Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07313-2

ORIGINAL ARTICLE

Uncertainty handling in convolutional neural networks

Elyas Rashno[1] · Ahmad Akbari[1] · Babak Nasersharif[2]

Received: 9 May 2019 / Accepted: 18 April 2022

Abstract
The performance of convolutional neural networks is degraded by noisy data, especially in the test phase. To address this challenge, a new convolutional neural network structure with data indeterminacy handling in the neutrosophic (NS) domain, named as Neutrosophic Convolutional Neural Networks, is proposed for image classification. For this task, images are firstly mapped from the pixel domain to three sets true (T), indeterminacy (I) and false (F) in NS domain by the proposed method. Then, NCNN with two parallel paths, one with the input of T and another with I, is constructed followed by an appropriate combination of paths to generate the final output. Here, two paths are trained simultaneously, and neural network weights are updated using back propagation algorithm. The effectiveness of NCNN to handle noisy data is analyzed mathematically in terms of the weights update rule. Proposed two paths NS idea is applied to two basic models: CNN and VGG-Net to construct NCNN and NVGG-Net, respectively. The proposed method has been evaluated on MNIST, CIFAR-10 and CIFAR-100 datasets contaminated with 20 levels of Gaussian noise. Results show that two-path NCNN outperforms CNN by 5.11% and 2.21% in 5 pairs (training, test) with different levels of noise on MNIST and CIFAR-10 datasets, respectively. Finally, NVGG-Net increases the accuracy by 3.09% and 2.57% compared to VGG-Net on CIFAR-10 and CIFAR-100 datasets, respectively.

Keywords Convolutional neural network · Neutrosophic theory · Data indeterminacy · Image classification

1 Introduction

Although crisp sets with 0 and 1 membership degrees were developed into fuzzy sets with continuous membership degrees, the uncertainty of each data point is not considered and described in the classical fuzzy set. In each application, uncertainty is a concept that indicates how membership degree is described for each set. The traditional fuzzy set describes the membership degree with a real number $\mu_A(x) \in [0, 1]$ [1]. In this situation, the uncertainty about $\mu_A(x)$ itself is not considered and handled [2]. In systems such as expert systems, information fusion, and belief systems, the truth-membership supported by the evidence can not solve and model the related problems, alone. So, the classical fuzzy set cannot solve these problems as well [2]. To achieve a complete model, falsity and indeterminacy memberships should be considered. Neutrosophic (NS) set is an extension of the fuzzy set that attempts to solve this problem by considering the truth, indeterminacy, and falsity memberships. Where the truth and indeterminacy memberships can be considered independently [3].

1.1 Literature review

Neutrosophy theory, proposed by Smarandache in 1995 [4], is a branch of philosophy which studies the nature and scope of the neutralities and their interactions with different ideational spectra which is the basis of neutrosophic logic and set [5, 6]. This theory has been applied for image processing first by Guo et. al [3], and then, it has been successfully used for other image processing domains

✉ Ahmad Akbari
akbari@iust.ac.ir

Elyas Rashno
e_rashno95@comp.iust.ac.ir

Babak Nasersharif
bnasersharif@kntu.ac.ir

[1] Department of Computer Engineering, Iran University of Science and Technology, Narmak, Tehran 1684613114, Iran

[2] Department of Computer Engineering, K.N. Toosi University of Technology, Tehran, Iran

Published online: 18 June 2022

including image segmentation [7–10], image thresholding [11], image edge detection [12], retinal image analysis [13–21], liver image analysis [22, 23], breast ultrasound image analysis [24], data classification [25], uncertainty handling [26, 27], data clustering [28–30], content-based image retrieval [31, 32] and skeletal muscle analysis [33, 34].

Today, deep learning methods have been widely used for feature extraction, feature compression, and classification. Over the past few years, convolutional neural networks (CNNs), introduced as one of deep learning approaches, have significantly improved classification accuracy and reduced processing costs [35, 36]. CNNs use variations of multilayer perceptron such as shift invariant or space invariant artificial neural networks (SIANN) to analyze visual imagery automatically [37]. Different CNN architectures have been proposed as reference models for image processing and machine vision tasks. An eight-layer CNN named as Alex-Net has been proposed in which CNN features were visualized [38]. In addition, two proposed CNN models including a very deep convolutional network (VGG-Net) [39] and GoogLe-Net [40] have been proposed by Oxford University's Visual Geometry Group and Google Inc., respectively. CNNs have been applied in many application such as fluid segmentation [41–43], coronavirus disease (covid-19) detection [44], and vegetation remote sensing [45].

Any prediction or classification by CNN may be associated with a degree of uncertainty. There are two common types of uncertainty which stem from the uncertainty in the training dataset and uncertainty in the model structure [46–49]. If uncertainty is not handled, it may lead to disastrous consequences. For example, in May 2016, the perception system confused the white side of a trailer and the bright sky which caused the first fatality from an assisted driving system [50]. Also, an image classification system for racial discrimination erroneously identified two African Americans as gorillas [51]. In these examples, if systems were able to handle errors by assigning a high level of uncertainty to erroneous predictions, better decisions were achieved and disaster could be avoided. Uncertainty handling has been discussed in CNN structures. Quantify uncertainty in vision tasks was modeled in CNN by Kendall and Gal [52]. Considering uncertainty in the convolution layer was proposed in [53–55]. Also, uncertainty was processed in FC layers and convolution layers; e.g., network morphism; by Wei Ma and Jun Lu [56, 57]. Consideration of uncertainty in CNN layers has some issues such as needing more memory storage and more computation. Some state-of-the art methods use Monte Carlo dropout sampling to extract model uncertainty in CNN [58].

CNNs have been used in NS domain in the literature. In [59], a model for skin dermoscopic was proposed based on NS multiple deep CNNs. Then, NS similarity score (NSS) was applied to determine the number of training set for each epoch during the training process. In [60], a hybrid method using NS and CNNs was introduced for the classification of tumor region areas in MRI images as benign and malignant. For this task, MRI images were segmented using the neutrosophic set expert maximum fuzzy sure entropy (NS-EMFSE) approach. Feature vectors were extracted from segmented brain images and CNNs and classified using SVM and KNN classifiers. Effects of NS on deep transfer learning models Alexnet, Googlenet, and Restnet18 were investigated to categorize X-ray images as COVID-19, normal, pneumonia bacterial, and pneumonia virus [61]. Breast tumor segmentation was formulated as a classification problem in the NS domain for removing speckle noise and enhancing images contrast. The similarity set score and homogeneity value for each pixel were calculated in the NS domain, and then, seed regions are selected by an adaptive Otsu-based thresholding method and morphology operations. Finally, a deep CNN, based on VGG-16 network, was applied for false-positive rate reduction [62].

The proposed scheme in this research, named as neutrosophic convolution neural network (NCNN), is a CNN model with uncertainty handling in NS domain for image classification. This model can be easily adapted for other applications such as speech processing [63]. In the first step, images are mapped to three sets True (T), Indeterminacy (I) and False (F) in NS domain. The proposed definition for data indeterminacy leads to more highlighted noisy pixels in comparison with conventional NS transformation. Then, T and I sets in NS domain are presented as inputs of the proposed two-path NCNN followed by the combination of the outputs of two paths to achieve the final output. In fact, one path is trained with T set and another with I set. It has been shown that when clean and noisy data points with the same label are fed to NCNN, the difference between weight updates for these training samples is not significant. Also, back-propagation equations are computed for each path to show its difference with conventional one-path CNN and it is shown that NCNN is more robust against noisy images in comparison with conventional CNN. Key differences between the proposed NCNN and other CNNs are based on NS. Despite [59–62] which use NS for training set determination in NCNN, indeterminacy set in NS domain is used as a part of CNN training set process. It means that information provided by this set affects weights of neurons in back-propagation process and makes CNNs more robust against noisy samples. It is worth mentioning that path combination in NCNN is different from conventional combination of current CNNs in which

each CNN is considered as an independent path. In NCNN, two paths are trained simultaneously, and each one affects another for updating weights. As it will be concluded from experiments and mathematical analysis, NCNN handles noisy data with higher indeterminacy and it is not overfitted in the training phase as quickly as conventional CNNs.

The rest of this paper is organized as follows: Sect. 2 presents a review on NS sets. The proposed method is discussed in Sect. 3. Experimental results are reported in Sect. 4. Finally, this work is discussed and concluded in Sects. 5 and 6, respectively.

2 Neutrosophic set

NS is an extension of the fuzzy set in which data indeterminacy is considered in addition to truth membership degree. In fact, for each data point, the confidence of assigning truth membership degrees is also considered. Suppose that there are two observers which are going to detect a flower in a picture. The first one assigns 0.8 truth membership degree for the existence of flower, while the second one assigns 0.5. At a higher level, the confidence for the observer's decision is also considered. Therefore, if these confidences for first and second observers are 0.3 and 0.9, respectively, the first observer affects the final decision with more weight. In NS, the confidence level for each observer is considered in the indeterminacy set. These are the main differences between NS and fuzzy sets. Generally, considering set A in NS, each member x in A is denoted by three real subsets: true, false and indeterminacy in the interval [0, 1] referred as T, F and I, respectively. Each element is expressed as $x(t, i, f)$ which means that it is $t\%$ true, $i\%$ indeterminate, and $f\%$ false. In each application, domain experts propose the concept behind true, false and indeterminacy.

To use NS in the image processing domain, each pixel is considered as a data point and should be mapped to NS sets. The first method for this mapping was proposed by Guo [3]. Mapping methods completely depend on the image processing application. An image g in pixel domain is represented with three subsets: T, I and F in NS domain. Therefore, pixel $p(i, j)$ in g is shown with $PNS(i,j) = T(i,j), I(i,j), F(i,j))$ or $PNS(t, i, f)$. T, I and F indicate white, noise and black pixel sets, respectively. $PNS(t, i, f)$ provides useful information about white, noisy and black percentages in this pixel that is $t\%$ to be a white pixel, $i\%$ to be a noisy pixel and $f\%$ to be a black pixel. T, I and F are computed as follows [2, 3].

$$T(i,j) = \frac{\overline{g(i,j)} - \overline{g_{\min}}}{\overline{g_{\max}} - \overline{g_{\min}}} \tag{1}$$

$$F(i,j) = 1 - T(i,j) \tag{2}$$

$$I(i,j) = \frac{\delta(i,j) - \delta_{\min}}{\delta_{\max} - \delta_{\min}} \tag{3}$$

$$\overline{g(i,j)} = \frac{1}{w^2} \sum_{m=-\frac{w}{2}}^{m=\frac{w}{2}} \sum_{n=-\frac{w}{2}}^{n=\frac{w}{2}} g(i+m, j+n) \tag{4}$$

$$\delta = |g(i,j) - \overline{g(i,j)}| \tag{5}$$

where g is grayscale image, $\overline{g}$ is filtered image g with mean filter, w is window size for mean filter, and $\overline{g_{\max}}$ and $\overline{g_{\min}}$ are the maximum and minimum of the $\overline{g}$, respectively. δ is the absolute difference between g and $\overline{g}$, and $\delta_{\max}$ and $\delta_{\min}$ are the maximum and minimum values of δ, respectively.

3 Proposed method

The main contributions of this work are first proposing data indeterminacy concept as well as the truth set in NS domain and then using these concepts in CNN models which leads to a new two-path network. It is theoretically and experimentally proved that the proposed CNN structure handles noisy and outlier data points more efficiently than conventional CNN models and converges quicker. Mentioned contributions are discussed in the following sections.

3.1 Data indeterminacy in NS domain

In the proposed scheme, noisy data points are modeled as high indeterminacy data. This is the first and the most important step since it should interpret noisy pixels for CNN model correctly. Therefore, in the proposed definition for data indeterminacy, it is expected that a high indeterminacy is assigned to noisy pixels and this concept is used for the proposed CNN structure. The proposed method for pixel indeterminacy is summarized in Eqs.(6)–(9):

$$T(i,j) = \frac{\overline{g(i,j)}}{\overline{g_{\text{mean}}}} \tag{6}$$

$$I(i,j) = \frac{\delta(i,j)}{\delta_{\text{mean}}} \tag{7}$$

$$\overline{g(i,j)} = \frac{1}{x \times y} \sum_{m=-\frac{x}{2}}^{m=\frac{x}{2}} \sum_{n=-\frac{y}{2}}^{n=\frac{y}{2}} g(i+m, j+n) \tag{8}$$

$$\delta = \left| g(i,j) - \overline{g(i,j)} \right| \tag{9}$$

where $\overline{g(i,j)}$ is the input image convolved by the mean filter, and $\overline{g_{\text{mean}}}$ and δ_{mean} represent the mean of pixels in $\overline{g}$ and δ, respectively. The reason for dividing by mean is that pixels with higher indeterminacy are revealed brighter. Therefore, as shown in Fig. 1, edges and noisy pixels are appeared brighter in comparison with the basic NS operator.

3.2 Proposed network structure

Presenting indeterminacy (I) and truth (T) sets for CNN structures leads to a two-parallel-path network referred as Neutrosophic CNN. Indeterminacy and truth membership degrees are trained in the first (I-path) and second path (T-path), respectively. The outputs of these paths are combined to compute the final label predicted by the network. The proposed structure is illustrated in Figs. 2 and 3.

It may be worth mentioning that two paths are trained simultaneously in contrast with two-path structures in which each path is trained separately, and then, frozen weights are combined in the final step.

In fact, in the first epoch of training, weights in two paths are updated simultaneously by multiplying the outputs of paths which leads to gradient switching. In this case, paths affect and help each other resulting in robust weight updates.

3.3 Network training

Suppose that W_T and W_I are two weights in I-path and T-path, respectively, shown in Fig. 3. It is clear from the figure that these weights are multiplied by the neurons in the previous layer to construct I and T. Note that I and T are the predicted labels by I-path and T-path networks, respectively. These labels are combined (multiplied) to compute the final label f (see Fig. 3).

To prove how NCNN handles noisy data points with the indeterminacy concept, we discuss how network weights are updated for noisy and clean data points. Here, we explain the weight update for T-path. The scenario for I-path is the same.

Each weight in T-path network is updated by the general update rule as follow:

$$W_T = W_T + \Delta W_T \tag{10}$$

where ΔW_T is computed based on the neural networks update rule in Eq. (11)

$$\Delta W_T = -\eta . \frac{\partial E}{\partial w_T} \tag{11}$$

Applying chain rule to compute $\frac{\partial E}{\partial w_T}$ leads to:

$$\frac{\partial E}{\partial w_T} = \frac{\partial E}{\partial f} . \frac{\partial f}{\partial T} . \frac{\partial T}{\partial w_T} \tag{12}$$

Based on network structure in Fig. 3, f and T are computed by Eq. (13):

$$f = T.I \text{ and } T = \sum w_T . y_T \tag{13}$$

In addition,η is learning rate. Therefore,

$$\frac{\partial f}{\partial T} = I \text{ and } \frac{\partial T}{\partial w_T} = y_T \tag{14}$$

By substituting (14) in (12):

$$\frac{\partial E}{\partial w_T} = \frac{\partial E}{\partial f} . I . y_T \tag{15}$$

To compute $\frac{\partial E}{\partial f}$, chain rule is used again which tends to:

$$\frac{\partial E}{\partial f} = \frac{\partial E}{\partial e} . \frac{\partial e}{\partial p} . \frac{\partial p}{\partial f} \tag{16}$$

In Eq. (16), p is the final network output, e represents difference between data label L and predicted label p. E is sum squared error. These parameters are defined by Eq. (17):

$$e = p - L(\text{Label}) \ , \ E(n) = \frac{1}{2} \sum e^2(n) \text{ and } p = \varphi(f) \tag{17}$$

Applying derivation rules leads to Eq. (18):

(a) (b) (c)

Fig. 1 Proposed pixel Indeterminacy: **a** Input image, **b** Indeterminacy matrix computed by basic NS and **c** Indeterminacy matrix computed by the proposed method

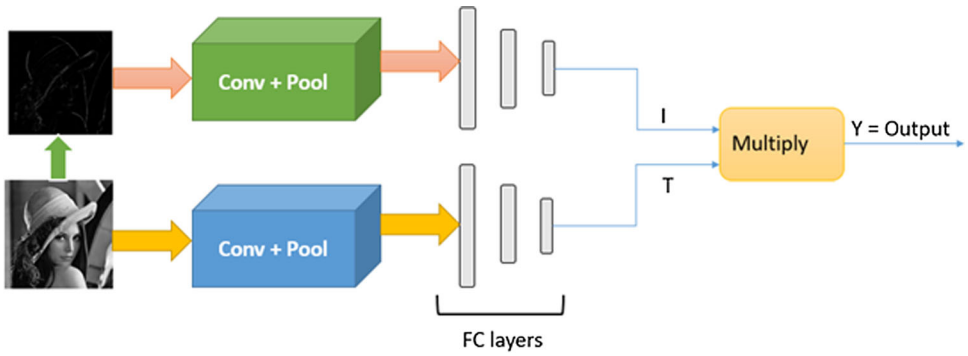

Fig. 2 The proposed NCNN model

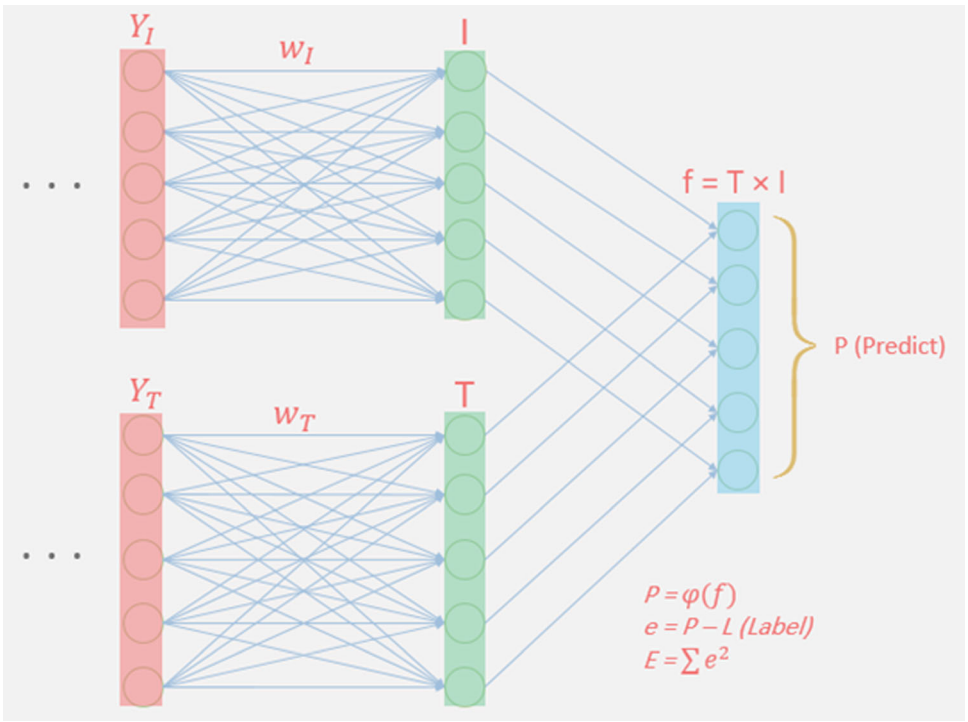

Fig. 3 Network weights for NCNN

$$\frac{\partial E}{\partial e} = e,\ \frac{\partial e}{\partial p} = 1 \text{ and } \frac{\partial p}{\partial f} = \varphi'(f) \tag{18}$$

Therefore, $\frac{\partial E}{\partial f}$ is computed as follow:

$$\frac{\partial E}{\partial f} = e.\varphi'(f) \tag{19}$$

By substituting (19) in (15), the update rule is done by:

$$\frac{\partial E}{\partial w_T} = e.\varphi'(f).I.y \tag{20}$$

$$\Delta w_T = -\eta.e.\varphi'(f).I.y \tag{21}$$

It can be concluded from these equations that weight updates for neurons in T-path are affected by I subset in NS domain and vice versa.

3.4 Network behavior analysis

Finally, in this section, it is explained how the proposed idea for NCNN leads to better results in comparison with CNN. The main difference between the weight update rule in NCNN and conventional CNN is the parameter I. To find out the effect of I in Δw_t computation, Eq. (21) is extended to Eq. (22) as follow:

$$\begin{aligned}\Delta w_I &= \Big[-\eta.(p-L).\varphi'(f).I.y\Big] \\ &\approx \Big[-\eta.(f-L).\varphi'(f).I.y\Big] \\ &\approx [(f-L).I] = |f-L|.I \\ &= [|T.I-L|.I] \\ &= |T.I^2 - L.I|\end{aligned} \tag{22}$$

here there are two possible class labels 0 and 1 for L. Therefore, separated values in Eq. (23) can be considered in Eq. (22):

$$\begin{cases} \left|I - T.I^2\right|, & \text{if } L=1 \\ \left|T.I^2\right|, & \text{if } L=0 \end{cases} \tag{23}$$

Since T and I are the output of softmax layer, these parameters are also 0 and 1, therefore:

$$\begin{cases} \left|T.I^2 - I\right|, & \text{if } L=1 \\ \left|T.I^2\right|, & \text{if } L=0 \end{cases} \tag{24}$$

Equation (24) in NCNN is compared with Eq. (25) in conventional CNN which is as follows:

$$\begin{cases} |1-T|, & \text{if } L=1 \\ |T|, & \text{if } L=0 \end{cases} \tag{25}$$

One of the main challenges of CNN is its sensitiveness against noisy data. It means that when clean data are fed as input and its label is predicted correctly, network weights are updated slightly. If the same data (with the same label) with noise are fed to the network, it tries to update weights significantly. It means that in CNN, weight update for clean and noisy data with the same label is significantly different. This behavior misleads the network for noisy data. Here, it is shown that NCNN handles noisy data and does not update weights for clean and noisy data with the same label differently. To compare the amount of weight updates for clean and noisy data in NCNN and CNN, if true label 1 is considered for noisy and clean data, weight updates for clean and noisy data in CNN are $(1-T_c)$ and $(1-T_n)$, respectively. Therefore, the difference between weight updates for clean and noisy data in CNN is:

$$T_c - Tn \tag{26}$$

Also, weights update for clean and noisy data in NCNN is $(I_c - T_c I_c^2)$ and $(I_n - T_n I_n^2)$, respectively. The difference between weight update for clean and noisy data in NCNN is as follows:

$$I_n - T_n I_n^2 - I_c + T_c I_c^2 = \Big(I_n - I_c\Big) + \Big(T_c I_c^2 - T_n I_n^2\Big) \tag{27}$$

If it is proved that (27) is less than (26), it is equivalent to this fact that NCNN does not make a difference in weight update for clean and noisy data in comparison with CNN. It means that NCNN is more robust against noisy data.

It is clear that for a data point with label 1, I_c and T_c are bigger than I_n and T_c. Since $(I_n - I_c)$ is a negative value, to show that (27) < (26), it is enough to show that:

$$\Big(T_c I_c^2 - T_n I_n^2\Big) < \Big(T_c - T_n\Big) + \Big(I_c - I_n\Big) \tag{28}$$

As it will be shown numerically in the discussion section, if the I-path and T-path networks are trained separately, they almost predict the same label. Therefore, elements in pair T_c and I_c as well as pair T_n and I_n have near values. Note that in each image, I-path learns labels from noisy pixels and pixels with a high gradient, while T-path learns labels from all pixels in the image. Therefore, in NCNN, both noisy pixels (with a high indeterminacy) and clean pixels are used to predict the label. From the result of Table 8 in the Discussion section, it can be concluded that:

$$\Big(T_c - T_n\Big)\Big(T_c^2 + T_n T_c + T_n^2\Big) < 2\Big(T_c - T_n\Big) \tag{29}$$

$$\Big(T_c^2 + T_c T_n + T_n^2\Big) < 2 \tag{30}$$

It is worth mentioning that inequality (30) does not satisfy when $(T_n, T_c) < 0.8$. Clean data have high values for T_n and T_c. Therefore, inequality (30) is not satisfied for clean data. Therefore, in clean datasets, NCNN not only does not improve the performance but also has lower accuracy in comparison with CNN. For noisy datasets, T_n and T_c are far from each other and T_n leads to values near to zero. Therefore, inequality (30) is almost satisfied for noisy data. As experimental results will also show, this is the main reason that NCNN handles noisy data points and has a very good performance for noisy data classification.

4 Experimental results

4.1 Dataset

NCNN model is proposed for the image classification task. Therefore, to show the effectiveness of NCNN, it has been evaluated on MNIST, CIFAR-10 and CIFAR-100 datasets. To evaluate the robustness of NCNN and other models against noise, Gaussian noise with different means and standard deviations is added to images. For each dataset in each model, 5 levels of noise are considered. Since we evaluate 2 models and 2 datasets for each model, a total of 20 noise levels are used. In this section, each dataset is described briefly.

4.1.1 MNIST

The first dataset is MNIST (Modified National Institute of Standards and Technology database), which is a common dataset used for evaluating various image processing and machine learning methods. This dataset contains 70,000 images of handwritten digits divided into 60,000 training images and 10,000 testing images. All images in MNIST were normalized to fit into a 28 × 28 pixel bounding box and anti-aliased in grayscale levels [64].

4.1.2 CIFAR-10

The second dataset is CIFAR-10 (Canadian Institute For Advanced Research) which is the most widely used for research in machine learning and computer vision algorithms. The CIFAR-10 dataset contains 60,000 color images with a dimension of 32 × 32 in 10 different classes including airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks with 6000 images in each class. The main purpose of CIFAR-10 is to teach a computer how to recognize objects. This dataset allows researchers to quickly try different machine learning and machine vision algorithms since images are in low-resolution (32 × 32). Different models of CNNs have tried to achieve the best accuracy in recognizing CIFAR-10 images [65].

4.1.3 CIFAR-100

CIFAR-100 is the same as CIFAR-10 except for the number of classes and samples in each class. CIFAR-100 contains 100 classes with 600 samples in each class. The number of training and testing samples in each class is 500 and 100, respectively. Classes are further divided into 20 superclasses. Therefore, each sample is assigned to "fine" and "coarse" labels which are the label of class and superclass, respectively. Classes are aquatic mammals, fish, flowers, food containers, fruit and vegetables, household electrical devices, household furniture, insects, large carnivores, large man-made outdoor things, large natural outdoor scenes, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals, trees, vehicles 1 and vehicles 2 [66].

4.2 CNN models

NCNN is a structure that can be applied to any CNN model. It means that CNN models with a different number of layers can be placed in NCNN structure. In this section, we want to show how NCNN can be applied to any CNN model to improve its robustness against noisy data with higher indeterminacy. For this task, two basic CNN models are considered. The first one is the proposed 11-layer CNN model, and the second one is the reference model VGG-net. Applying NS-based two parallel paths idea to the first and second mentioned models constructs NCNN and NVGG-net, respectively. NCNN and CNN have been applied to MNIST and CIFAR-10 datasets. VGG-net and NVGG-net are evaluated on CIFAR-10 and CIFAR-100 datasets. All CNN structures have been implemented with python and TensorFlow libraries installed in Linux operating system on a machine with 3.26 GHz Corei7 CPU, 32 GB of DDR4 RAM and GeForece 1080ti 8GB RAM GPU.

4.2.1 The first model

The first proposed model is an 11-layer CNN shown in Fig. 4. In this structure, "5 × 5 conv, 64" means convolution layer with 64 filters in dimension 5 × 5. "3 × 3 Pooling" means polling layer with a 3 × 3 filter, and "FC, 384" means fully connected layer with 384 neurons. In each convolution layer, the step size is 1 × 1; therefore, the first and second dimensions of the input image remain fixed, while the third layer may be changed depending on the number of convolution layers. In the pooling layer, since the step size is 2 × 2, the first and second dimensions are reduced by half. Finally, there are three fully connected layers with different numbers of neurons shown in Fig. 4. After each convolution and pooling layer, activation function "RELU" is used.

NCNN and CNN have been evaluated on MNIST and CIFAR-10 datasets. In each experiment, a level of Gaussian noise is considered which makes different combinations of training and test sets with noise levels including "clean," "NT1: Mean = 0.8, Std = 0.6," "NT2: Mean = 0.6, Std = 0.9," "NT3: Mean = 0.7, Std = 1.2" and "NT4: Mean = 0.8, Std = 1.6." Therefore, from each dataset, we obtain 25 datasets. The first dataset is the original dataset with clean training and clean test data. The 25th dataset is a dataset in which training and test data are added with NT4 noise. The classification accuracies on MNIST test data are reported in Table 1. In all tables, vertical and horizontal data types are used for training and test sets, respectively.

It can be concluded from the reported results in Table 1 (where it is also shown mathematically in Sect. 3) that if both training and test data are clean, CNN outperforms NCNN slightly. The best performance of NCNN is revealed in cases which noise is presented either in training or test data. For example, when training data are clean and test data are NT4, NCNN outperforms CNN with 10.02%. This behavior of NCNN is shown in Fig. 5. It can be seen that the higher amount of noise, the better improvement in NCNN (in comparison with CNN) is achieved. These results verify the main advantage of NCNN for uncertainty handling in noisy data.

Fig. 4 The first proposed CNN model

Table 1 Results of CNN and NCNN for MNIST

		Clean	NT1	NT2	NT3	NT4	Average
Clean	CNN	98.97	49.51	35.23	24.27	18.63	45.32
	NCNN	98.47	48.83	43.4	32.84	28.65	50.43
NT1	CNN	97.16	93.96	80.67	63.07	46.2	76.21
	NCNN	97.9	94.86	85.85	70.3	54.88	80.75
NT2	CNN	96.58	94.41	87.95	75.36	57.31	82.32
	NCNN	95.66	94.52	88.21	75.41	59.01	82.56
NT3	CNN	94.72	89.16	86.32	76.99	61.78	81.79
	NCNN	90.17	92.6	87.39	76.85	62.26	81.85
NT4	CNN	90.24	90.05	84.09	74.87	61	80.05
	NCNN	88.85	91.28	85.69	76.06	64.83	81.34
Average	CNN	95.534	83.418	74.852	62.912	48.984	–
	NCNN	94.21	84.418	78.108	66.292	53.926	–

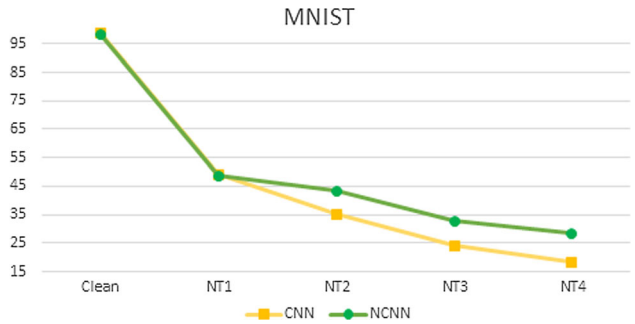

Fig. 5 NCNN and CNN accuracies with clean training and noisy test data for MNIST

Another conclusion from reported results on MNIST dataset is that training network with noisy data makes it more robust against noisy data in the test phase. Figure 6 illustrates the results of CNN and NCNN with NT3 training data and all noise levels for test data. In this experiment, similar to Fig. 5, applying more noise (NT3) in test data leads to a better improvement (3%) in the results of NCNN in comparison with CNN. When clean data are considered in the test phase, CNN outperforms NCNN by about 1%.

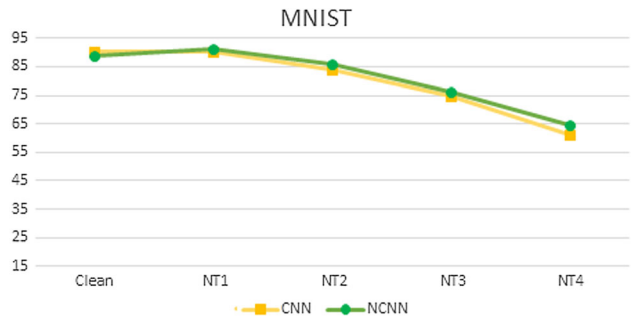

Fig. 6 NCNN and CNN accuracies with noisy training and noisy test data for MNIST

CIFAR-10 is the second dataset considered for evaluation of the first model. By applying 5 models of data including "Clean," "NT5: Mean = 0.1, Std = 0.1," "NT6: Mean = 0.4, Std = 0.2," "NT7: Mean = 0.2, Std = 0.4" and "NT8: Mean = 0.3, Std = 0.6," 25 datasets are created. The classification accuracies on test data for CIFAR-10 are reported in Table 2. Here, each column shows experiments with the same training data and test data with different levels of noise, while each row is for the training data with different noise levels and the same test data. In CIFAR-10, the best improvement of NCNN (3.74%) appeared in the case that training and test data are contaminated with NT7 noise.

The two columns with "Clean" header report the classification accuracies of NCNN and CNN with the clean training data and test data with all noise levels. These classification accuracies are depicted in Fig. 7. For these experiments, NCNN outperforms CNN by 3.45% for test data with NT6 noise. In CIFAR-10 dataset, the trend is the same with MNIST with a slower slope. It means that

Table 2 Results of CNN and NCNN for CIFAR-10

		Clean	NT5	NT6	NT7	NT8	Average
Clean	CNN	71.35	63.13	45.94	23.61	16.84	44.174
	NCNN	72.76	65.12	49.39	22.73	19.23	45.846
NT5	CNN	68.56	67.3	58.29	35.41	23.17	50.546
	NCNN	71.06	70.68	61.39	36.91	22.27	52.462
NT6	CNN	62.67	62.78	61.59	40.29	29.04	51.274
	NCNN	64.08	64.75	63.83	43.33	28.83	52.964
NT7	CNN	51.72	49.08	36.06	51.24	43.48	46.316
	NCNN	53.77	51.45	36.38	54.98	46.03	48.522
NT8	CNN	41.77	39.45	23.96	48.22	45.5	39.78
	NCNN	44.77	41.46	24.08	48.46	48.42	41.438
Average	CNN	59.214	56.348	45.168	39.754	31.606	–
	NCNN	61.288	58.692	47.014	41.282	32.956	–

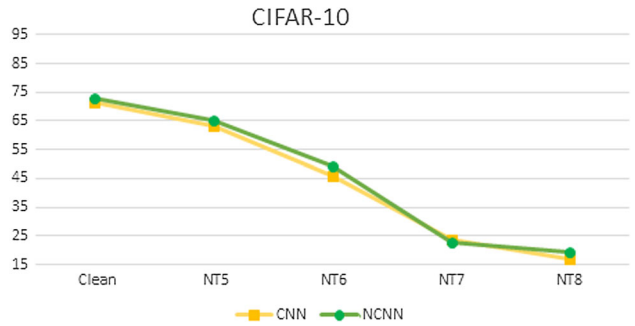

Fig. 7 NCNN and CNN accuracies with clean training and noisy test data for CIFAR-10

MNIST is more affected by noisy test data in comparison with MNIST.

Evaluation of NCNN and CNN with NT7 training data and different noise levels in test data is shown in Fig. 8. In these cases, instead of a down trend toward increasing noise, the trend is upward after NT6 which means the similarity between noise applied to training and test data.

Generally, as it is illustrated in Figs. 9, 10, 11 and 12, if the noise is increased from NT5 to NT8, the accuracies of both networks are decreased with negative slopes in all charts. In cases where the noise level in training and test is

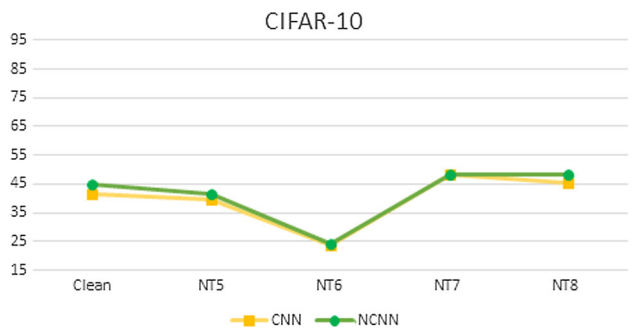

Fig. 8 NCNN and CNN accuracies with noisy training and noisy test data for CIFAR-10

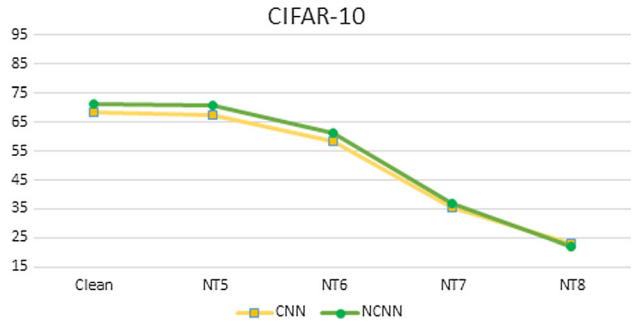

Fig. 9 NCNN and CNN accuracies for NT5 training data

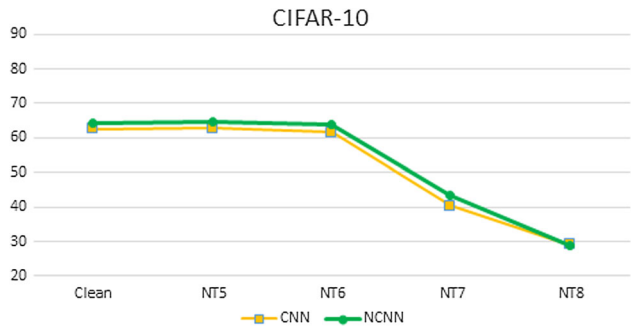

Fig. 10 NCNN and CNN accuracies for NT6 training data

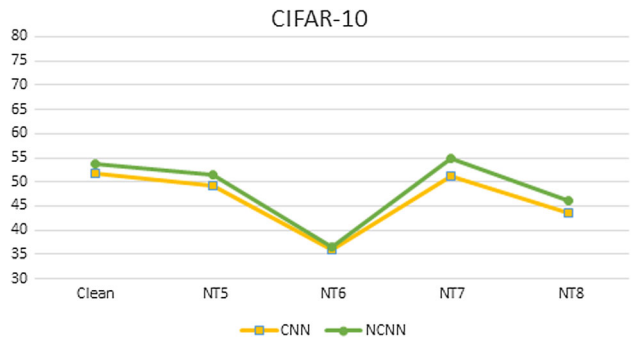

Fig. 11 NCNN and CNN accuracies for NT7 training data

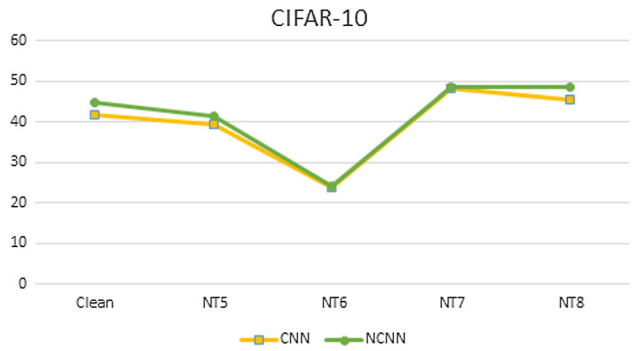

Fig. 12 NCNN and CNN accuracies for NT8 training data

the same, not only there is no significant decrement in the accuracy, but also there is an improvement in some cases. For example, in Fig. 10, since the noise level in training is NT6, the accuracy is not decreased noticeably when the noise level is increased from NT5 to NT6. In Fig. 11 which represents training data with noise level NT7, when noise is increased from NT6 to NT7, the accuracy is improved significantly.

4.2.2 The second model (VGG-Net)

The second base model is VGG-Net [39] as illustrated in Fig. 13. This model has 16 layers with "RELU" activation functions. After convolution and fully connected layers, batch normalization is used. For convolution layers which are not followed by pooling layers, dropout is considered with the coefficients of 0.4 and 0.3. Also, in all fully connected layers, except the last one, dropout with the coefficient of 0.5 is used. In convolution layers, since the step size is 1x1, the dimensions of images before and after passing these layers are the same, while in the dropout layer, they are decreased by half because of step size 2x2. Applying NS-based two parallel paths to VGG-Net creates NVGG-Net.

VGG-Net and NVGG-Net have been evaluated on CIFAR-10 and CIFAR-100 datasets. In CIFAR-10, 5 levels of noise including "Clean," "NT9: Mean =0.2, Std=0.1," "NT10: Mean = 0.1, Std = 0.2," "NT11: Mean = 0.2, Std = 0.3" and "NT12: Mean = 0.3, Std = 0.5" are considered which can be applied to either training or test data. Table 3 reports the results of VGG-Net and NVGG-Net for CIFAR-10. NVGG-Net achieves better classification accuracy rather than VGG-Net except for pairs (Clean, NT10), (NT9, NT11), (NT11, NT10), (NT11, Clean) and (NT11, NT11) out of all 25 pairs for (Training, Test). The best average improvement of 3.09% was obtained by NVGG-Net in cases with NT12 test set and noisy training sets (with all levels of noise). In these cases, an improvement of 4.9% was achieved by NVGG-Net in pair (Clean, NT12).

Figures 14, 15, 16, 17 and 18 show how networks are affected by changing training and test data and can be very useful. It can be that NVGG-Net outperforms VGG-Net in the majority of cases especially when the noise in the test set is NT12. In both networks, if the network is trained with more noisy data, it is more robust against increasing noise in test data. Therefore, Fig. 14 for clean training data has the most descent in the slope, while Fig. 18 for NT12 training data is near to a horizontal line (the least descent in slope). Also, using the same noise in training and test data makes the network more robust.

For CIFAR-100, different noises were applied in cases "clean," "NT13: Mean = 0.1, Std = 0.1," "NT14: Mean = 0.2, Std = 0.1," "NT15: Mean = 0.1, Std = 0.2" and "NT16: Mean = 0.2, Std = 0.2." Table 4 reports the results of VGG-Net and NVGG-Net for CIFAR-100. The best improvement of 4.5% was obtained by NVGG-Net in pair (NT14,NT15). Also, the best average improvement of 2.57% was obtained by NVGG-Net in cases with NT16 test set and noisy training sets (with all levels of noise).

Finally, as it is shown in Fig. 19, networks training with the highest level of noise (NT16) does not make a significant change in the robustness of the networks. The reason is that the amount of noises added to CIFAR-100 is much less than those in CIFAR-10.

4.3 Window size effect

Window size has an important effect on computing indeterminacy and true sets in image transformation to NS domain. In this section, the effect of window size is evaluated on MNIST dataset for the first model and CIFAR-10 and CIFAR-100 dataset for the second model. Classification accuracies with different window sizes are reported in

Fig. 13 VGG-Net structure

Table 3 Results of VGG-Net and NVGG-Net for CIFAR-10

		Clean	NT9	NT10	NT11	NT12	Average
Clean	VGG-Net	92.32	75.28	54.18	43.6	30.54	59.18
	NVGG-Net	92.52	78.22	53.9	47.94	35.44	61.6
NT9	VGG-Net	88.31	84.67	72.32	59.64	42.54	69.5
	NVGG-Net	88.53	86.15	73.62	59.13	44.04	70.29
NT10	VGG-Net	83.6	81.8	76.76	67.57	52.23	72.39
	NVGG-Net	84.83	82.48	78.56	69.38	55.09	74.07
NT11	VGG-Net	79.14	77.93	75	68.83	56.18	71.42
	NVGG-Net	78.12	79.44	74.18	68.18	59.59	71.9
NT12	VGG-Net	69.82	69.47	68.32	65.16	57.66	66.09
	NVGG-Net	71.5	70.55	69.05	67.22	60.44	67.75
Average	VGG-Net	82.638	75.28	69.316	60.96	47.83	–
	NVGG-Net	83.1	78.22	69.86	62.37	50.92	–

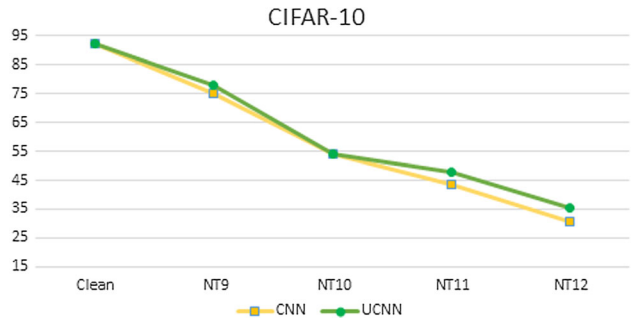

Fig. 14 NVGG-Net and VGG-Net accuracies with Clean training data

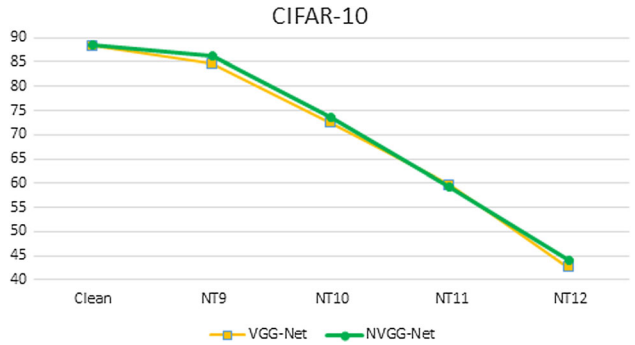

Fig. 15 NVGG-Net and VGG-Net accuracies with NT9 training data

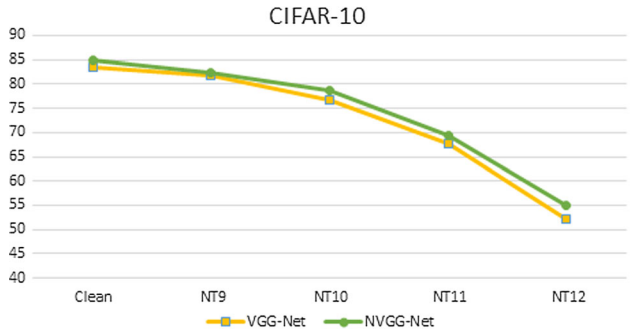

Fig. 16 NVGG-Net and VGG-Net accuracies with NT10 training data

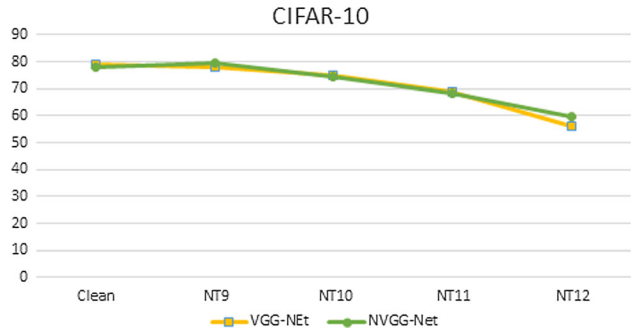

Fig. 17 NVGG-Net and VGG-Net accuracies with NT11 training data

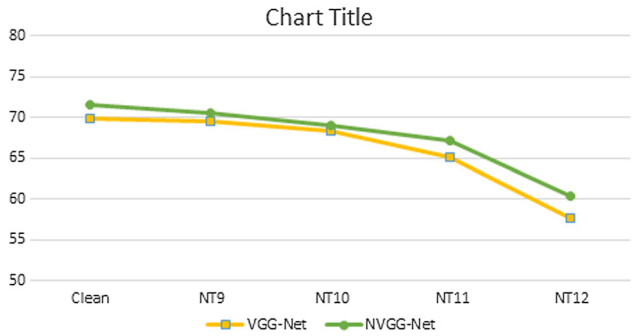

Fig. 18 NVGG-Net and VGG-Net accuracies with NT12 training data

Table 5. It can be concluded that for clean data, smaller window sizes have better performance, while for noisy data, bigger window sizes are preferred. The reason is that with a bigger window size, more general information around each pixel is considered in NS domain and the robustness against noisy pixels in local neighbors is increased. On the other hand, for clean data, we need only local information and we do not need general information around each pixel. Window size 3×3 has the best performance in all experiments in average.

Table 4 Accuracies of VGG-Net and NVGG-Net for CIFAR-100

		Clean	NT13	NT14	NT15	NT16	Average
Clean	VGG-Net	69.39	56.8	56.05	46.06	46.03	54.87
	NVGG-Net	69.51	57.82	55.53	49.72	50.64	56.64
NT13	VGG-Net	63.12	59.67	59.38	50.2	50.64	56.6
	NVGG-Net	63.075	61.42	61.72	52.96	52.63	58.36
NT14	VGG-Net	63.93	59.2	59.19	49.75	50.98	56.61
	NVGG-Net	64.14	61.69	61.78	54.25	53.82	59.13
NT15	VGG-Net	57.43	55.68	55.51	50.52	51.3	54.09
	NVGG-Net	60.87	55.51	56.32	51.82	52.63	55.43
NT16	VGG-Net	57.1	55.54	55.92	50.85	50.94	54.07
	NVGG-Net	58.75	57.33	56.28	50.49	53.01	55.17
Average	VGG-Net	62.194	57.378	57.21	49.476	49.978	–
	NVGG-Net	63.269	58.754	58.326	51.848	52.546	–

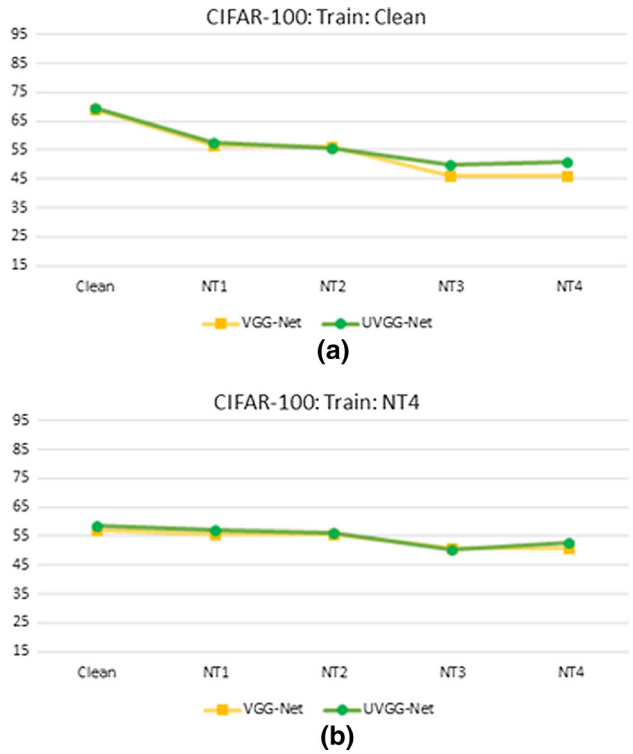

Fig. 19 NVGG-Net and VGG-Net accuracies with training data: (a) Clean and (b) NT4

4.4 Combination method effect

The proposed model for indeterminacy consideration in CNN is a two parallel paths network, one path with the input of indeterminacy set I and another with true set T. Combination method of two paths is very important. Here, three combination methods including Maximum, Mean and Product are assessed. Table 6 reports the classification accuracies with different combination methods applied to MNIST dataset for the first model and CIFAR-10 and CIFAR-100 dataset for the second model. In these experiments, window sizes are selected as 3×3. It is clear from the reported results that Product method has the best accuracy among other methods The effect of Mean and Maximum combination methods will be mathematically analyzed in the Discussion section.

4.5 Training methods

The last experiments are performed to evaluate training methods. There are two options to train parallel paths in the network. The first one is an ensemble method in which two paths are trained separately and then their outputs are combined. In the second method, two paths are trained simultaneously. The first and second methods are called

Table 5 Classification accuracies with different window sizes

Dataset	MNIST			CIFAR-10			CIFAR-100		
Window size	2×2	3×3	4×4	2×2	3×3	4×4	2×2	3×3	4×4
Clean	50.89	50.43	50.4	62.03	61.6	61.62	56.69	56.64	56.11
NT1	80.88	80.75	80.79	70.05	70.29	69.77	58.58	58.36	57.9
NT2	82.4	82.56	82.5	73.88	74.07	73.92	58.73	59.13	59.1
NT3	80.1	81.65	81.99	70.93	71.9	71.99	54.11	55.43	55.61
NT4	79.99	81.34	82.01	66.01	67.75	68.01	53.89	55.17	55.23
Average	74.85	75.34	75.53	68.58	69.12	69.06	56.4	56.94	56.79

Table 6 Classification accuracies with different combination methods

Dataset	MNIST			CIFAR-10			CIFAR-100		
	Max	Mean	Prod	Max	Mean	Prod	Max	Mean	Prod
Clean	49.32	50.12	50.43	60.22	61.01	61.60	55.87	56.83	56.64
NT1	76.33	79.41	80.75	69.76	69.18	70.29	57.68	57.19	58.36
NT2	81.00	81.92	82.56	73.10	73.55	74.07	58.19	58.37	59.13
NT3	80.40	81.77	81.65	69.94	69.22	71.90	54.78	55.10	55.43
NT4	79.33	81.10	81.34	66.10	67.45	67.75	54.24	53.72	55.17
Average	73.27	74.86	75.34	67.82	68.08	69.12	56.15	56.24	56.94

"Separate Training" and "Joint Training," respectively. In these experiments, window size 3×3 and Product combination methods are considered. As reported in Table 7, the second training method has better classification accuracy with the 2.13% of improvement on CIFAR-100 dataset. Therefore, we used the second training in this research.

The source code of NCNN and other CNN models used in this research as well as three datasets in clean and noisy formats will publicly available online as supplementary material in the journal website and Github after the publication of the paper.

4.6 Computational cost

In this section, the computational cost of CNN and NCNN models is analyzed. For this task three datasets CIFAR-Clean, CIFAR10-NT6 and CIFAR10-NT8 are selected for cost evaluation. All models have been implemented in python programming language with TensorFlow and Pandas libraries with NDIVIA 1080ti graphical processing unit (GPU), 32G RAM DDR4, corei7 CPU and 500G hard drive SSD. Time costs for running 1 patch from all 10000 data points in test set for CNN and NCNN are 1.18 and 2.08 seconds, respectively. Since NCNN uses two paths T-path and I-path, the number of parameters (weights) is as twice as the number of parameters in CNN with one path, so NCNN training time is more than CNN one.

The number of iterations as well as the accuracy of each iteration in the training phase is shown in Figs. 20, 21 and 22 for CIFAR-Clean, CIFAR10-NT6 and CIFAR10-NT8 datasets, respectively. As it is clear from these figures, CNN is overfitted in iteration 4000, while NCNN leads to better results in the training phase which can be considered as an advantage for NCNN.

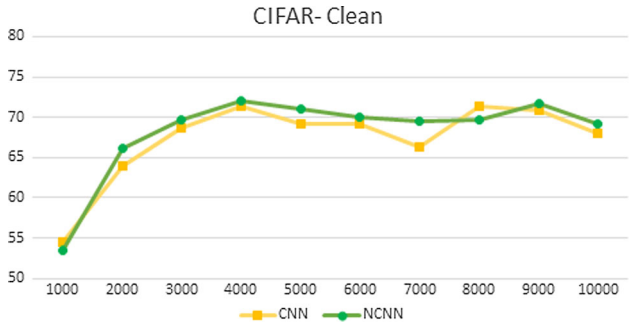

Fig. 20 Accuracy versus number of iterations in CIFAR-Clean dataset

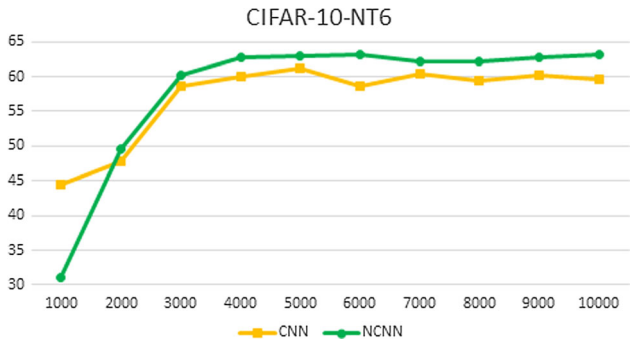

Fig. 21 Number of iteration versus accuracy in CIFAR-10-NT6 dataset

Table 7 Classification accuracies with different training methods

Dataset	MNIST		CIFAR-10		CIFAR-100	
Training method	Separate	Joint	Separate	Joint	Separate	Joint
Clean	50.12	50.43	60.99	61.80	55.32	58.64
NT1	79.32	80.75	70.01	70.29	56.75	58.36
NT2	81.02	82.56	73.12	74.07	58.22	59.13
NT3	79.98	81.65	70.27	71.90	54.68	57.43
NT4	80.11	81.34	65.91	67.75	53.11	55.17
Average	74.11	75.34	68.06	69.16	55.61	57.74

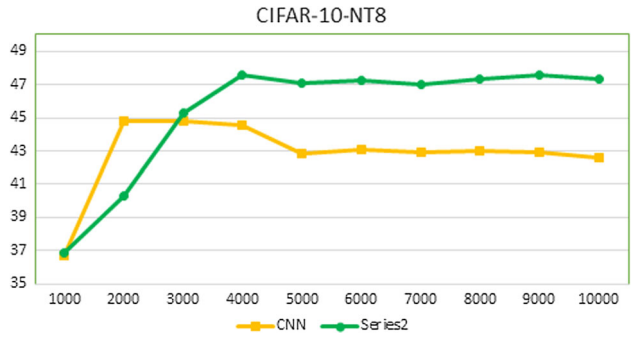

Fig. 22 Number of iteration versus accuracy in CIFAR-10-NT8 dataset

5 Discussion

To prove inequalities (29)–(30), it was supposed that the output of networks in the first and second paths is highly correlated. Note that the input of the first and second paths is true and indeterminacy sets, respectively. It means that the label of data can be learned either from its true set or indeterminacy set. This behavior is very useful especially for noisy data in which the label is also learned from noisy pixels with higher indeterminacy. To support this consideration experiments have been done as follows. First, MNIST, CIFAR-10 and CIFAR-100 with the highest level of noise NT4 in both training and test sets are considered. Then, the I-path ad T-path in NCNN and NVGG-Net are separated, and the outputs of each path are reported in Table 8. As it is clear from Table 8, the output of two networks is highly correlated.

It may be worth mentioning that in backpropagation, if the product combination method is used, *gradient switcher* is generated. It means that in two parallel paths, backpropagated error of each path is switched to another one. For Mean combination method, *gradient distributor* is constructed in backpropagation in which backpropagated error is distributed equally in two paths. Here, the weights update rule is investigated for Mean combination method:

$$f = \frac{T+I}{2} \text{ and } T = \sum w_T.y_T \tag{31}$$

Therefore,

Table 8 Correlation between the outputs of two paths

Dataset	MNIST		CIFAR-10		CIFAR-100	
	Clean	Noisy	Clean	Noisy	Clean	Noisy
T-path output	98.97	61	92.32	57.66	69.39	50.94
I-path output	98.52	60.55	90.01	57.72	68.91	50.36

$$\frac{\partial f}{\partial T} = \frac{1}{2} \text{ and } \frac{\partial T}{\partial w_T} = y_T \tag{32}$$

By substituting (32) in (12):

$$\frac{\partial E}{\partial w_T} = \frac{1}{2}\frac{\partial E}{\partial f}.y_T \tag{33}$$

To compute $\frac{\partial E}{\partial f}$, chain rule is used again which reaches to:

$$\frac{\partial E}{\partial f} = \frac{\partial E}{\partial e}\frac{\partial e}{\partial p}\frac{\partial p}{\partial f} \tag{34}$$

In Eq. (34), p is the final output of the network, e represents the difference between data label L and predicted label p. E is the sum square error. These parameters are defined as follows:

$$e = p - L\ ,\ E(n) = \frac{1}{2}\sum e^2(n) \text{ and } p = \varphi(f) \tag{35}$$

Applying derivation rule:

$$\frac{\partial E}{\partial e} = e\ ,\ \frac{\partial e}{\partial p} = 1 \text{ and } \frac{\partial p}{\partial f} = \varphi\prime(f) \tag{36}$$

Therefore, $\frac{\partial E}{\partial f}$ is computed as follow:

$$\frac{\partial E}{\partial f} = e.\varphi\prime(f) \tag{37}$$

By substituting (37) in (33), the update rule is performed by:

$$\frac{\partial E}{\partial w_T} = e.\frac{1}{2}\varphi\prime(f).y \tag{38}$$

$$\Delta w_T = -\frac{1}{2}.\eta.e.\varphi\prime(f).y \tag{39}$$

The main difference in the weight update equation of NCNN with Mean combination method and CNN is that is error term e. In this case, $e = p - l$ and $p = \varphi(T + I)$. Therefore, error e is affected by both T and I sets. This weights update rule can be interpreted as: The bigger difference between predicted labels by two parallel paths (T-path and I-path) and output (real) label leads to bigger amounts of updates for weights. If the predicted label by either T-path or I-path is almost the same with the real output label, low amount of error is back propagated in comparison with CNN. Finally, if both T-path and I-path predict the label close to the real output label, the error is decreased significantly and weights are updated slightly. It means that NCNN considers the predicted output of both T-path and I-path for updating weights. Finally, if the Max combination method is selected, a gradient router is constructed which means that the error is back propagated in a path in which its output is maximum. The weights of another path are not updated. If a path wins repeatedly in

the first interactions, this path will be updated in the next iterations and weights of another path are not updated. This behavior leads the network to work similar to one-path CNN with the input of T or I. This is the main issue of the Max method. As it can be seen from the reported results in Table 6, this combination method has lower accuracy compared with the Mean and Product combination methods.

6 Conclusion

In this work, a two parallel paths NCNN model with data indeterminacy handling was proposed in NS domain for image classification task. In the proposed scheme, images were transformed to NS domain, and then, T and I sets were placed as inputs of T-path and I-path in NCNN. The output of two paths was combined to make the final output. In the proposed structure, two paths are trained simultaneously, and each path helps another for updating weights in backpropagation steps. Therefore, path combination is not a combination of previously fine-tuned structures. Also, computational cost analysis showed that NCNN is not overfitted in the training phase as quick as conventional CNN. Although the cost of NCNN is more than CNN in each training epoch. The effectiveness of NCNN to handle noisy data with higher indeterminacy was analyzed mathematically to show how weights are updated in each path. NCNN was further evaluated contaminated on three different datasets contaminated with 20 levels of Gaussian noise. Results showed that NCNN outperforms CNN models significantly in noisy test data. Future efforts will be directed toward using the proposed model in other applications such as noisy speech recognition by proposing indeterminacy for noisy speech spectrogram in NS domain. Finally, using the proposed model in other deep neural networks such as LSTM can be considered as other future works.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353
2. Wang H, Smarandache F, Sunderraman R, Zhang YQ (2005) Interval neutrosophic sets and logic: theory and applications in computing: theory and applications in computing, vol 5. Infinite Study
3. Guo Y, Cheng H-D (2009) New neutrosophic approach to image segmentation. Pattern Recogn 42(5):587–595
4. Smarandache F (1999) A unifying field in logics: neutrosophic logic. Philosophy. American Research Press, New Mexico, pp 1–141
5. Smarandache F (2003) A unifying field in logics: neutrosophic logic. neutrosophy, neutrosophic set, neutrosophic probability: neutrosophic logic: neutrosophy, neutrosophic set, neutrosophic probability. Infinite Study
6. Smarandache F (2005) A unifying field in logics: neutrosophic logic. neutrsophic logic. neutrosophy, neutrosophic set, neutrosophic probability, infinite study, neutrosophy, neutrosophic set, neutrosophic probability
7. Zhang M, Zhang L, Cheng H-D (2010) A neutrosophic approach to image segmentation based on watershed method. Signal Process 90(5):1510–1517
8. Sengur A, Guo Y (2011) Color texture image segmentation based on neutrosophic set and wavelet transformation. Comput Vis Image Underst 115(8):1134–1144
9. Heshmati A, Gholami M, Rashno A (2016) Scheme for unsupervised colour-texture image segmentation using neutrosophic set and non-subsampled contourlet transform. IET Image Proc 10(6):464–473
10. Guo Y, Akbulut Y, Şengür A, Xia R, Smarandache F (2017) An efficient image segmentation algorithm using neutrosophic graph cut. Symmetry 9(9):185
11. Guo Y, Şengür A, Ye J (2014) A novel image thresholding algorithm based on neutrosophic similarity score. Measurement 58:175–186
12. Guo Y, Şengür A (2014) A novel image edge detection algorithm based on neutrosophic set. Comput Electr Eng 40(8):3–25
13. Rashno A, Nazari B, Koozekanani DD, Drayna PM, Sadri S, Rabbani H, Parhi KK (2017) Fully-automated segmentation of fluid regions in exudative age-related macular degeneration subjects: kernel graph cut in neutrosophic domain. PLoS ONE 12(10):e0186949
14. Rashno A, Koozekanani DD, Drayna PM, Nazari B, Sadri S, Rabbani H, Parhi KK (2018) Fully automated segmentation of fluid/cyst regions in optical coherence tomography images with diabetic macular edema using neutrosophic sets and graph algorithms. IEEE Trans Biomed Eng 65(5):989–1001
15. Rashno A, Parhi KK, Nazari B, Sadri S, Rabbani H, Drayna P, Koozekanani DD (2017) Automated intra-retinal, sub-retinal and sub-rpe cyst regions segmentation in age-related macular degeneration (amd) subjects. Investig Ophthalmol Vis Sci 58(8):397
16. Parhi KK, Rashno A, Nazari B, Sadri S, Rabbani H, Drayna P, Koozekanani DD (2017) Automated fluid/cyst segmentation: a quantitative assessment of diabetic macular edema. Investig Ophthalmol Vis Sci 58(8):4633
17. Guo Y, Ashour AS, Sun B (2017) A novel glomerular basement membrane segmentation using neutrsophic set and shearlet transform on microscopic images. Health Inf Sci Syst 5(1):15
18. Guo Y, Budak Ü, Şengür A, Smarandache F (2017) A retinal vessel detection approach based on shearlet transform and indeterminacy filtering on fundus images. Symmetry 9(10):235
19. Kohler J, Rashno A, Parhi KK, Drayna P, Radwan S, Koozekanani DD (2017) Correlation between initial vision and vision improvement with automatically calculated retinal cyst volume in treated dme after resolution. Investig Ophthalmol Vis Sci 58(8):953
20. Salafian B, Kafieh R, Rashno A, Pourazizi M, Sadri S (2018) Automatic segmentation of choroid layer in edi oct images using graph theory in neutrosophic space. arXiv preprint arXiv:1812.01989

21. Rashno E, Rashno A, Fadaei S (2019) Fluid segmentation in neutrosophic domain. In: 2019 5th Iranian conference on signal processing and intelligent systems (ICSPIS). IEEE, pp 1–5
22. Siri SK, Latte MV (2017) Combined endeavor of neutrosophic set and chan-vese model to extract accurate liver image from ct scan. Comput Methods Programs Biomed 151:101–109
23. Siri SK, Latte MV (2019) A novel approach to extract exact liver image boundary from abdominal ct scan using neutrosophic set and fast marching method. J Intell Syst 28:517–532
24. Lotfollahi M, Gity M, Ye JY, Far AM (2018) Segmentation of breast ultrasound images based on active contours using neutrosophic theory. J Med Ultrason 45(2):205–212
25. Akbulut Y, Sengur A, Guo Y, Smarandache F (2017) NS-k-NN: neutrosophic set-based k-nearest neighbors classifier. Symmetry 9:179–191
26. Dhar S, Kundu MK (2017) Accurate segmentation of complex document image using digital shearlet transform with neutrosophic set as uncertainty handling tool. Appl Soft Comput 61:412–426
27. Rashno E, Norouzi SS, Minaei-bidgoli B, Guo Y (2019) Certainty of outlier and boundary points processing in data mining. In: 2019 27th Iranian conference on electrical engineering (ICEE). IEEE, pp 1929–1934
28. Guo Y, Sengur A (2015) Ncm: neutrosophic c-means clustering algorithm. Pattern Recogn 48(8):2710–2724
29. Akbulut Y, Şengür A, Guo Y, Polat K (2017) Kncm: kernel neutrosophic c-means clustering. Appl Soft Comput 52:714–724
30. Rashno E, Minaei-Bidgolia B, Guo Y (2020) An effective clustering method based on data indeterminacy in neutrosophic set domain. arXiv preprint arXiv:1812.11034
31. Rashno A, Sadri S (2017) Content-based image retrieval with color and texture features in neutrosophic domain. In: 2017 3rd international conference on pattern recognition and image analysis (IPRIA). IEEE, pp 50–55
32. Rashno A, Smarandache F, Sadri S (2017) Refined neutrosophic sets in content-based image retrieval application. In: 2017 10th Iranian conference on machine vision and image processing (MVIP). IEEE, pp 197–202
33. Rahmati M, Rashno A (2021) Myoview: fully-automated image segmentation method to analyse skeletal muscle cross section in exercise-induced regenerating myofibers. bioRxiv
34. Rahmati M, Rashno A (2021) Automated image segmentation method to analyse skeletal muscle cross section in exercise-induced regenerating myofibers. Sci Rep 11:1–16
35. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117
36. Li Z, Liu F, Yang W, Peng S, Zhou J (2021) A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst
37. Zhang W, Doi K, Giger ML, Nishikawa RM, Schmidt RA (1996) An improved shift-invariant artificial neural network for computerized detection of clustered microcalcifications in digital mammograms. Med Phys 23(4):595–601
38. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp 1097–1105
39. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
40. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp 1–9
41. Bogunović H, Venhuizen F, Klimscha S, Apostolopoulos S, Bab-Hadiashar A, Bagci U, Beg MF, Bekalo L, Chen Q, Ciller C et al (2019) Retouch: the retinal oct fluid detection and segmentation benchmark and challenge. IEEE Trans Med Imaging 38(8):1858–1874
42. Rashno A, Koozekanani DD, Parhi KK (2018) Oct fluid segmentation using graph shortest path and convolutional neural network. In: 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC). IEEE, pp 3426–3429
43. Azimi B, Rashno A, Fadaei S (2020) Fully convolutional networks for fluid segmentation in retina images. In: 2020 international conference on machine vision and image processing (MVIP). IEEE, pp 1–7
44. Narin A, Kaya C, Pamuk Z (2021) Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. Pattern Anal Appl 1–14
45. Kattenborn T, Leitloff J, Schiefer F, Hinz S (2021) Review on convolutional neural networks (cnn) in vegetation remote sensing. ISPRS J Photogramm Remote Sens 173:24–49
46. Heskes T (1997) Practical confidence and prediction intervals. In: Advances in neural information processing systems. pp 176–182
47. Papadopoulos G, Edwards PJ, Murray AF (2001) Confidence estimation methods for neural networks: a practical comparison. IEEE Trans Neural Netw 12(6):1278–1287
48. Dybowski R, Roberts SJ (2001) Confidence intervals and prediction intervals for feed-forward neural networks. Clin Appl Artif Neural Netw 298–326
49. Mazloumi E, Rose G, Currie G, Moridpour S (2011) Prediction intervals to account for uncertainties in neural network predictions: methodology and application in bus travel time prediction. Eng Appl Artif Intell 24(3):534–542
50. Blake A, Curwen R, Zisserman A (1993) A framework for spatiotemporal control in the tracking of visual contours. Int J Comput Vis 11(2):127–145
51. He X, Zemel RS, Carreira-Perpiñán MÁ (2004) Multiscale conditional random fields for image labeling. In: Proceedings of the 2004 IEEE computer society conference on computer vision and pattern recognition. CVPR 2004, vol 2. IEEE, pp II–II
52. Kendall A, Gal Y (2017) What uncertainties do we need in bayesian deep learning for computer vision?. In: Advances in neural information processing systems. pp 5574–5584
53. Gal Y (2016) Uncertainty in deep learning. Ph.D. thesis, PhD thesis, University of Cambridge
54. Gal Y, Hron J, Kendall A (2017) Concrete dropout. In: Advances in neural information processing systems. pp 3581–3590
55. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424
56. Chen T, Goodfellow I, Shlens J (2015) Net2net: accelerating learning via knowledge transfer. arXiv preprint arXiv:1511.05641
57. Wei T, Wang C, Rui Y, Chen CW (2016) Network morphism. In: International conference on machine learning. pp 564–572
58. Gal Y, Islam R, Ghahramani Z (2017) Deep bayesian active learning with image data. In: Proceedings of the 34th international conference on machine learning, vol 70. JMLR. org, pp 1183–1192
59. Guo Y, Ashour AS (2019) Neutrosophic multiple deep convolutional neural network for skin dermoscopic image classification. Neutrosophic set in medical image analysis. Elsevier, Amsterdam, pp 269–285
60. Özyurt F, Sert E, Avci E, Dogantekin E (2019) Brain tumor detection based on convolutional neural network with neutrosophic expert maximum fuzzy sure entropy. Measurement 147:106830
61. Khalifa NEM, Smarandache F, Manogaran G, Loey M (2021) A study of the neutrosophic set significance on deep transfer

learning models: an experimental case on a limited covid-19 chest x-ray dataset. Cogn Comput 1–10
62. Jiang X, Guo Y, Chen H, Zhang Y, Lu Y (2019) An adaptive region growing based on neutrosophic set in ultrasound domain for image segmentation. IEEE Access 7:60584–60593
63. Rashno E, Akbari A, Nasersharif B (2019) A convolutional neural network model based on neutrosophy for noisy speech recognition. arXiv preprint arXiv:1901.10629
64. Deng L (2012) The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Mag 29(6):141–142
65. Krizhevsky A, Nair V, Hinton G. (2010) The cifar-10 dataset, online: http://www.cs.toronto.edu/kriz/cifar.html55
66. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images. Tech. rep, Citeseer