizing the input testing samples as a means of meaningful interpretation, we propose clustering-based analysis in the filter dimensions. The proposed clustering-based analysis effectively evaluates the recognition capability of the deep neural network by reconstructing the classified testing samples from the representative cluster centroids.

iv) Feature Interpolation Algorithm: In order to analyze the deep neural network in terms of the neuron collaboration, we propose a feature interpolation algorithm. The feature interpolation algorithm explores the feature space, where we sample the feature code from a uniform distribution, then use two approaches to select the initial samples for interpolation. In the first approach, we begin by clustering a feature at a specific location in the filter dimension, then find the k-nearest centroids of those clusters to the feature code sampled from the uniform distribution. Then, the nearest feature codes are obtained from the centroids, the response field is filled from those feature codes, and then interpolation is applied between them. In the second approach, rather than clustering a feature in a specific location in the feature space, we find the nearest feature codes to the sampled code obtained from the uniform distribution and filled the receptive field by the proposed response field-based reconstruction algorithm. The selected receptive fields are then used for the inter as well as intraclass interpolation. The interpolation is described in detail in section VIII.

The rest of the paper is organized as follows. Section II discusses the related work, while Section III introduces the proposed feature extraction and reconstruction CNN. In Section IV, detail about the collaboration of neurons in the response field using the response field-based reconstruction algorithm is given. In Section V, we conducted a comparative analysis based on the representation capabilities of Feature Extraction and Reconstruction Convolutional Neural Network (FER-CNN). Section VI discusses the implications of the feedback weights while using the clustering-based approach for interpreting the classification of input testing samples. Section VII shows the comparative analysis based on the classification accuracy of FER-CNN, while Section VIII describes the feature interpolation algorithm. Finally, Section IX concludes our work.

## II. RELATED WORK

In order to investigate the reasons behind the successful training of deep neural networks, their capabilities in terms of the meaningful representation of the input data for successful recognition and collaboration of neurons to a specific input for an efficient visualization plays a key role. A variety of methods based on visualization have been proposed.

### A. GRADIENT BASED ITERATIVE OPTIMIZATION

One of the existing approaches for the interpretation of deep neural networks, based on iterative optimization, has been proposed in [15] and [16]. Simonyan *et al.* [16] used gradient descent optimization of the class score with respect to the input image. In their approach, they maximized the score of a specific class while updating the input image using $L_2$ regularization on the generated image and the initial randomly selected image representation. The approach of Simonyan et al. [16] involves maximizing the class score, which results in the mean representation of that class. The mean representation acts as the candidate of that class; however, in order to make the representation more specific and typical, Mahendran and Vedaldi [15] used image prior and invert a differentiable $\phi$ representation of an image while using gradient descent optimization. They sought an image x$*$ given a feature vector $\phi_0$, by minimizing the Euclidean distance loss between $\phi_0$ and $\phi$(x) and using a regularizer enforcing a natural image prior. Although these iterative optimization approaches allow for an arbitrarily assigned code to be reconstructed, both of these methods involve test time optimizations, resulting in high computation time associated with computing the gradient of the feature representation. The test time per image on GPU is about six seconds, which is very high and considered to be unacceptable for real time applications. By contrast, the proposed approach is only computationally expensive at training time, while the testing is done in real time and requires up to 3ms due to a single pass of the input samples through the network. Secondly, the approach used by Mahendran and Vedaldi requires a hand-designed natural image prior, while in our case the network implicitly learns such a prior. Furthermore, the proposed

approach evaluates the activities of different neurons in different layers, which provides more insight into the interpretation of deep neural networks.

### B. UP-SAMPLING

Separate from the iterative optimization as a means of interpretation, the up-sampling convolution and deconvolution methods have been proposed in [12], [17], and [18]. Springenberg and Dosovitskiy [17] used an up-sampling deconvolutional architecture to invert the features in the upper layer back to the input space. However, due to the non-invertibility of max pooling in the deconvolution architecture, an approximate inversion of the pooling has been proposed in [12] by recording the locations of the maxima within each pooling region in a set of switch variables, then considering those location values during reconstruction in the input space. Similarly, Dosovitskiy *et al.* [18] used up-sampling convolutional layers, rather than deconvolution and max-pooling, so as to avoid the max pooling inversion problem. However, in all of the above-mentioned approaches, the representation was end-to-end, which does not show the activities of each neuron or their collaboration with each other. In the proposed network, the learning rule is based on feature extraction and reconstruction both locally and globally in a hierarchical fashion. This makes it possible to extract useful features, rendering it an efficient way to reconstruct a typical sample from the local layers along with their mean representation from the global layers. Furthermore, the proposed approach investigates the activities of each neuron in different layers without any additional computational burden. The approaches in [17] and [18] do not provide information about the recognition capability of the deep neural network, and only reconstruct the input samples as a means of an end-to-end representation capability of the deep neural network. By contrast, the proposed approach not only reconstructs the input testing samples, but also analyzes its recognition performance based on the reconstructed sample from the representative clusters while using its recognition probabilities.

### C. INPUT CROPPING

In order to investigate the interpretation of neurons at different layers, Bolei *et al.* proposed a receptive field cropping-based method in [19]. They found a receptive field in the input image corresponding to a specific neural activation in the feature map. The receptive field in the input space represents the interpretation of that specific neural activation in the feature space. This method only provides details about the input space without any consideration of the accurate representation of the input samples. For example, if a noisy sample is provided to the network, the activation corresponding to that noisy sample may not be the same, which may result in a wrong receptive field. In addition, this approach does not provide explicit information about the activities of neurons in terms of collaboration towards a specific response, as it does not use the learned parameters for the input representation in the backward pass. By contrast, the proposed approach not only finds the corresponding receptive field, but also shows the group of neurons responsible for the representation of that specific receptive field. In addition, the proposed approach uses the trained parameters to reconstruct the specific feature, and is thus robust to noise. The receptive field-based cropping approach does not provide information about the recognition of the input testing samples, whereas the proposed approach provides meaningful information about the classified testing samples by reconstructing them from their representative cluster centroids.

### D. NORMALIZED CONTRIBUTIONS

A decomposition-based approach for the explanation of deep neural networks has been proposed in [20] in the form of so-called Layer-wise relevance propagation (LRP). LRP is based on pixel-wise decomposition, which aims to understand the contribution of a single pixel of an input image to its corresponding prediction made by a classifier in an image classification task. The evaluation of a deep neural network based on normalized contribution using the class label along with its corresponding input sample is end-to-end and is limited in terms of evaluating the collaboration of neurons in the middle layers. The proposed approach not only analyzes the deep neural network from the classification layer, but also has the capability to reconstruct the input sample from any layer and any spatial location in the feature space, which provides more insight into the behaviors of neurons and their collaborations in different layers. The layer-wise relevance propagation does not explore the cluster space of different class samples in an explicit way; hence, it results in a blurry reconstruction, whereas the proposed approach effectively explores the cluster regions and has the capability to reconstruct the typical samples from the local layers as well as their mean reconstruction from the classification layer. Furthermore, LRP [20] does not provide information about the misclassified testing samples, while the proposed approach provides meaningful information about the recognition of the input testing samples by reconstructing them from their representative cluster centroids.

## III. FEATURE EXTRACTION AND RECONSTRUCTION CNN

For automatic local and global feature extraction along with automatic reconstruction, we use FER-CNN [21] as a basic platform and modify it as shown in Fig. 1 for the purpose of interpretation.

FER-CNN is composed of two sub-networks: the Encoder and Decoder. Since the main purpose of our architecture is to interpret a deep neural network, we modified FER-CNN so that it could perform classification by mapping the 3D input object $x_i$, through a series of layers, to a probability vector $\tilde{y}_i$ over C different classes, along with meaningful reconstruction through the Decoder. Both the Encoder and Decoder are mirrored, where they each consist of five convolution layers with the parameters of $6 \times 6 \times 6$, $5 \times 5 \times 5$, $4 \times 4 \times 4$, $3 \times 3 \times 3$, $2 \times 2 \times 2$, and $1 \times 1 \times 1$, as well as fully connected layers with dimensions of 1500, 500, and 10/40. The filter dimensions are [64, 196, 512, 1024, 2048, 1500, 500] respectively.

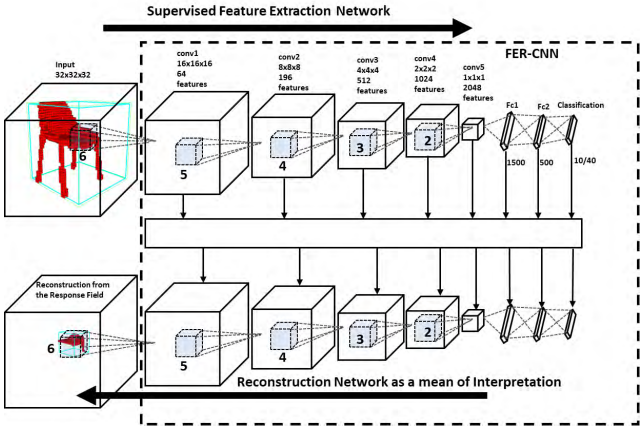


**FIGURE 1.** FER-CNN model, The top represent the encoder part, whereas the bottom represents the decoder part along with skip connections between the encoder and decoder. Both the encoder and decoder have five convolution layers and deconvolution layers, respectively, along with three fully connected layers.

### A. TRAINING DETAILS OF FER-CNN

The training procedure of the proposed architecture is composed of the following three steps:

Step 1: End to end training of the encoder part of FER-CNN by optimizing Softmax classification loss:

$$L_1 = \sum_i L_c(G_{enc}(x_i), T_i), \tag{1}$$

where $y_i = G_{enc}(x_i)$ is the output of the encoder and $T_i$ is the corresponding target for the given input $x_i$. $L_c$ represents the log classification loss for the given $x_i$ and $T_i$, and $L_1$ is the total classification loss over all of the input samples.

Step 2: This step involves independent unsupervised layer-wise training of the decoder weights based on the encoder trained in the step 1, individually from layer 1 to layer 8, using the layer-wise objective function:

$$L_2 = \sum_l L_w \left( \left(y_l - D_{decl}(y_{l+1})\right)^2 + \left(y_{l+1} - G_{encl}\left(D_{decl}\left(y_{l+1}\right)\right)\right)^2 \right), \tag{2}$$

where $l$ represents the $l_{th}$ layer, $y_l$ and $y_{l+1}$ respectively represent the input and the output of the $l_{th}$ layer of the encoder, $G_{encl}$. and $D_{decl}$. respectively represent the outputs of the encoder and decoder at their $l_{th}$ layer, and $L_w$ and $L_2$ respectively represent the layer-wise loss and total loss among all of the layers.

Step 3: This step involves the sequential unsupervised training of multiple layers jointly, starting with layers 1-2, followed by layers 1-3, up to layers 1-8 based on the following objective function:

$$L_3 = \sum_l L_H \left( \left(y_1 - D_{dec_1}(G_{encl}(y_1))\right)^2 + \left(y_{l+1} - G_{encl}\left(D_{dec_1}\left(y_{l+1}\right)\right)\right)^2 \right), \tag{3}$$

**Algorithm 1** Response Field-Based Reconstruction Algorithm

1. Select a location at a specific layer for which we need to find the response field.
2. Find the receptive field for that specific response in the given layer.
3. Calculate the response field in that layer for all of the neurons in the input space.
4. Copy that response field and make the rest of the feature space zero.
5. Reconstruct the input space from the computed response field at that particular layer.
6. Crop the receptive field obtained in the second step.

where $D_{dec_1}(G_{encl}(y_1)$ and $G_{encl}\left(D_{dec_1}\left(y_{l+1}\right)\right)$ respectively represent the first layer output of the decoder when the $l_{th}$ layer input of the decoder is given as the $l_{th}$ layer output of the encoder and the $l_{th}$ layer output of the encoder when the first layer output of the decoder is given as the first layer input of the encoder, and $L_H$ and $L_3$ respectively represent a joint training loss at a particular sequence and the total training loss over the entire sequence.

The combined loss of FER-CNN is shown below.

$$L_T = L_1 + L_2 + L_3. \tag{4}$$

We use ModelNet dataset for training FER-CNN with resolutions of [32×32×32] and a minibatch size of 32. In consideration of the importance of learning rate, we carefully select the learning rate to be 0.0001 for training the feedforward weights and 0.00025 for training the feedback weights. Similarly, we also used the ADAM optimizer algorithm [22] for gradient-based optimization of the parameters of FER-CNN with beta=0.5.

For the implementation of FER-CNN, we used TensorFlow open-source software. By contrast, for training the network, we used intel core i7-5960X CPU with 64.0 GB RAM and an NVIDIA TITAN X graphics card.

## IV. RESPONSE FIELD-BASED RECONSTRUCTION

In order to assess the performance of the deep convolutional neural network, the activities of each neuron in the intermediate layers must first be known. FER-CNN is composed of convolutional layers, so reconstruction from a single feature is not straightforward. The receptive field in input space is not only the function of the given feature in that particular layer, but is also dependent on the neighboring features in that spatial location. This is due to the fact that overlapping convolution windows are dictated by the choice of layer configuration (i.e. window size and stride). In order to address this problem, we use a response field-based reconstruction algorithm (Algorithm. 1), which is presented in Fig. 2.

## V. COMPARATIVE ANALYSIS

In order to illustrate the effectiveness of the proposed approach, we conduct comparative analysis in this section based on the 2D dataset and ModelNet dataset.

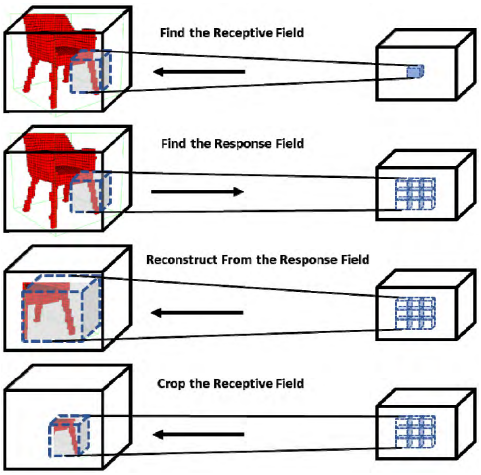


**FIGURE 2.** The first row shows the feature location in the feature space which is mapped back to the input space, the second row shows the response field from the corresponding input receptive field, the third row shows the reconstruction from the effective response field, and the last row represents the required receptive field.

### A. 2D ANALYSIS OF FER-CNN RECONSTRUCTION

The deep neural network learns the representation of the input distribution by partitioning the feature space in different cluster regions. The local clusters formed in the lower layers combine the local features with interclass or intraclass feature similarities, whereas the higher layers combine those local clusters into global clusters. The classification objective combines the cluster space in the final layer such that each sample in a particular class belongs to its representative cluster region.

In order to evaluate the capability of the deep neural network in terms of the representation of the internal structure of the data, we first conducted an analysis on the toy dataset for simplicity and later used the ModelNet dataset for further analysis. Regarding the toy dataset, we sampled two-dimensional data for different classes from a multinomial distribution. The distributions of different classes are represented by different colors in Figs. 3 and 4. The total number of

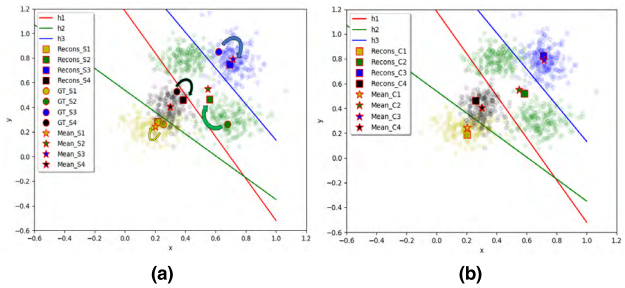


**FIGURE 3.** FER-FCN. (a) Sample-based Reconstruction (left). (b) Code-based Reconstruction (right). The colors show the distributions of four different classes, and the circles in (a) represent the input samples while the squares represent their corresponding reconstructed samples. Further, the stars represent the corresponding cluster centers and the lines represent the linear approximation of the projected input space partitioned by the hidden units in the first layer.

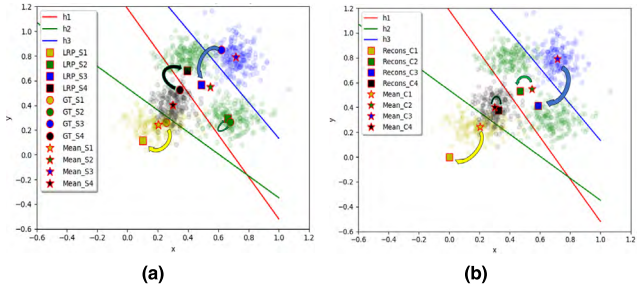


**FIGURE 4.** LRP (a) Sample-based Reconstruction (left), (b) code-based Reconstruction (right): The colors show the distributions of four different classes, and the circles in (a) represent the input samples while the squares represent their corresponding reconstructed samples. Further, the stars represent the corresponding cluster centers and the lines represent the linear approximation of the projected input space partitioned by the hidden units in the first layer.

training samples is 1000, where each class has 200 samples, except for the green color class with 400 samples. As this experiment is based on the two-dimensional toy dataset, we modified FER-CNN to a two-layer Feature Extraction and Reconstruction Fully Connected Network (FER-FCN). The configuration of the FER-FCN architecture is described in [2]–[4], where the input is two-dimensional input data transformed to three-dimensional feature space and in the second layer. The three-dimensional feature space is projected to four-dimensional feature space, which represents the final partitions of the feature space at the class level. Based on these experiments, we performed sample-based and code-based analysis to interpret the neurons in the network.

#### 1) SAMPLE-BASED ANALYSIS

For the sample-based analysis, we first trained FER-FCN on two-dimensional training samples. After training FER-FCN, we selected samples from each class and reconstructed them using feedback weights of FER-FCN. The selected input samples are represented by circles and their reconstructed results are represented by squares along with their corresponding class color in Fig. 3a. The reconstructed samples have a small offset from the input. Ideally, the representative sample of each distribution is located at the center. Thus, if the network is properly learned, the reconstructed samples should be more biased toward the cluster representative, which is the mean of that cluster. This is clear from our experiments, where the reconstructed samples are more biased toward the mean of each cluster or distribution, which are represented by stars with the corresponding color. During this analysis, we also show the input space partitioning by the neurons in the first layer. As the FER-FCN configuration uses three neurons in the first layer, the input is partitioned into seven possible regions by these three neurons. In the real scenario, the feature space is non-linear; however, for the ease of analysis, we approximate the space by the linear function as shown in equation 5. The linear approximation of the non-linear space shows that the different class regions are well separated, as depicted in Fig. 3a. For the purpose

of comparison, we also applied layer-wise relevance propagation [20] to the recognition part of FER-FCN, which is based on the normalized contributions accumulated from the top down to the input space. Fig. 4a shows the LRP-based analysis, where the resulting contributions are represented by squares and the corresponding input samples are represented by circles. The mean of each distribution is represented by a star with the corresponding color. The reconstructed samples, after the normalized contributions, show large offset from the mean of the corresponding distribution as compared to FER-FCN. Furthermore, the samples from the blue and black class regions are mapped to the green class region, as shown in Fig. 4a.

$$V \approx W^T H \tag{5}$$

where $V$ is the input space to be partitioned, $H$ are the features which partition the input space, and $W^T$ are the weight parameters between $V$ and $H$.

#### 2) CODE-BASED ANALYSIS

During the code-based analysis of FER-FCN, rather than selecting a sample from the input data, we pick a class representative of each distribution to feed to the network at the final layer. The codes are then reconstructed from the final layer using the feedback weights of FER-FCN, as shown in Fig. 3b, where the squares represent the reconstructed samples based on the class codes. The mean of each distribution is represented by a star with the corresponding color. The reconstructed results are more biased toward the mean of their corresponding distributions, showing the representation capability of FER-FCN in terms of the interpretation of neurons. The space partition is the same in both cases (sample-based and code-based reconstruction), as the input space is divided into different cluster regions by three neurons in the first layer in each case. We conducted a comparative analysis with LRP [20] in terms of the code-based reconstruction. The LRP approach is based on the input sample along with its corresponding class label. However, only class label information is available in this analysis, so we use feedforward weights and transpose them so as to obtain lower layer activations. The code-based analysis also shows a large offset as compared to that of the proposed approach, as shown in Fig. 4b.

**TABLE 1. The mean square error of the reconstructed sample from the corresponding ground truth target samples and from the mean of each cluster.**

| Approach | LRP [20] | **Ours** |
|---|---|---|
| MSE based on reconstructed samples | 0.0212 | **0.0098** |
| MSE from cluster center (based on sample) | 0.1468 | **0.0110** |
| MSE based on class code | 0.0580 | **0.013** |
| MSE from cluster center (based on class code) | 0.1334 | **0.0057** |

The interpretation of FER-FCN in terms of qualitative analysis is shown in Table 1, which shows the mean square errors between the reconstructed samples and their ground truth input samples as well as between the reconstructed samples and the means of their corresponding distributions. FER-FCN shows less error than LRP [20] in both cases.

### B. COMPARATIVE ANALYSIS OF FER-CNN AND LRP BASED ON MODELNET DATASET

In order to perform a comparative analysis for the interpretation of deep neural networks using FER-CNN and LRP [20] based on high dimensional 3D data, we selected the ModelNet dataset [23]. The neural network clusters the regions based on the interclass or intraclass feature similarities in the local layers, whereas the cluster regions become class-specific in the final layers due to the classification objective. We explore this capability of the neural network in terms of FER-CNN and LRP [20]. Exploring the cluster regions for each class sample in terms of representations helps identify the learning capability of the deep neural network. In order to explore such a cluster space, LRP [20] uses a normalized contribution-based approach; however, it does not explore the cluster regions well.

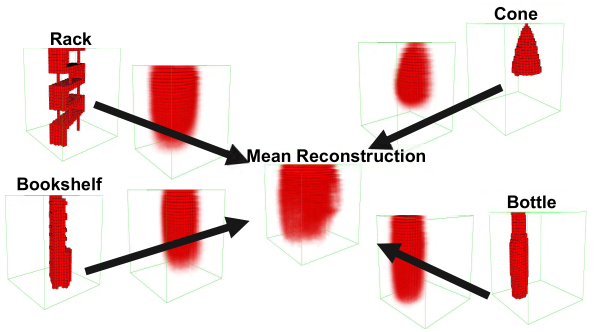


**FIGURE 5. LRP-based reconstruction: The reconstructed samples as a result of normalized contributions [20] are more biased toward the mean representation from different class samples.**

Experiments based on the toy dataset in Fig. 4a show that a single class cluster region is used as a representative for multi-class samples. The experimental results based on the ModelNet dataset [23] are shown in Fig. 5, where the different class samples such as Rack, Cone, Bookshelf, and Bottle are represented by LRP. The final representation of these samples shows more resemblance to the intraclass mean representation than explicit class-based sample representation. Fig. 6 further elaborates upon the representation of different class samples based on the normalized contribution, where the different class samples belonging to Chair, Car, Bench, Bathtub, Bed, and Bowl are represented. The representations of these samples are more biased toward the intraclass mean representation than the typical sample-based representation, implying that the cluster regions are not explored well for its corresponding cluster representation, hence making it blurrier.

By contrast, the FER-CNN learning rule is based on feature extraction and reconstruction, where the features are clustered based on their local as well as global similarities. The local layers maintain the typical structures of the

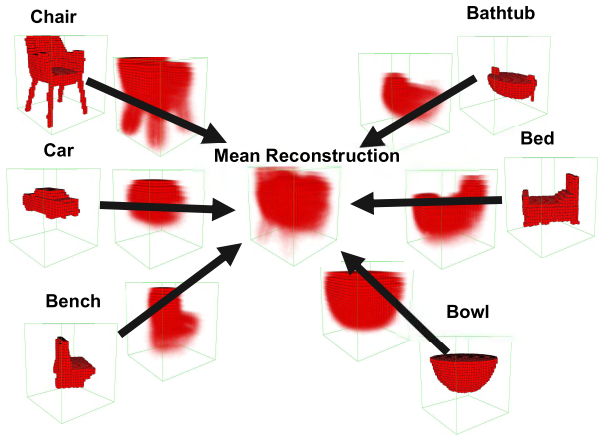


**FIGURE 6.** LRP-based reconstruction: The reconstructed samples as a result of normalized contributions [20] are more biased toward the mean representation from different class samples.

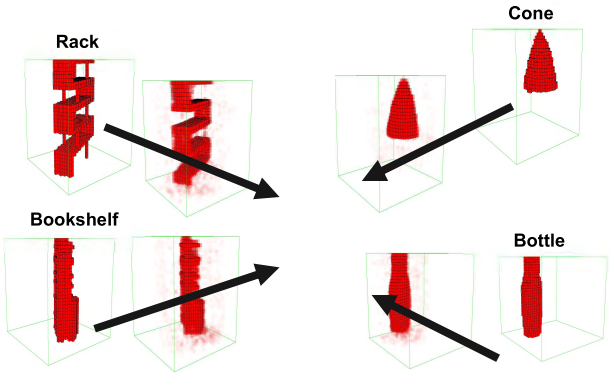


**FIGURE 7.** FER-CNN-based reconstruction: The reconstructed samples as a result of FER-CNN show more explicit representation.

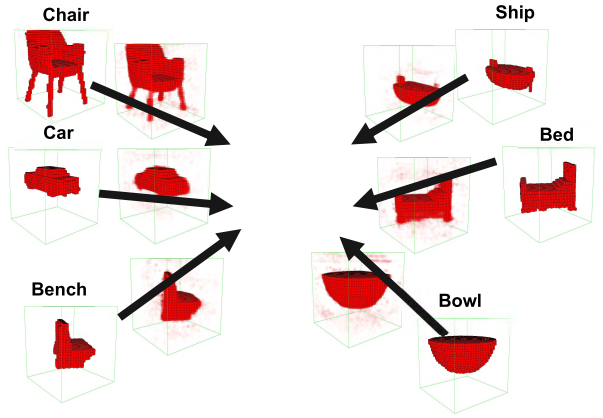


**FIGURE 8.** FER-CNN-based reconstruction: The reconstructed samples as a result of FER-CNN show more explicit representation.

samples, whereas the final layers converge the samples to their corresponding labels as their representatives. In contrast to LRP, the results based on FER-CNN using the ModelNet dataset are shown in Fig. 7. As discussed previously, the regions which are assigned to different classes must be explicit so as to ensure successful training of the deep neural network. Fig. 7 shows the explicit representation of these samples as compared to Fig. 5. Similarly, Fig. 8 also represents the explicit reconstruction of different class samples, which ensures the successful learning of the deep neural network.

**TABLE 2.** Mean square error and cosine distance between the reconstructed samples from full layer as well from a specific filter from the target samples.

| Approach | LRP [20] | **Ours** |
|---|---|---|
| MSE (Filter Based Reconstruction) | 581.72 | **119.35** |
| MSE (Full Layer Reconstruction) | 970.18 | **373.64** |
| Cosine distance (Filter Based Reconstruction) | 16.09 | **0.56** |
| Cosine distance (Full Layer Reconstruction) | 14.97 | **1.61** |

We also evaluated FER-CNN based on the quantitative results shown in Table 2. This analysis is based on testing samples that were randomly selected. We calculate the Euclidean distance and cosine distance between the reconstructed samples and the ground truth testing samples. The results show that FER-CNN has a lower per voxel error than LRP [20].

### C. ROBUSTNESS TO THE NOISE (STRUCTURAL REPRESENTATION)

The capability of the deep neural network to learn the representation of the internal structure of the data makes it useful under heavy corruption in the testing data. This internal representation of the data is made possible by the successful collaboration of neurons at different layers, which in turn provides robustness to the corruption in the input distribution.

However, to ensure the successful representation of the data in terms of the activities of neurons, FER-CNN adopts an effective learning mechanism which maintains the relationship between the receptive field and its corresponding feature representation both locally and globally. We evaluate and interpret the performance of FER-CNN by embedding random noise to the input samples. While the random noise distorts the structure of the data, the effect of the corrupted input on the learned representation of the network provides information about the capability of robustness of the network toward the variation in the input data. During this evaluation procedure, we analyzed reconstruction from the full layer as well as from the specific filter. Regarding reconstruction from the full layer, we pass the corrupted input sample and obtain the full code of a specific layer, and then use feedback weights to reconstruct the full object with the code obtained from the corrupted input. Similarly, regarding reconstruction from a filter, we select a specific feature in the feature space, then use the response field-based reconstruction algorithm to find the response field of that feature based on the receptive field, and then reconstruct that response field as a part of the corrupted input object. Fig. 9 shows the comparative results from the receptive field-based cropping [19] and FER-CNN-based reconstruction. The first row of Fig. 9 shows the clean testing samples from the ModelNet dataset [23] which are corrupted by uniform random noise. The resulting corrupted samples

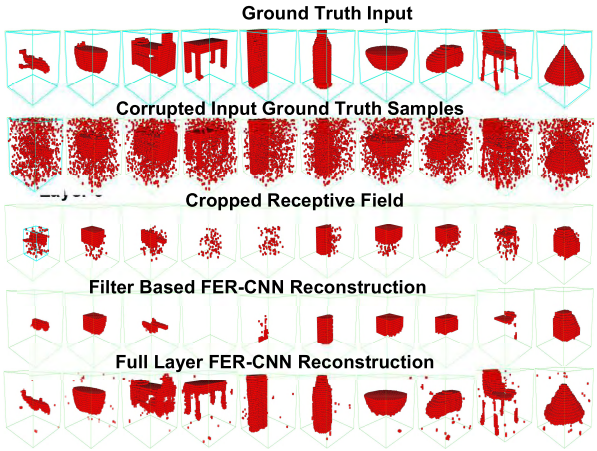


**FIGURE 9.** The first row shows the input testing samples from ModelNet dataset, the second row shows the corrupted samples as input to FER-CNN, the third row shows receptive field cropping [19] from a specific location in the feature space of the second layer, and the fourth and fifth rows represent FER-CNN reconstruction from that specific location in the feature space of second layer and full layer reconstruction, respectively.

are shown in the second row of Fig. 9. The corrupted samples are then propagated through the feedforward network, and a specific layer code is obtained. Regarding full layer reconstruction, we used trained feedback weights of FER-CNN and reconstructed the corrupted samples as shown in the fifth row. Similarly, the fourth row shows the reconstruction results from a specific filter from the proposed network, while the third row shows receptive field-based cropping [19]. The proposed approach shows prominent results as compared to the receptive field-based cropping [19].

Notably, FER-CNN takes advantage of some of the corrupted voxels. These corrupted voxels in the specified samples may be parts of other samples in the same class, which adds to the internal representation of the input distribution. For this reason, FER-CNN produces more prominent results, showing its capability to enhance the internal representation of the data. By contrast, the receptive field-based cropping [19] is specific to the current input samples and does not explore the capability of the network in terms of the representation of the internal structure of the data.

## VI. IMPLICATIONS OF FEEDBACK WEIGHTS AS A MEAN OF INTERPRETATION

The proposed feedback network with its weights trained based on the input dataset plays an important role in interpreting what is learned in the feedforward network to which it is attached. The reconstruction of an input sample through the feedback network connotes the way in which the feedforward network clusters a hierarchy of features for classification. As such, the reconstruction ignores the noise deformation of input samples but generates more typical representatives corresponding to the classification result. FER-CNN provides an effective means for achieving the desired goal of interpreting the cluster formed in the feedforward path of the classification network, where the feedback layers preserve information about the individual samples locally as

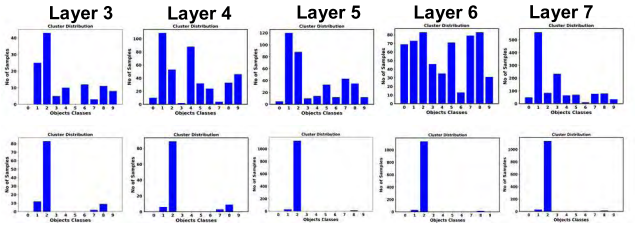


**FIGURE 10.** The implication of feedback weights as a mean of the interpretation of deep neural network using clustering-based analysis. The top row represents layer-wise representative clusters in the feedforward path while the bottom row represents the layer-wise representative clusters in the feedback path.

well as their mean representation globally. The clustering-based interpretation explores the effect of feedback weights on the interpretation of deep neural networks, where we first cluster the feature space in each layer along the filter dimensions using the k-means clustering algorithm. As we used the ModelNet10 dataset, which has 10 classes, for this analysis, we set k=10 in the last fully connected layer. For the rest of the layers, we set k= [60,50,40,30,30,20,20] as appropriate. Upon completion of the clustering of the feature space, we select the testing samples and pass them through the network, then find the nearest cluster in each layer at each location, as shown in Fig. 10. The representative clusters at each layer are implicit in the feedforward path, as shown in the top row of Fig. 10. By contrast, the bottom row represents the representative clusters in the feedback path, which are more biased to the typical representation of the input sample, and contains information about their representative classes which we explain further in the following section.

### A. CLUSTERING BASED INTERPRETATION AND FER-CNN RECONSTRUCTION

One of the most important and desirable aspects of this analysis is to interpret and evaluate what is learned by the deep neural network. One way to evaluate the performance of the deep neural network is based on classification accuracy; however, classification accuracy-based analysis is abstract and does not provide comprehensive information regarding the interpretation of the deep neural network. For example, if an object is misclassified, the classification-based analysis merely provides the score of its classification and does not provide any reasons for the failure of an object classification. In order to explore the interpretation of the deep neural network and analyze what is learned in the recognition part of the network, the trained feedback connections play a key role. We use clustering-based analysis using FER-CNN which provides an effective means for the interpretation of a deep neural network by reconstructing the output of the recognition framework, then evaluate its recognition capability both qualitatively and quantitatively.

In order to perform the clustering-based analysis for the interpretation of a deep neural network while using FER-CNN, we cluster the feature space of all of the layers along the filter dimensions using feedforward and feedback weights, then perform interpretation of the deep neural

network using three different approaches. In the first approach, we cluster the feature space of all of the layers along the filter dimensions using the feedforward parameters of the network, then pass the testing sample through the network. In this step, we select the cluster centroid at each spatial location nearest to the corresponding location in the input testing sample and then pick the selected clusters as representatives of that testing sample. The cluster distribution at the local layers provides us with information about the features which are more biased to the intraclass feature similarities than the specific object class. The cluster distribution becomes more representative of that class as we go up the network, and the last layer cluster represents the class of that object. During the second approach, we cluster the feature space of all of the layers along the filter dimensions using the feedback parameters of the network.

After clustering the feature space in the feedback path, we pass the input testing sample through the network, then obtain its code at each layer using the feedback parameters. Then, using the KNN algorithm, the nearest representative cluster centroids at each layer and each spatial location to the corresponding spatial location in the input testing sample are obtained.

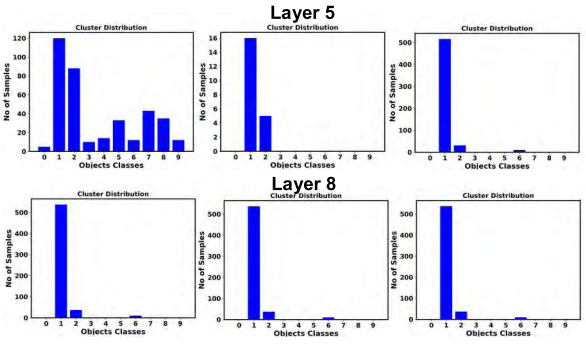


**FIGURE 11.** Clustering-based analysis of the correctly classified input testing samples from layers 5 and 8. The first column represents the clustering in the feedforward direction along with the testing sample features computed from the feedforward parameters. The second column represents the clustering in the feedback direction, whereas the testing sample features are computed from the feedforward parameters. In the third column, both the clustering as well as the testing sample features are computed using feedback parameters. Each row represents layer-wise analysis.

The clusters formed in the feedback path carry information about the misclassified samples to the input space as a mean of qualitative and quantitative analysis for the interpretation of the network. In the third approach, we obtain the code of the testing samples using the feedforward weights, whereas the clusters are obtained using the feedback weights, and then the candidate cluster centroids are selected following the same procedure as that of the first and second approaches. Figs. 11 to 14 shows the interpretation of the deep neural network based on the approaches discussed above. First, we analyze the correctly classified input testing sample using the cluster-based interpretation shown in Fig. 11. The first column represents the clustering in the feedforward direction along with the testing sample features computed from the

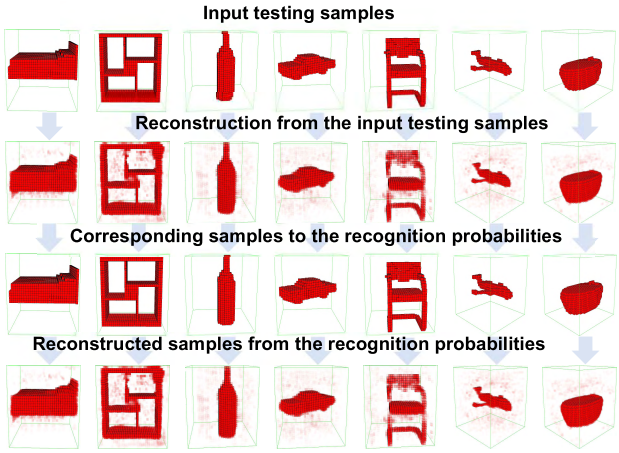


**FIGURE 12.** Clustering-based analysis of the correctly classified input testing samples by reconstructing them from the local layers as well as from the correctly classified class probabilities. The first row shows the input testing samples and the corresponding columns in the second row represent the generated samples from the local layers, while the third and fourth rows show the nearest ground truth and their corresponding reconstructed samples to the cluster centroid based on the recognition probabilities, respectively.

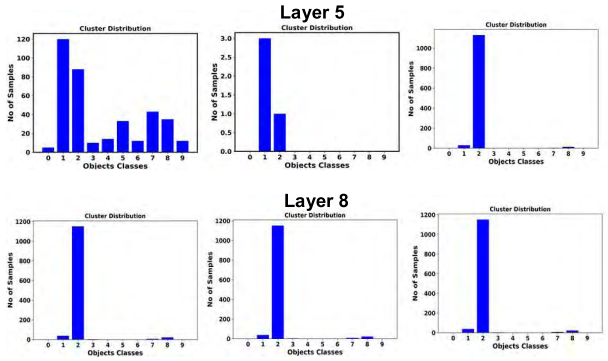


**FIGURE 13.** Clustering-based analysis of the wrongly classified input testing samples from layers 5 and 8. The first column represents the clustering in the feedforward direction along with the testing sample features computed from the feedforward parameters. The second column represents the clustering in the feedback direction, whereas the testing sample features are computed from the feedforward parameters. In the third column, both the clustering and the testing sample features are computed using feedback parameters. Each row represents layer-wise analysis.

feedforward parameters. The second column represents the clustering in the feedback direction, where the testing sample features are computed from the feedforward parameters. The third column shows clustering in the feedback direction with the testing sample features computed from the feedback parameters. Each row in Fig. 11 represents a layer-wise analysis. This analysis indicates that the selected clusters are specific to the corresponding class based on the three approaches discussed above. In the second analysis, we used the wrongly classified input testing sample, and the candidate clusters obtained from the feedback weights in the designated layers represent the wrong class of the objects as shown in the third column, whereas the feedforward candidate clusters show the correct class corresponding to the input testing sample, as shown in the first and second columns of Fig. 13. The qualitative analysis for the interpretation of the

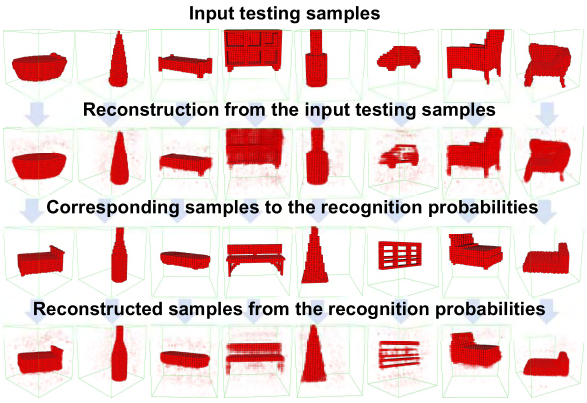


**FIGURE 14.** Clustering-based analysis of the wrongly classified input testing samples by reconstructing them from the local layers as well as from the correctly classified class probabilities. The first row shows the input testing samples and the corresponding columns in the second row represent the generated samples from the local layers, while the third and fourth row show the nearest ground truth and their corresponding reconstructed samples to the cluster centroid based on the recognition probabilities, respectively.

deep neural network using feedback weights is shown in Figs. 12 and 14. In Fig. 12, we first analyze the samples which have been correctly classified by the recognition network. As the first step of the analysis, we provide these input testing samples to the network then reconstruct these from the local layers using the feedback weights. As discussed previously, the local layers are more biased to the feature similarities than the classification objective; therefore, it reconstructs the samples having similarities with the input testing samples. The reconstructed input testing samples are shown in the corresponding columns in the second row in Fig. 12. The recognition network generates the class probabilities based on the input testing samples. These class probabilities are fed to the feedback network, which generates their features at each layer. The features at each layer then pick the nearest cluster centroids and reconstruct them. The nearest sample to the cluster centroid and its corresponding reconstructed results are shown in the third and fourth rows of Fig. 12, respectively, and the results indicate that the feature similarity-based reconstructions from the local layers as well as from the generated class probability match each other, hence, proving the correct classification accuracy. For example, the bed, bookshelf, bottle, car, chair, airplane, and bathtub are all correctly classified by the recognition network and their reconstructed results from the correct class codes show similarities with the input testing samples, as shown in Fig. 12. In Fig. 14, we analyze the samples which have been wrongly classified by the recognition network. During this analysis, the network reconstructs these input testing samples from the local layers using the feedback weights as shown in the corresponding columns in the second row of Fig. 14, which shows similarities to the input testing samples regardless of their classification. Based on these input testing samples, the recognition network generates the class probabilities. These class probabilities are fed to the feedback network, which generates their features at each layer. The features at each layer then pick the nearest cluster centroids and reconstruct them. The nearest sample to the cluster centroid and its corresponding reconstructed results are shown in the third and fourth rows of Fig. 14, respectively. The results show that the samples reconstructed from the local layers match the input testing samples regardless of their recognition probability, whereas the samples generated from the cluster centroids using the class probabilities as input to the feedback network do not match the input testing samples, and it selects the clusters belonging to the recognition probabilities generated by the network. This analysis further elaborates that the samples which are wrongly classified show structural similarities to the input testing samples. For example, the bathtub, cone, table, bookshelf, bottle, car, and chair are wrongly classified as a bed, bottle, bathtub, bench, cone, bookshelf, and bed, respectively. The misclassified samples show structural similarities with the input testing sample. Such analysis provides insight into the classification of the recognition network by qualitatively interpreting its recognition.

**TABLE 3.** Mean square error between the input sample and reconstructed samples.

| | |
|---|---|
| Error between input representative and reconstructed output | 19.185 |
| Error between the misclassified sample and reconstructed output | 22.610172 |
| Error between reconstructed input representative sample and reconstructed misclassified | 26.466608 |
| Error between the input representative and reconstructed misclassified sample | 35.837 |

We also performed quantitative analysis in terms of the mean square error of the misclassified input testing sample and the generated sample from the wrong recognition probabilities presented in Table 3. This analysis shows that the error between the input testing sample and the reconstructed sample from the misclassified class probability is almost double the error between the input testing sample and its reconstruction from the local layers. This error difference provides information about the misclassification of the input testing samples.

**TABLE 4.** Comparison based on computational complexity.

| Approaches | Computational complexity | Evaluation based on the class probability | Image generation from any layer | Image generation from any filter |
|---|---|---|---|---|
| Iterative approach [15] | 6s per image (approx.) | × | √ | × |
| Iterative approach [16] | 6s per image (approx.) | × | √ | × |
| LRP [20] | 3ms per image (approx.) | × | × | × |
| Zeiler [6] | 3ms per image (approx.) | × | √ | × |
| **Ours** | 3ms per image (approx.) | √ | √ | √ |

Table 4 shows the comparative analysis of the proposed approach with the iterative approaches [15], [16] and

LRP [20] in terms of computational complexity, their abilities to evaluate the misclassified samples, and image generation from any layer and any filter. The computational complexities of the proposed approach and LRP [20] are the same at the testing time; however, LRP [20] cannot evaluate the misclassified samples from the class probabilities. By contrast, the proposed approach effectively evaluates the samples which are misclassified based on their class probabilities. On the other hand, the iterative approaches [15], [16] are computationally expensive and cannot evaluate the misclassified samples. The proposed approach can effectively generate images from any layer and any filter, whereas LRP is based on an end-to-end evaluation and reconstructs the images from the top in terms of the normalized contributions. By contrast, the iterative approaches generate images from any layer, yet they do not consider the filter-based reconstruction.

## VII. CLASSIFICATION BASED ANALYSIS

We also evaluated FER-CNN for the input testing samples classification as a means of interpreting the recognition network. For this analysis, we used the 3D ModelNet dataset. This dataset is based on Modelnet10 and ModelNet40. A comparative analysis of FER-CNN with the current state of the art networks based on ModelNet10 and ModelNet40 is presented in Table 5. This analysis shows that the proposed approach outperformed state-of-the-art approaches in terms of ModelNet10 and ModelNet40 dataset classification. Furthermore, these approaches were unable to evaluate the classified testing samples, whereas the proposed approach effectively evaluates the wrongly classified testing samples and provides meaningful information about the misclassified testing samples, as was already discussed in Section VI.

**TABLE 5. Classification accuracy based on modelnet10 and modelnet40.**

| Algorithm | ModelNet10 | ModelNet40 |
|---|---|---|
| Minto et al. [24] | 93.6% | 89.3% |
| LonchaNet [25] | 94.37 | - |
| Achlioptas et al. [26] | 95.4% | 84.5% |
| Soltani et al. [27] | - | 82.10% |
| Arvind et al. [28] | - | 86.50% |
| Zanuttigh and Minto [29] | 91.5% | 87.8% |
| ECC [30] | 90.0% | 83.2% |
| LightNet [31] | 93.94% | 88.93% |
| **FER-CNN** | **97%** | **89.5%** |

## VIII. FEATURE INTERPOLATION

A deep neural network capable of self-recognizing and visualizing what is learned in a top-down manner without a given input has significant meaning, as it resembles imagination in the human thinking process. For example, a deep learning network that has already been trained is able to reconstruct the virtual input by itself that corresponds to a code assigned to the filter associated with a particular node of any layer. In this case, the virtual input thus constructed does not necessarily belong to the dataset for training, but may represent an interpolation of multiple data in the dataset. Such a capability of the deep neural network is made possible by the successful representation of the internal structure of the input data in terms of the collaboration of neurons. There are a few different ways to explore the representation capability of the deep neural network. One way is to simply put the samples at the input and then reconstruct them, but this approach cannot provide insight about the capability of the network toward the representation of the input data in terms of the neuron collaboration. For example, what is the behavior of the network in the case of a transition from one sample to the other? In order to address this problem, we proposed a feature interpolation algorithm (Algorithm. 2) which effectively verifies the behavior of the network in terms of the neuron collaboration towards the variations in the feature space of the network. For this interpolation, we transform the input samples into their feature representations at a specific layer in the network. We then sample the feature space at that particular location along the filter dimensions from the uniform distribution. Next, we select the range of the uniform distribution between the minimum and maximum values as that of the selected node in the filter dimensions. In order to select the k-nearest feature representation of the samples to the code sampled from uniform distributions, we adopted the following two approaches.

### A. CENTROID-BASED APPROACH

In this approach, we select one particular location in the feature space and apply the K-mean clustering algorithm along the filter dimensions at that specific location. Following clustering in the filter dimensions, we select the k-nearest centroids of the clusters to the sampled code using the K-Nearest Neighbor (KNN) algorithm. Once the candidate k-nearest centroids are selected, we apply the KNN algorithm with the k equal to 1 in order to find the k-nearest samples feature representations to each centroid. The required response field of the contributing neurons is calculated by the response field-based reconstruction algorithm and filled by the selected feature representation of those samples.

### B. K-NEAREST SAMPLES-BASED APPROACH

In this approach, we select one particular location in the feature space and apply the KNN algorithm to find k-nearest samples features to the sampled code. The response field of each node is computed and filled by the selected k-nearest features.

Once the response fields for all of the selected centroids are defined, we calculate the offset distance $d_k$ between the sampled code and the associated centroids or k-nearest feature codes. The last step involves interpolating among the response field of the selected feature codes, where we define $\alpha_k$ as a transition parameter from one response field to the next, and the transitioned response fields are reconstructed as the interpolated results.

The feature interpolation algorithm is summarized as follows.

**Algorithm 2** Feature Interpolation

1) Compute the codes for node 'i' at layer 'L' based on the input testing samples

$$n_i = \Omega(x_i), \tag{6}$$

where $n_i$ represent the feature codes of neuron i, $\Omega(.)$ represents the mapping function from input to a specific layer and $x_i$ shows the input testing samples.

2) Apply the clustering algorithm in the filter dimensions to that particular node,
3) Select the centroid of the clusters as

$$C_k = \frac{1}{|S_k|} \sum_{n_i \epsilon S_k} n_i, \tag{7}$$

$C_k$ shows the centroid of the clusters and $S_k$ represents the total number of features codes in the cluster $k$.

4) Sample the code from a uniform distribution,

$$p(\xi) = \begin{cases} 0, & \xi < \min(n_i) \\ \frac{1}{\max(n_i) - \min(n_i)} & \min(n_i) \leq \xi \leq \max(n_i) \\ 0, & \xi \geq \max(n_i), \end{cases} \tag{8}$$

where $p(\xi)$ represents the feature code sampled from the uniform distribution.

5) Apply KNN algorithm to find the k-nearest samples codes or k-nearest centroids of the clusters in the filter space to $p(\xi)$,
6) Find the corresponding response field using response field-based reconstruction algorithm,
7) Compute the distance $d_k$ from each sample code to the $p(\xi)$,
8) Calculate $\alpha_1, \alpha_2, \ldots, \alpha_k$, where,

$$\alpha_k = \frac{d_k}{\sum_{n=1}^{k} d_n}, \tag{9}$$

$$d_k = d_k^{'} e^{-0.006t}, \tag{10}$$

where $\alpha_k$ is the normalization parameter for mixing the response fields, $d_k^{'}$ is the offset distance between an arbitrary selected code in a feature space and the code of a sample chosen from a nearby cluster $k$, and $d_k$ is used to control the influence of individual clusters on interpolation by adjusting $t$.

The final response field is obtained by

$$R = \alpha_1 R_1 + \alpha_2 R_2 + \ldots + \alpha_K R_K. \tag{11}$$

$R$ is the final response field in the feature space which is used to reconstruct the input samples.

#### 1) RESULTS AND DISCUSSION OF FEATURE INTERPOLATION

interpolation algorithm for investigating the activities of neurons in terms of their collaboration, we first trained FER-CNN on a large-scale ModelNet dataset. The ModelNet dataset consists of ModelNet10, having 10 classes of around 48000 3D CAD models, of which 38400 are used for training and 9600 are used for testing, and the ModelNet40, with 40 classes of 151,128 3D CAD models, where 121512 are used for training and the remaining 29616 are selected as testing samples. After training, we used the interpolation algorithm based on the two approaches discussed above.

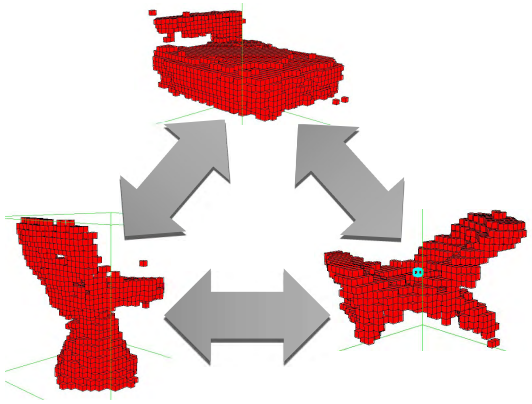

**FIGURE 15.** Centroid-based interpolation: Three clusters centroids are selected and the samples nearest to each centroid are a bed, gaming chair, and easy chair. This shows interpolation from the bed to the gaming chair and from the gaming chair to the easy chair as well as the other way around.

The first approach involves finding the k-nearest centroids and its corresponding feature codes to the feature code sampled from the uniform distribution. The selected feature codes are then used to fill the response field of each centroid. In this experiment, the selected response fields belong to the Gaming chair, Bed, and Easy chair. The distance $d_n$ between the centroids and the generated feature code from the uniform distribution is then computed. The mixing parameter $\alpha_k$ is also computed based on eq (9). The interpolation results from the centroid-based approach are shown in Fig. 15, where the transformations from the bed to the gaming chair and from the gaming chair to the bed are shown. Fig. 16. shows a detailed description of our interpolation algorithm, where the first row shows the transformation from the bed to the gaming chair. The in-between samples show the intermediate representation of the transformation between these two samples. Similarly, the first sample of the second row shows the last interpolated sample from the bed, which is a gaming chair, and this sample is transformed to an easy chair. The easy chair is then transformed back to the bed so as to complete the interpolation cycle.

The second approach involve selecting the k-nearest feature codes which are obtained from the testing samples to the feature code sampled from the uniform distribution. The selected feature codes of the k-nearest samples in our case belong to the lounge chair, bed, and carver chair. These selected feature codes are then used to fill the response fields of a specific feature at a particular layer. The distance $d_n$ between the feature codes of the k-nearest samples and the


**FIGURE 16.** Centroid-based interpolation: First, the centroids of the three clusters to be interpolated are selected. Then, the nearest sample code from an individual centroid is selected, as exemplified in the figure with a bed, gaming chair, and easy chair. The transformation from the bed to the gaming chair is shown in the first row, where the first sample shows the starting point of interpolation and the last sample shows the interpolated sample by changing its feature code in the response field. The in-between samples show the intermediate representation of the transformation from the bed to the gaming chair. The first sample of the second row shows the last interpolated sample from the bed, which is the gaming chair, and this is transformed to the easy chair, whereas the first sample of the last row is the result of interpolation from the gaming chair to the easy chair, where it is transformed back to the bed.

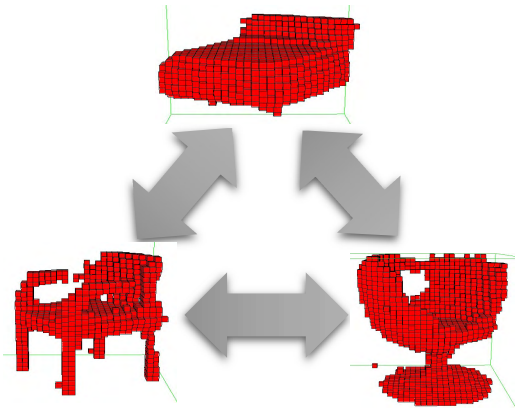

**FIGURE 17.** k-Nearest samples-based interpolation: Three nearest samples codes to the uniformly generated code are selected which belong to the bed, lounge chair, and carver chair. This shows interpolation from the bed to the lounge chair and from the lounge chair to the carver chair, as well as the other way around.

generated feature code from the uniform distribution is computed along with the mixing parameter $\alpha_k$ using eq (9). The interpolation results from the k-nearest samples-based approach are shown in Fig. 17, where the transformations from the bed to a lounge chair, and from the lounge chair to a carver chair and then back to the bed, are shown. A detailed description of the k-nearest sample-based approach is shown in Fig. 18, where the first row shows the transformation from the bed to a lounge chair. The in-between samples show the intermediate representations of the transformation between these two samples. Similarly, the first sample of the second row shows the last interpolated sample from the bed, which is a lounge chair, and this sample is transformed to a carver chair. The carver chair is then transformed back to the bed.

**FIGURE 18.** k-Nearest samples-based interpolation: Three nearest samples codes to the uniformly generated code are selected, which belong to the bed, lounge chair, and carver chair. The first row shows interpolation from the bed to the lounge chair, where the first sample shows the start of the interpolation and the last sample shows the interpolated sample by changing its feature code in the response field. The in-between samples show the intermediate representation of the transformation from the bed to the lounge chair. The first sample of the second row shows the last interpolated sample from the bed, which is the lounge chair, and this is transformed to a carver chair, whereas the first sample of the last row is the result of interpolation from the lounge chair to the carver chair, and this is transformed back to the bed.

### 2) HYPER-PARAMETERS SETTING

The hyper-parameters play a crucial role in the learning of a deep neural network. One of the ways to find a suitable learning rate for training a deep neural network is to start with a very low learning rate and then linearly or exponentially increase it in each iteration. Once the loss begins to increase drastically, stop the training and then select the optimal learning rate. The proposed network is based on two independent steps, where the selection of learning rate is more challenging and computationally expensive based on the procedure discussed above. In the proposed network, the feedforward CNN loss is convex, so a comparatively lower learning rate is used. On the other hand, the second step is based on training feedback CNN, where the space is highly non-linear, so we used a comparatively higher learning rate in order to avoid local minima.

## IX. CONCLUSIONS

We investigate the implications of feedback weights in deep neural network as a means of interpreting the activities of neurons at different layers toward the recognition of the input testing samples by proposing a clustering-based interpretation algorithm. The proposed algorithm effectively reconstructs the input testing samples from the class probabilities by selecting the centroids of their representative clusters in the feedback path. The reconstructed samples are evaluated both qualitatively and quantitatively in terms of their recognition, and the results provide meaningful information about the cause of failure. The cluster-based analysis in the feedback path of the network is made possible by a modified feature extraction and reconstruction neural network with the capability of learning hierarchy of the features. The feature extraction and reconstruction network provides typical sample reconstruction from the local layer and its corresponding mean representation from the global layers. Furthermore, we propose the use of a feature interpolation

algorithm to analyze the deep neural network in terms of the neuron collaboration. The feature interpolation algorithm provides meaningful visual information about the behavior of the neurons during the transformation among different objects in the feature space. For future work, we will extend our work of interpretation as a means of connecting the semantic knowledge while extracting meaningful features using a deep neural network.

## Acknowledgment

Sukhan Lee proposed the methodology of clustering-based interpretation of FER-CNN, while Naeem Ul Islam implemented the concept and carried out experimentation.

## REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[2] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[5] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, Feb. 2015, pp. 2048–2057.
[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[7] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 1746–1754.
[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with. Deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
[9] V. Badrinarayanan, A. Kendall, R. Cipolla. (2015). "SegNet: A deep convolutional encoder-decoder architecture for image segmentation." [Online]. Available: https://arxiv.org/abs/1511.00561
[10] C. Ledig *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4681–4690.
[11] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2414–2423.
[12] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 694–711.
[13] Q. Zhang and S. C. Zhu, "Visual interpretability for deep learning: A survey," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 1, pp. 27–39, Jan. 2018.
[14] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0130140.
[15] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5188–5196.
[16] K. Simonyan, A. Vedaldi, and A. Zisserman. (Dec. 2013). "Deep inside convolutional networks: Visualising image classification models and saliency maps." [Online]. Available: https://arxiv.org/abs/1312.6034
[17] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806
[18] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 4829–4837.
[19] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. (Dec. 2014). "Object detectors emerge in deep scene CNNs." [Online]. Available: https://arxiv.org/abs/1412.6856
[20] A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," in *Proc. Int. Conf. Artif. Neural Netw.*, 2016, pp. 63–71.
[21] S. Lee, A. M. Naguib, and N. U. Islam, "3D deep object recognition and semantic understanding for visually-guided robotic service," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 922–928.
[22] D. P. Kingma and J. Ba. (Dec. 2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980
[23] Z. Wu *et al.*, "3D shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1912–1920.
[24] L. Minto, P. Zanuttigh, and G. Pagnutti, "Deep learning for 3D shape classification based on volumetric density and surface approximation clues," in *Proc.* 5 *VISAPP*, 2018, pp. 317–324.
[25] F. Gomez-Donoso, A. Garcia-Garcia, J. Garcia-Rodriguez, S. Orts-Escolano, and M. Cazorla, "LonchaNet: A sliced-based cnn architecture for real-time 3D object recognition," in *Proc. Int. Joint Conf. Neural Netw.*, May 2017, pp. 412–418.
[26] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. (2018). "Learning representations and generative models for 3D point clouds." [Online]. Available: https://arxiv.org/abs/1707.02392
[27] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum, "Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 1511–1519.
[28] V. Arvind, A. Costa, M. Badgeley, S. Cho, and E. Oermann. (2017). "Wide and deep volumetric residual networks for volumetric image classification." [Online]. Available: https://arxiv.org/abs/1710.01217
[29] P. Zanuttigh and L. Minto, "Deep learning for 3D shape classification from multiple depth maps," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2017, pp. 3615–3619.
[30] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3693–3702.
[31] S. Zhi, Y. Liu, X. Li, and Y. Guo, "Toward real-time 3D object recognition: A lightweight volumetric CNN framework using multitask learning," *Comput. Graph.*, vol. 71, pp. 199–207, Apr. 2018.

**NAEEM UL ISLAM** received the B.S. degree in electrical and electronics engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2008. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea.

His research interests include artificial intelligence, machine learning, computer vision, and deep learning.

**SUKHAN LEE** received the B.S. and M.S. degrees in electrical engineering from Seoul National University, South Korea, in 1972 and 1974, respectively, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1982.

From 1983 to 1997, he was with the Departments of Electrical Engineering and of Computer Science, University of Southern California. From 1990 to 1997, he was with the Jet Propulsion Laboratory, California Institute of Technology, as a Senior Member of the Technical Staff. From 1998 to 2003, he was the Executive Vice President, and also the Chief Research Officer with the Samsung Advanced Institute of Technology. He has been working as a Professor of information and communication engineering and WCU Professor of interaction science with Sungkyunkwan University, since 2003. He was designated the Dean of the Graduate School of Sungkyunkwan University, in 2011.He is also serving as the Director of the Intelligent Systems Research Institute. His research interests are in the areas of cognitive robotics, intelligent systems, and micro/nano electro-mechanical systems.

Dr. Lee is currently a Fellow of the Korean National Academy of Science and Technology.