TRUSTWORTHINESS EVALUATION AND TRUST-AWARE DESIGN OF CNN ARCHITECTURES

Mingxi Cheng, Tingyang Sun, Shahin Nazarian, and Paul Bogdan
Department of Electrical and Computer Engineering
University of Southern California
Los Angeles, CA, 90089, USA
{mingxic, suntingy, shahin.nazarian, pbogdan}@usc.edu

ABSTRACT

Convolutional neural networks (CNNs) are known to be effective tools in many deep learning application areas. Despite CNN's good performance in terms of classical evaluation metrics such as accuracy and loss, quantifying and ensuring a high degree of trustworthiness of such models remains an unsolved problem raising questions in applications where trust is an important factor. In this work, we propose a framework to evaluate the trustworthiness of CNNs. Towards this end, we develop a trust-based pooling layer for CNNs to achieve higher accuracy and trustworthiness in applications with noise in input features. We further propose TrustCNets consisting of trustworthiness-aware CNN building blocks, i.e., one or more *conv* layers followed by a trust-based pooling layer. TrustCNets can stack together as a trust-aware CNN architecture or be plugged into deep learning architectures to improve performance. In our experiments, we evaluate the trust of popular CNN building blocks and demonstrate the performance of TrustCNet empirically with multiple datasets.

1 INTRODUCTION

Within the recent rapid developments of deep learning (DL), the convolutional neural networks (CNNs) emerged as a critical component in various computer vision, machine learning, and cyber-physical systems tasks including the novel view synthesis Zhou et al. (2016), SARS-CoV-2 vaccine design Yang et al. (2021), SARS-CoV-2 misinformation combating Cheng et al. (2021a), or perception and decision making modules of autonomous systems Gkioxari et al. (2019); Zhu et al. (2015); Cheng et al. (2021c); Luo et al. (2018); Phillips et al. (2021). Although numerous evaluation metrics for the analysis of neural networks (e.g., accuracy, precision-recall, area under the curve) exist, they fail to provide a measure of trust / trustworthiness when quality of the dataset and the degree of noise are unknown. Consequently, to overcome the challenges related to unknown corrupted or imprecisely curated training datasets as well as to account for the intrinsic uncertainty of dynamic environments not captured in training datasets, the trustworthiness and fairness become critical evaluation metrics for characterizing the performance of DL models Jiang et al. (2018); Eshete (2021); Du et al. (2020); Mehrabi et al. (2021).

Fairness-, robustness-, and trustworthiness-related research has received an increasing amount of attention in recent years. Such efforts include fairness-aware dataset and model development Hazirbas et al. (2021); Mehrabi et al. (2021); Bobadilla et al. (2020), robust model and training methods developed for achieving better generalization Madry et al. (2017); Cohen et al. (2019); Salman et al. (2019); Yang et al. (2020), as well as trust-aware model design in various application areas Deng et al. (2017); Bernabe et al. (2016); Cheng et al. (2021b). A robust model should be able to generalize well to adversarial and out-of-distribution (OOD) samples Uwimana & Senanayake (2021), especially for real-world and open-world applications. Fairness in DL has also been linked to OOD literature Creager et al. (2021) and classical approaches from OOD field are applicable to fairness issues. According to Shen et al. (2021), if the invariance assumption adopted for OOD could be seen as a fairness notion, then pursuing OOD could be considered as pursuing fairness. Trustworthiness concept can also be linked to OOD issues. For example, OOD generalization can be realized through domain generalization using feature normalization methods to constrain the domain discrepancy Pan et al. (2018). One can adopt trustworthiness measures to make a trust-aware feature normalization to capture and eliminate domain variance and distill trustworthy task-related features. Note that the evaluation of the OOD generalization performance remains challenging. It is shown empirically that the test accuracy evaluation metric in a single environment would mislead the judgment Ye et al. (2021), hence, we argue that trustworthiness as another evaluation metric could be useful in such scenarios.

Trustworthiness quantification in DL models has emphasized the limitations of relying solely on accuracy when dealing with unknown noise sources. For instance, a recent work from Google research proposes to quantify the trust score of a classifier's prediction on a specific test sample by measuring the agreement between the classifier and a modified nearest-neighbor classifiers Jiang et al. (2018). This work does not consider the inner architecture of a classifier nor the trustworthiness of datasets. The DeepTrust Cheng et al. (2020) framework provides analytical strategies for quantifying the trustworthiness of deep neural networks (DNNs) by exploiting a binomial subjective opinion Jøsang (2016). While DeepTrust quantifies the trustworthiness for classical DNNs and is innovative, it is not directly applicable to CNNs and does not utilize trust values in inference.

To overcome these shortcomings, in this work, we develop a trustworthiness evaluation framework for CNN building blocks and evaluate the trustworthiness of several popular CNN architectures. We further propose *trust-aware CNN blocks (TrustCNets)* composed of one or more *conv* layers followed by a trust-based *max-pooling* layer. The trust-based pooling layer works on the evaluated trustworthiness maps, while the traditional pooling layers work on feature maps. The main contributions of this work are as follows:

- We develop an analytical framework for quantifying the trustworthiness of *conv* and *max-pooling* layers in CNNs.
- We evaluate and compare the trustworthiness of popular CNN building blocks and CNN architectures, e.g., VGG16 and AlexNet.
- We propose a *max-trust-pooling layer* which operates based on the trust map instead of the feature map. In addition, we empirically show that with this *max-trust-pooling*, the trained CNNs achieve higher trustworthiness and accuracy in some cases and possess specific noise-tolerant properties.

This paper is organized as follows. Sections 2 and 3 introduce the related work and preliminaries on trust and opinion definition. The proposed trust evaluation framework for CNNs, and trust-aware CNN building blocks (TrustCNets) involving *max-trust-pooling* are introduced in Sec. 4 and Sec. 5. We evaluate the trust of popular DL model building blocks in Sec. 6, and then study the impact of *max-trust-pooling* and TrustCNet in Sec. 7. Sec. 8 concludes this work.

2 Related Work

Trust modeling and quantification. To model and quantify trust / trustworthiness in various AI decisions, early efforts led to the Dempster-Shafer theory (DST) and more recently to the subjective logic (SL) formalism. In SL, the probability density function is used to quantify a so called opinion Jøsang (2016). In order to form an opinion from a subject, the theory of belief functions, DST, is used for modeling epistemic uncertainty – a mathematical theory of evidence Shafer (1976). However, the corresponding fusion operators in DST are controversial and confusing Smarandache (2004). Therefore, we propose to use the flexible operators in SL for trust and opinion calculations. Note that the core of our work is to quantify trust of CNNs and propose trust-aware CNN building blocks, which is significantly beyond the realm of SL.

Trust in DL. Several evidence theory and trust propagation methods in the online reputation systems and network systems are proposed in the literature Guha et al. (2004); Su et al. (2014); Urena et al. (2019). However, these methods are not applicable to the field of AI and DL. Uncertainty, on the other hand, has been studied in DL research, like uncertainty propagation in deep neural networks in Titensky et al. (2018), piece-wise exponential approximation in Astudillo & Neto (2011), and evidential DL in Sensoy et al. (2018). DeepTrust Cheng et al. (2020), an SL-inspired framework, is proposed to evaluate the trustworthiness of DNNs, however, trust is not embedded in the network architecture or utilized in inference. In contrast, in this work, we propose a trust framework for CNNs and design a new trust-based *max-pooling* function to improve CNNs' performance in some cases. We leverage trust information in the training and inference of CNNs. In addition, one of the biggest limitations of DeepTrust mentioned by the authors is the requirement of data's trustworthiness. However, we don't have this limitation as discussed in Sec. 6.2.

Uncertainty quantification. Uncertainty quantification (UQ) is a well-studied field in deep learning and machine learning. Generally speaking, there are two sources of uncertainty Gawlikowski et al. (2021); Hüllermeier & Waegeman (2021), namely model uncertainty and data uncertainty. Model uncertainty is known as epistemic uncertainty and is reducible with infinite data. Data uncertainty is known as aleatory uncertainty and usually is irreducible, such as label noise or attribute noise. Bayesian approximation such as variational inference Hinton & Van Camp (1993); Barber & Bishop (1998); Gal & Ghahramani (2016), and ensemble learning techniques such as weight sharing Gal et al. (2017) are two popular methods in UQ. In Bayesian methods, model parameters are modeled as random variables, and the parameters are sampled from a distribution in a single forward pass Abdar et al. (2021). In ensemble methods, there are multiple models, and the prediction is the combination of several models' outputs. In contrast, our framework works with one neural network model and collects evidence for each parameter to measure an opinion (which includes belief mass, disbelief mass, and uncertainty mass). However, our uncertainty mass in opinion is a second-order uncertainty, while epistemic and aleatory uncertainty are first-order uncertainties. More specifically, second-order uncertainty represents a probability density function (PDF) over the first-order probabilities (e.g, $p(x)$). According to the Beta binomial model in SL, a binomial opinion is equivalent to a Beta PDF ($Beta(p(x))$) under a specific bijective mapping. The density expresses where the probability $p(x)$ is believed to be along the continuous interval $[0, 1]$. The probability density, is then to be interpreted as second-order probability. This interpretation is the foundation for mapping high probability density in Beta PDFs into high belief mass in opinions, therefore, flat probability density in Beta PDFs is represented as uncertainty in opinions Jøsang (2016). Since this evidence-based uncertainty reflects the vacuity of information, it is reducible as we collect information during training.

The uncertainty mass in opinions is an evidence-based uncertainty and can be characterized by the spread of the Beta or Dirichlet PDFs Shi et al. (2020); Corbière et al. (2021). Perhaps the closest uncertainty notion to our work is grounded in a recently developed family of models: Dirichlet-based uncertainty family (DBU) Kopetzki et al. (2021). Evidential deep learning (EDL) Sensoy et al. (2018) is one of the works in DBU models, proposed to quantify classification uncertainty. Given a standard neural network classifier, the softmax output of the classifier for a single sample is interpreted as a probability assignment over the available classes. The density of each of such probability assignments is represented by a Dirichlet distribution, and it models second-order probabilities and uncertainty. It has been shown that data, model, and distributional uncertainty, can all be quantified from Dirichlet distributions; and these DBU measures show outstanding performance in the detection of OOD samples Malinin & Gales (2018); Sensoy et al. (2018); Corbière et al. (2021). In our experiments, we find that OOD and adversarial samples are less trustworthy compared to clean in-distribution data, and this could be further explored in future works for OOD detection and generalization studies.

Pooling functions. The classical general pooling functions, such as *max-pooling* and average pooling are frequently used in CNN architectures. The forward propagation of *max-pooling* is to pass the largest value in the patch to the next layer, which reduces the computational cost and provides basic translation invariance to the internal representation. Although many novel pooling methods have been proposed along the years, such as global second order pooling Cai et al. (2017); Gao et al. (2016), overlapping-pooling Krizhevsky et al. (2012), spatial pyramid pooling He et al. (2015), background-aware pooling Oh et al. (2021), they still perform the operation of pooling functions based on feature values. In this work, we propose to define a new pooling function based on CNNs' trust values, to achieve higher trustworthiness and accuracy after training.

3 Background

An opinion can be seen as a summary of experience / evidence, e.g., we have opinions about a restaurant based on our past experience or collected evidence, and a good experience leads to a positive opinion. Then, trustworthiness is formed from opinions. In this work, given a DL model, with the values of each neuron, cell, and weight known, we would like to assign a *trustworthiness* value and an *opinion* to these components and build a framework to quantify the trustworthiness of CNNs. The definitions of opinion and trust are described as follows:

Definition 3.1 (Opinion Jøsang (2016)) *Formally, an opinion about the truth or presence of x is defined as a tuple containing 4 components: $\overline{W}_x = [b_x, d_x, u_x, a_x]$, where $b_x + d_x + u_x = 1$, b_x, d_x, u_x, and a_x are belief, disbelief, uncertainty, and base rate, respectively. Each ranges $[0, 1]$.*

Definition 3.2 (Trustworthiness Cheng et al. (2020)) *With opinion, we can further calculate the trustworthiness value of the object x as belief mass plus uncertainty multiplied by the base rate, i.e., trustworthiness $p_x = b_x + u_x * a_x$.*

There are several extreme cases, e.g., if the belief mass takes the maximum value of 1, it represents full belief, for which the trust value reaches the maximum value of 1. Similarly, if the disbelief mass takes the maximum value, it represents full disbelief and the trust value reaches the minimum value of 0. After introducing opinion and trustworthiness, we now describe how to quantify an opinion of an object x from *evidence*. In the previous restaurant example, good experience broadly leads to positive opinion, and this mapping between opinion and evidence is defined as follows:

Definition 3.3 (Evidence Jøsang (2016)) *Given an object x, if its behavior satisfies some pre-defined property, a positive evidence r is quantified, otherwise, a negative evidence s is quantified. With collected evidence about x, we can map the evidences to an opinions using the following equations: $b_x = \frac{r}{r+s+W}$, $d_x = \frac{s}{r+s+W}$, $u_x = \frac{W}{r+s+W}$, where W is a non-informative prior weight, which has a default value of 2 to ensure that the prior Beta PDF is the uniform PDF when $r = s = 0$ and $a_x = 0.5$.*

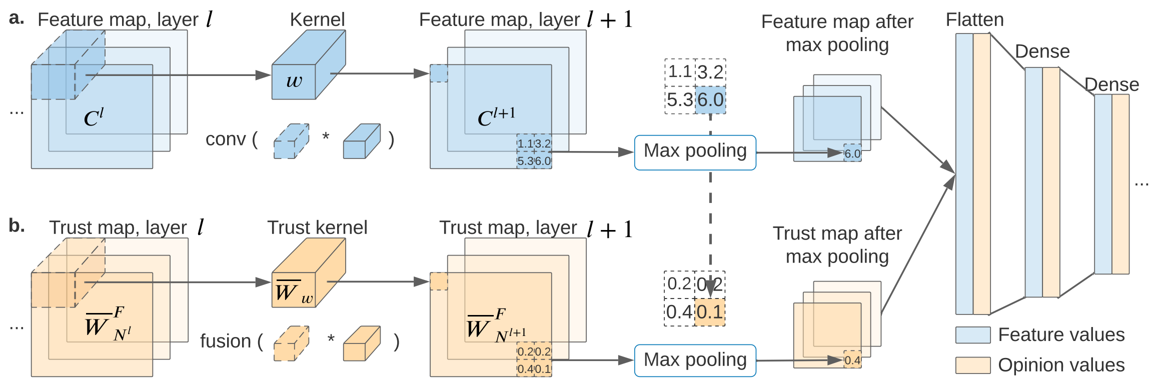

Figure 1: Trustworthiness quantification in *conv* and *max-pooling* layers. Trust calculation and feature calculation are accomplished at the same time. In *conv* layer, the feature calculation includes a convolution calculated as shown in **(a)**, and the trust calculation includes a fusion calculation done in parallel as shown in **(b)**. Note that the feature map in (a) is the same as in classical CNNs, and the trust map in (b) is our contribution. The resulting feature map and trust map have the same shape. A *max-pooling* layer with 2×2 window size then takes the feature map in layer $l+1$ and outputs the maximum feature value (e.g., 6.0) in the window. The corresponding cell in the trust map contains the trust value 0.4, which is the trust value of cell 6.0 in feature map.

4 Trustworthiness Evaluation in CNNs

In this section, we introduce our opinion and trustworthiness quantification framework for CNN building blocks, i.e., *conv* and *pooling* layers. We demonstrate our trustworthiness evaluation with a simple CNN architecture as shown in Fig. 1, which consists of one *conv* layer using same padding method and one 2×2 classical *max-pooling* layer followed by fully-connected layers. We perform trustworthiness evaluation along with neural network training. Fig. 1a shows a normal CNN calculation and Fig. 1b shows the trust calculation. In each layer, there are ordinary feature values saved in cells / neurons and weights, as well as trust values and opinions corresponding to cells / neurons and weights. We denote a neuron or a cell by N, and its feature and opinion / trust values are denoted by C and $\overline{W}_N$, respectively.

4.1 Opinion and Trust Evaluation in CNNs

The trust of an object as defined in Sec. 3 is calculated from an opinion. Hence, the trust evaluation is dependent on opinion quantification. To determine the opinion of a CNN, we first evaluate the opinion of output neurons. On a high level, the opinion of a CNN comes from opinions of the output neurons, i.e., neurons in the output layer. We denote the opinion of a CNN as $\overline{W}_{CNN}$ and the opinion of an output neuron as $\overline{W}_{neuron}$ (or $\overline{W}_N$ in short). With output neurons' opinions known, we calculate the opinion of a CNN by combining or fusing all output opinions together utilizing the average fusion operator[1]:

$$\overline{W}_{CNN} = fusion(\overline{W}_{neuron_1}, ..., \overline{W}_{neuron_n}), \tag{1}$$

where n is the total number of output neurons. Generally speaking, the fusion operators combine the opinions together and take care of the conflict information expressed in these opinions. Out of many fusion operators defined in SL, we use average fusion here to combine the opinions from output neurons as they hold opinions over the same variable / object, a CNN, and there exist dependencies within opinions. In what follows, we will (*i*) review how to evaluate the opinion of a single neuron in a dense layer, then introduce how to evaluate opinion of (*ii*) a *conv* layer, and (*iii*) a *max-pooling* layer.

4.2 Opinion Evaluation of a Single Neuron

Inspired by forward and backward propagation, the authors in Cheng et al. (2020) proposed that the opinion calculation of a single neuron in DNNs is quantified in a procedure called *opinion propagation*. Similarly in CNNs, for an output neuron, the opinion of neuron ($\overline{W}_{neuron}$) combines the forward passing opinion ($\overline{W}^F_N$) and backward updating opinion

[1] Mathematical details of fusion operators can be found in Appendix Sec. A.2.2.

($\overline{W}^B_N$) as defined in Eq. 7. For a hidden neuron, the opinion calculation only contains the forward pass opinion. (More detailed description can be found in Appendix Sec. A.3). In the following sections, we introduce the calculations of $\overline{W}^F_N$, $\overline{W}^B_N$, and opinions of weights $\overline{W}_w$'s (where w is short for $weight$).

4.3 Forward Opinion Propagation

For a neuron N^l in layer l, it takes input from the previous layer, and we denote the opinion of an input from the previous layer as $\overline{W}^F_{N^{l-1}}$. For neurons connected with input layer, $\overline{W}^F_{N^{l-1}}$ is the opinion of input features ($\overline{W}_x$, we will introduce how to initialize it in Sec. 7), and for neurons in other layers, $\overline{W}^F_{N^{l-1}}$ is the calculated opinion of neuron in the previous layer. Note that only the neurons in output layer have backward opinions, and neurons in all hidden layers only have forward opinions.

In this section, we study frequently used layers in CNNs, namely, fully connected layer, *conv* layer, and pooling layer. The opinion evaluation in these architectures is different. We use two examples shown in Fig. 1 and 2 to illustrate the framework.

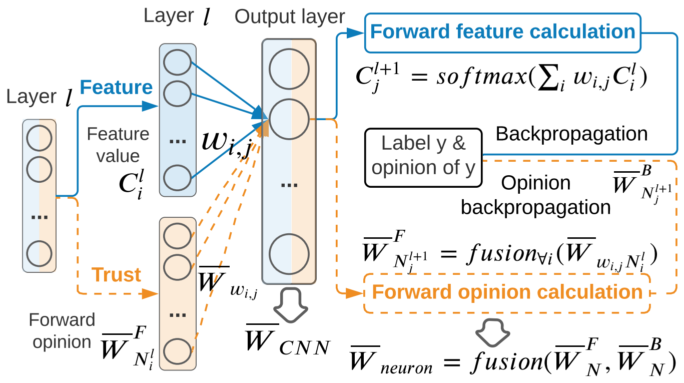
Layer l
Output layer
Layer l
Feature
Feature value C^l_i
$w_{i,j}$
Trust
Forward opinion $\overline{W}^F_{N^l_i}$
$\overline{W}_{w_{i,j}}$
$\overline{W}_{CNN}$
Forward feature calculation
$C^{l+1}_j = softmax(\sum_i w_{i,j}C^l_i)$
Label y & opinion of y
Backpropagation
Opinion backpropagation
$\overline{W}^B_{N^{l+1}_j}$
$\overline{W}^F_{N^{l+1}_j} = fusion_{\forall i}(\overline{W}_{w_{i,j}N^l_i})$
Forward opinion calculation
$\overline{W}_{neuron} = fusion(\overline{W}^F_N, \overline{W}^B_N)$

Figure 2: Opinion calculation in a dense layer. Layer l is a dense layer and each neuron in this layer has a feature value and a corresponding opinion and trust value. Both opinions of neurons and opinions of weights are used to calculate the forward opinion of a neuron in the next layer using Eq. 2. If the next layer is the output layer, backward opinion is also calculated using Eq. 5.

Forward propagation in fully connected layer. Given a neuron i in layer l, we assume its feature value and forward opinion are C^l_i and $\overline{W}^F_{N^l_i}$, respectively, as shown in Fig. 2. To calculate the opinion of a neuron j in the next layer, we follow Cheng et al. (2020) and get inspiration from ordinary forward propagation calculation in dense layers, e.g., $C^{l+1}_j = f(\sum_i w_{i,j}C^l_i)$, where $f(\cdot)$ is an activation function and $w_{i,j}$'s are weights. The opinion calculation of neuron j in layer $l+1$ takes similar form:

$$\overline{W}^F_{N^{l+1}_j} = fusion(\overline{W}_{w^l_{1,j}N^l_1}, ..., \overline{W}_{w^l_{n,j}N^l_n}), \quad (2)$$

where $\overline{W}_{w^l_{i,j}N^l_i} = \overline{W}_{w^l_{i,j}} \cdot \overline{W}^F_{N^l_i}$, $\forall i,\ j$, is an opinion multiplication (for details please see Appendix Sec. A.2.1). Opinion $\overline{W}_{w^l_{i,j}}$ represents the opinion of a weight passing from i^{th} neuron in layer l to the j^{th} neuron in layer $l+1$. To replace the addition operation in $\sum_i w_{i,j}C^l_i$, we use fusion operator $fusion(\cdot)$. For neurons in hidden layers, they only have forward opinion updated along with the training process, and for neurons in the output layer, both forward opinion $\overline{W}^F_N$ and backward opinion $\overline{W}^B_N$ are used to calculate the final opinion of an output neuron $\overline{W}_{neuron}$. We will introduce the backward opinion calculation in Sec. 4.4.

Forward propagation in *conv* layer. Operations in *conv* layer are nothing more but multiplication and addition, therefore, the forward propagation in *conv* layer is very similar to that in fully connected layer. As shown in Fig. 1b, we calculate the opinion of a neuron in *conv* layer by taking inputs from one filter window, e.g., an $m \times m$ filter, and multiplying corresponding weights. Formally, the forward opinion of a neuron / cell j in layer $l+1$ reads:

$$\overline{W}^F_{N^{l+1}_j} = fusion(\overline{W}_{w^j_1 N^l_1}, ..., \overline{W}_{w^j_{m\times m} N^l_{m\times m}}). \quad (3)$$

Forward propagation in *max-pooling* layer. The classical pooling layers are sometimes sandwiched between *conv* layers to compress the amount of data and parameters. Its operations are similar as that of the *conv* layers, except that the core of the pooling function is to take the maximum value or the average value of the corresponding position for maximum or average pooling, respectively. *Max-pooling* outputs the maximum value in the pooling window (e.g., a 2×2 window) and its corresponding opinion reads:

$$\overline{W}^F_{N^{l+1}_i} = \max_{feature}(\overline{W}^F_{N^l_{i,1}}, \overline{W}^F_{N^l_{i,2}}, \overline{W}^F_{N^l_{i,3}}, \overline{W}^F_{N^l_{i,4}}). \quad (4)$$

Fig. 1 shows that the *max-pooling* takes input opinions ($\overline{W}^F_{N^{l+1}_{i,1}}, \overline{W}^F_{N^{l+1}_{i,2}}, \overline{W}^F_{N^{l+1}_{i,3}}, \overline{W}^F_{N^{l+1}_{i,4}}$) from a 2×2 pooling window in layer $l+1$, and outputs $\overline{W}^F_{N^{l+2}_i}$, the opinion of the i^{th} cell in the layer $l+2$.

Latency analysis. The latency of the proposed trust network come from feature and trust calculation as shown in Fig. 1. The trust calculation latency takes similar form as convolution operation. For an image with $M \times N$ size and a filter with $k \times k$ size, the number of operations of convolution is $\mathcal{O}(MNkk)$. In our trust calculation, if the trust calculation in each computation operation is $\mathcal{O}(c)$, then the latency of trust calculation is $\mathcal{O}(MNkkc)$. Therefore, the latency of trust network in theory is $\mathcal{O}(MNkkc)$. Further acceleration techniques are available but not the focus of this work.

4.4 Backward opinion propagation

In classical neural network training process, model parameters are updated in back propagation. Inspired by this, we update the parameter opinion $\overline{W}_w$ and backward neuron opinion $\overline{W}_N^B$ in the backward opinion propagation phase.

Update parameter opinion. Weights and bias (parameters) are attached with opinions and their opinions and trust values are updated during training. In the beginning, parameters are initialized to have maximum uncertainty opinions (i.e., uncertainty mass equals to 1) due to lack of evidence, and later on they are updated based on partial derivative of the cost function $\frac{\mathrm{d}J}{\mathrm{d}w}$. We track the gradient of a parameter, and if a gradient is in a tolerance range, i.e., $\left|\frac{\mathrm{d}J}{\mathrm{d}w}\right| \leq \tau$ (where τ is the mean of all gradients), we count it as a positive evidence r to formulate the opinion of $\overline{W}_w$ based on evidence theory (Def. 3.3). Otherwise, it contributes to a negative evidence s. In simpler terms, if weights are updated by a larger step, it contributes to lower belief and trustworthiness. Intuitively, r and s represent the magnitude of parameters change between the current value and the optimal value. Positive evidences indicate that current parameter values are getting close to the optimal parameters.

Update output neuron opinion. Besides the parameter opinions, the backward opinion $\overline{W}_N^B$ of a neuron in an output layer needs to be updated during the training process. It is formulated based on the absolute error $|y' - y|$ between neuron' output y' (i.e., confidence value after *softmax* function) and the one-hot encoded ground truth label y. The evidence for conditional backward opinion formulation of a neuron ($\overline{W}_{y'|y}$) on each update is determined as follows: a positive evidence r is counted when $|y' - y| < \tau$, and a negative evidence s is counted when $|y' - y| \geq \tau$. After calculating the $\overline{W}_{y'|y}$, we update $\overline{W}_N^B$ as follows:

$$\overline{W}_N^B = \overline{W}_{y'} = \overline{W}_{y'|y} \cdot \overline{W}_y, \tag{5}$$

where $\overline{W}_y$ is the opinion of true label y and it is initialized based on the quality of dataset, e.g., for dataset without label noise, $\overline{W}_y$ could have the maximum belief mass. With backward opinion updated, we update output neurons' opinions following Eq. 7 as demonstrated in Fig. 2.

5 TrustCNet Framework

In this section, we first propose a novel trustworthiness-based *max-pooling* (*max-trust-pooling*) layer. Then, we take a CNN block and substitute the classic *max-pooling* layer with our *max-trust-pooling* layer to build our TrustCNet block, a novel trust-aware CNN block.

5.1 Max-trust-pooling

Pooling layers usually follow *conv* layers and take the average or maximum value in a window based on feature values. With opinion and trust evaluation, a *conv* layer has a trust map in addition to the feature map as shown in Fig. 3. Besides the feature-based pooling functions, trust-based pooling comes naturally. Therefore, we propose *max-trust-pooling*, a new pooling layer function based on trustworthiness values instead of feature values. Classical *max-pooling* selects maximum values from filters based on input values to extract most important features. We argue that with trustworthiness considerations, features can be selected according to their trust values to improve both trustworthiness of the CNN and accuracy under certain circumstances. We can interpret this as choosing the most trusted items in every window.

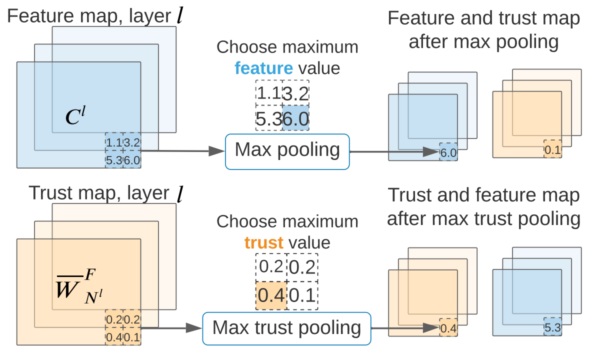

Figure 3: *Max-trust-pooling* layer. Different than *max-pooling* layer that operates on feature maps, a *max-trust-pooling* layer generates output based on trust map. In this example, with 2×2 window size, a *max-pooling* function will output the maximum value in feature window, which is 6.0 with the trust value of 0.1. A *max-trust-pooling* function in this case will output feature value 5.3 because it has the maximum trust value of 0.4. This demonstrates the difference between *max-pooling* and *max-trust-pooling*.

A comparison between *max-pooling* and *max-trust-pooling* is shown in Fig. 3. Given a feature map C^l and a trust map $\overline{W}_{N^l}$ in layer l, a 2×2 *max-pooling* function selects the maximum feature value, and the resulting cell in the next layer contains a feature of value 6.0 and its corresponding trust value is 0.1. If we replace the *max-pooling* by our proposed *max-trust-pooling*, the pooling function then looks into the trust map (instead of feature map), and the resulting cell in the next layer contains a trust value of 0.4 (and its corresponding feature value is 5.3). The pseudo code of *max-trust-pooling* is shown in Alg. 1.

As implied by Alg. 1, *max-trust-pooling* does not require much more storage space, and all the computation can be realized based on trust framework. We only need to replace the basic opinion propagation method in pooling layer with:

$$\overline{W}^F_{N_i^{l+1}} = \max_{trust}(\overline{W}^F_{N^l_{i,1}}, \overline{W}^F_{N^l_{i,2}}, \overline{W}^F_{N^l_{i,3}}, \overline{W}^F_{N^l_{i,4}}). \quad (6)$$

Algorithm 1 Max-trust-pooling function

```
Require:
  Feature map in layer $l$, $C^l$, trust map in layer $l$, $\overline{W}_{N^l}$, window size, $m \times m$, and pooling stride, $s$.
Ensure:
  Feature map and trust map in next layer, $C^{l+1}$ and $\overline{W}_{N^{l+1}}$, shape = $(H, W)$.
  for $i = 1 : H$ do
    for $j = 1 : W$ do
      Calculate sliding position:
      $vertical\_start = i * s$
      $vertical\_end = vertical\_start + m$
      $horizontal\_start = m * s$
      $horizontal\_end = horizon\_start + m$
      Slide two windows of size $m \times m$ in both feature map $C^l$ and trust map $\overline{W}_{N^l}$ based on the calculated position.
      Select the maximum value in trustworthiness window and cache the index $(id_i, id_j)$.
      $C^{l+1}[i][j] = C^l[id_i][id_j]$
      $\overline{W}_{N^{l+1}}[i][j] = \overline{W}_{N^l}[id_i][id_j]$
    end for
  end for
  return $C^{l+1}$, $\overline{W}_{N^{l+1}}$
```

5.2 TrustCNet Block

Our TrustCNet-n block is built based on a plain CNN network, e.g., n *conv* layers followed by a *max-pooling* layer, and we replace the *max-pooling* with our proposed *max-trust-pooling*. As shown in Fig. 4, *conv* layers have feature maps and trust maps, then a *max-trust-pooling* layer takes the trust map and generates output. In the process of training, the parameters of the model and the corresponding opinions and trust maps should be updated synchronously. As an architecture that operates based on trust values, TrustCNet-n is a basic building block that can be stacked together to form a trust-aware CNN, i.e., TrustCNet. By collecting and amplifying more trustworthy feature values in layers, the trust-aware model learns different information and features than the plain deep learning models.

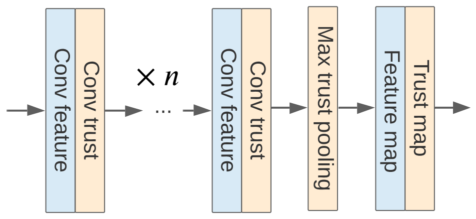

Figure 4: TrustCNet-n: a block with n *conv* layers followed by one *max-trust-pooling* layer.

6 Trust Quantification of CNNs and Data

Deep learning model architecture design is an active research area and researchers develop various CNN models throughout the years to achieve higher performance in multiple application areas. Many successful models become benchmarks such as AlexNet Krizhevsky et al. (2012), VGG Simonyan & Zisserman (2014), ResNet He et al. (2016), Inception Szegedy et al. (2016), EfficientNet Tan & Le (2019), etc. Various evaluation metrics have been proposed and used in evaluating the deep learning models. However, trustworthiness evaluation is still lacking. Therefore, in this section, we use the framework we proposed in Sec. 4 to quantify trustworthiness of several frequently used CNN building blocks. For example, n *conv* layers followed by a *max-pooling* layer, where $n \in \{1, 2, 3\}$, and we denote these CNN blocks as *conv*$\times$*n-max-pool*. We also evaluate trustworthiness of a CNN residual block. In addition, we provide trustworthiness evaluation of deeper CNNs such as VGG16 and AlexNet. Furthermore, we compare with the state-of-the-art trust quantification framework DeepTrust Cheng et al. (2020).

Besides model design, data quality also influences the final results of experiments. While prior research efforts have proposed multiple evaluation metrics for data quality, in this work, we would like to quantify trustworthiness of data and provide a demonstration of how to quantify trustworthiness of data using the framework we proposed in Sec. 4.4. Although data trustworthiness quantification is not the focus of this work, we would like to show that our framework has the flexibility to be modified and utilized in other application areas, such as future online healthcare and legal service, cyber physics systems involving human agents, and hope to provide inspirations to many other directions.

6.1 Quantify Trust of CNNs

Experiment setup. In this experiment, we train and test all six CNNs with the same procedure using a subset of ImageNet dataset Russakovsky et al. (2015); van den Oord et al. (2016). Training / test split is $80\%/20\%$. We preprocess the input by scaling the feature values in range $[0, 1]$ and resizing the image shape to $(180, 180, 3)$. For the CNN building blocks, Adam optimizer is used with 0.0005 learning rate and a sparse categorical cross entropy loss is used. ReLU activation function Fukushima (1969) is used in all *conv* layers. We develop CNNs containing n *conv* layers followed by a *max-pooling* layer and a fully connected output layer. For deeper CNNs, we experiment with VGG16 and AlexNet. The architecture details of CNN blocks are as follows:

- **1 *conv* layer followed by 1 *max-pooling* ($n = 1$):** *conv* (kernel window 3×3, stride 1, same padding, filter number 32) - *max-pooling* (kernel window 2×2, stride 2) - fully connected. This block type has been seen in AlexNet, ResNet, Inception, EfficientNet, etc.
- **2 *conv* layers followed by 1 *max-pooling* ($n = 2$):** *conv* (kernel window 3×3, stride 1, same padding, filter number 32) - *conv* (kernel window 3×3, stride 1, same padding, filter number 64) - *max-pooling* (kernel window 2×2, stride 2) - fully connected. This block type has been used in VGG, Inception, EfficientNet, etc.
- **3 *conv* layers followed by 1 *max-pooling* ($n = 3$):** *conv* (kernel window 3×3, stride 1, same padding, filter number 32) - *conv* (kernel window 3×3, stride 1, same padding, filter number 64) - *conv* (kernel window 3×3, stride 1, same padding, filter number 64) - *max-pooling* (kernel window 2×2, stride 2) - fully connected. This block type has been used in AlexNet, VGG, etc.
- **Residual block:** we use the first residual block architecture from ResNet He et al. (2016).

Experimental results.

Results on CNN blocks. Fig. 5 shows the training accuracy and trustworthiness results of four CNN building blocks we considered in this section. During the training process, training accuracy improves and trustworthiness also increases. The final test results of this experiment are listed in Tab. 1. After inspecting the results, we find that residual block's performance is not ideal, and conv×2-max-pool evolves to be a clear winner as it achieves the highest accuracy and trustworthiness. These results could be useful in future deep learning model design works.

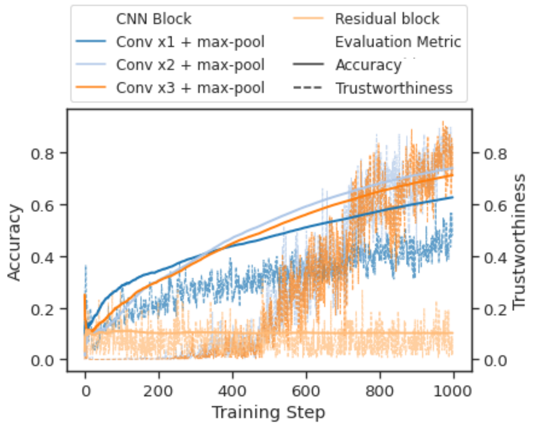

Figure 5: Accuracy and trustworthiness evaluation of CNN blocks. Conv×2 - max-pool architecture is the best among the four blocks tested as it achieves the highest trustworthiness and accuracy.

Results on deeper CNNs. The results of test accuracy and trustworthiness are shown in Fig. 6a. (Training results can be found in Appendix Sec. A.4.1.) We observe that VGG16 although results in lower accuracy than AlexNet, it achieves higher trustworthiness. This result shows that accuracy and trustworthiness are not necessarily positively related.

Comparison with DeepTrust. We further quantify the trustworthiness of VGG16 using modified DeepTrust and our framework. The comparison of trustworthiness results is shown in Fig. 6b. We find that DeepTrust generates much higher trustworthiness values than our method. The difference is caused by the difference in parameter opinion evaluation. Our evaluation considers difference of weight in training while DeepTrust is more output-dominated as its trust of weight is set to be the same as the backward opinion of output.

Table 1: Test accuracy and trustworthiness results of CNN blocks.

	Accuracy	Trustworthiness
conv×1 - max-pool	0.5342	0.5167
conv×2 - max-pool	**0.5522**	**0.5506**
conv×3 - max-pool	0.5248	0.5174
Residual block	0.1178	0.0694

Besides the empirical difference, our framework differs from DeepTrust in many aspects. First of all, DeepTrust is designed specifically for neural networks with only dense layers, and it is not directly applicable to CNNs. It does not utilize trust values in inference or training and only quantifies trust values and does not use the trust values to improve performance. Our framework can quantify trust of networks and also provide a way to utilize trust values for improvement. Secondly, DeepTrust requires opinion or trust of dataset as a priori. Our framework does not have this restriction and further provide a way to quantify trust of data as discussed in Sec. 6.2. Furthermore, we calculate the parameter opinion differently than DeepTrust. In DeepTrust, the opinion of a parameter is the same as the backward opinion of output, which means all weights and bias receive the same opinion and trustworthiness. To address this, our framework calculates opinion of weight based on their gradients during training, which assigns different trust value to different weight. This allows our framework to consider differences in parameters.

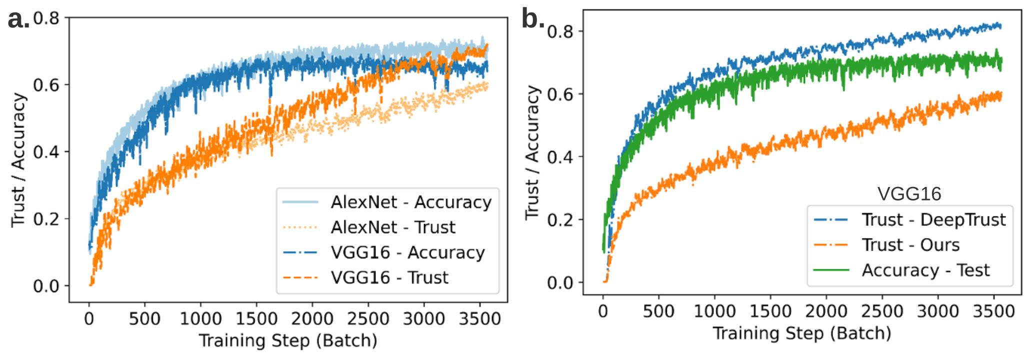

Figure 6: a. Accuracy and trustworthiness evaluation of VGG16 and AlexNet. b. Comparison between DeepTrust and our framework. Results are evaluated on VGG16.

6.2 Quantify Trust of Data

It has been shown that gradients and saliency maps are useful in providing explainability, distinguishing difficult samples, such as mislabeled, noisy, output of distribution samples, in computer vision tasks. As shown in Sec. 4.4, our trust quantification framework works on gradients. Therefore, we propose to evaluate opinion / trust of data as this information is often missing from the dataset. Our data trustworthiness quantification process is as follows. Given a CNN, we retrieve its saliency maps, then for each sample, its trustworthiness is calculated based on our evidence framework in Sec. 4.4, i.e., given a threshold, gradients larger / smaller than the threshold contributes to the negative / positive evidence. Once evidence is quantified, opinion and trustworthiness of data are evaluated accordingly.

Experiment setup. In this experiment, we train a CNN with 18M parameters on CIFAR10 dataset. We consider clean, mislabeled, out-of-distribution (OOD), and adversarial data. In all experiments, we train a CNN with clean data and make modification to test set and quantify trustworthiness of test data. We want to mimic the scenario where we have a pre-trained model deployed in non-ideal environment. For example, the test data fed to the model could include noise or even adversarial attacks. For adversarial samples we consider the following attacks: FGSM Goodfellow et al. (2014), Random FGSM (RFGSM) Tramèr et al. (2017), One Step Least Likely (Step l.l.) Kurakin et al. (2016), Basic Iterative Method (BIM) Kurakin et al. (2018), and Iterative Least Likely (ILL) Kurakin et al. (2018). For OOD data, we follow Liang et al. (2017); Agarwal et al. (2020) and consider Gaussian noise with $\mathcal{N}(0, 1)$. For mislabeled data, we follow Pleiss et al. (2020) and reassign labels from c to $c + 1$. The trustworthiness of data is evaluated per sample and we report the averaged trust values.

Experimental results and discussion. Evaluated mean trust of the clean data and contaminated data are shown in Tab. 2. From the results, we find it is clear that the clean data has the highest trust value, while contaminated data all result in relatively lower trust values. Note that we do not claim that our data trustworthiness quantification can identify data attacks, but we do want to show the results and hope to inspire more followup works on data attack detection, OOD detection, and OOD generalization using trust values.

Table 2: Averaged trustworthiness evaluation results of clean and contaminated data.

Data	Clean	FGSM	RFGSM	Step l.l.	BIM	ILL	OOD	Mislabeled
Trust	0.66	0.49	0.50	0.50	0.48	0.51	0.61	0.54

7 Evaluation of TrustCNet

In the previous section, we showed that our trust framework generated higher values for clean data and much lower values for damaged or noisy data. In applications where input data are damaged or noisy, deep learning models could generate sub-optimal results. Trust-aware CNNs that operate on both feature maps and trust maps might be able to dampen the effect of the noise infused in data, and therefore output better results compared to their non-trust-aware versions. Note that we do not focus on adversarial defense but only provide a study involving noisy data. In this section, we develop a CNN architecture, and then utilize TrustCNet blocks and construct a trust-aware version to compare them in terms of both accuracy and trustworthiness. For noise in dataset, we study three difficulty levels: given input data, (i) we know both the position and intensity of the noise, (ii) we know the position but not intensity of the injected noise, and (iii) both position and intensity are unknown. In addition, we further test our TrustCNet with three different noises under the highest difficulty level.

7.1 How Much Do We Know?

Noisy data is a frequent issue seen in machine learning and DL. Noise can affect either the labels, features, or both. Here, we consider feature noise, e.g., noise in input images in computer vision tasks, and the difficulty level of the problem depends on how much noise information (or prior knowledge of the noise) we know in the test phase. We assume that in training phase, the training data may or may not contain noise and we don't know this information. Therefore, in training phase, opinions of input feature and input label are $\overline{W}_x = [f, 0, 1-f, 0.5]$ and $\overline{W}_y = [1, 0, 0, 0.5]$, respectively, where f is normalized feature value. Then, in test phase, we consider three difficulty levels as follows.

I. Known position and known intensity. To calculate the opinion of CNNs, we propose the trust quantification framework in Sec. 4.1, and it needs opinions' of input features ($\overline{W}_x$) for initialization and updates opinions in propagation. If we know the location and intensity of noise, in the phase of opinion initialization, we set the opinion tuple of the input features without noise as $\overline{W}_x = [f, 0, 1-f, 0.5]$, where f is the normalized feature value, and the opinion of features with noise reads $\overline{W}_x = [f*(1-t), 1-f*(1-t), 0, 0.5]$, where t is the intensity of noise. Intuitively, features without noise obtain a belief value based on the feature value, and features with noise obtain a belief value dampened by the noise level, and they also obtain a disbelief value to fulfill the opinion requirement $belief + disbelief + uncertainty = 1$.

II. Known position and unknown intensity. When working with noisy input, case I is nearly perfect as we know a lot of useful information about noise. However, sometimes we only know the location information of noise. This key information is also proved to be functional. The opinion initialization of input in this case is as follows. Given a input, we set the opinion of all features with noise to $\overline{W}_x = [f, 1-f, 0, 0.5]$, (where f is feature value) and the opinion of all features without noise is $\overline{W}_x = [f, 0, 1-f, 0.5]$. This setup intuitively represents that, on the premise of knowing the location of noisy features, we disbelieve those positions containing noise and hence a low degree of trust on them is propagated during the convolutional trustworthiness computation.

III. Unknown position and unknown intensity. Another common situation when dealing with noise is that we lack information about noise type. In such situation, we set the opinion of all input features as $\overline{W}_x = [f, 0, 1-f, 0.5]$. By assigning the rest part of mass to the uncertainty mass, the TrustCNet learns to update parameters in the training processing in order to obtain the maximum trustworthiness.

7.2 Experiment Setup

The TrustCNet model we develop and test in this section contains a TrustCNet-1 block (i.e., 1 *conv* layer followed by 1 *max-trust-pooling* layer), a 3×3 *conv* layer followed by a 2×2 *max-pooling* layer, and fully connected layer. The *conv* layer in TrustCNet-1 has 8 filters with 3×3 window size, stride 2 using same padding and the *max-trust-pooling*'s window size is 2×2 with stride 2. The non-trust-aware version of this TrustCNet has the same architecture but with the *max-trust-pooling* layer replaced by a normal *max-pooling* layer. We use MNIST LeCun et al. (1998) and CIFAR-10 Krizhevsky et al. (2009) datasets and insert a self-defined Gaussian-distributed noise to input features following Zhang et al. (2017) in training and testing. We randomly add Gaussian noise to a certain number of pixels in input images. Adam optimizer is used with 0.001 learning rate and sparse categorical cross entropy loss is used to update the parameters during batch training process. ReLU activation is used in all *conv* layers. We use accuracy (and accuracy improvement), trust (and trust improvement) as evaluation metrics to compare TrustCNet and its non-trust-aware version. Improvement is calculated as the increase divided by the original value. We conduct the experiments 5 times and report mean and standard deviation results. We also provide comparison results between TrustCNets and their non-trust-aware variants under 4 different types of noise in Appendix Sec. A.4.2.

7.3 Experimental Results

The comparison results of TrustCNet and its non-trust-aware version are shown in Tab. 3. Non-trust-aware CNNs achieve similar results in terms of both accuracy and trustworthiness when facing with noisy input, the slight differences are contributed by random initialization. While with clean data, TrustCNets and its non-trust-aware versions perform similarly, TrustCNets perform much better than the non-trust-aware ones when there is noise involved in data, and different prior noise information leads to different results.

Case I. With known noise position and intensity, TrustCNets achieve much better accuracy and trustworthiness results compared to their non-trust-aware versions. *Max-trust-pooling*-based TrustCNets operate on both feature map and trust map, hence achieve better results in terms of trustworthiness. The feature value-based input opinion initialization contributes to the outperformance of accuracy and shows that TrustCNets have noise-tolerant ability.

Table 3: Comparison between TrustCNets and their non-trust-aware variants under noisy datasets. In case I, position and intensity information of noisy input are used in input opinion initialization. In case II, only position information is used, and in case III, no noise information is used in opinion initialization.

		Max-pooling		Max-trust-pooling (TrustCNet)			
Dataset	Level	Accuracy	Trustworthiness	Accuracy	Improvement	Trustworthiness	Improvement
MNIST	Clean	0.9757 ± 0.0144	0.9836 ± 0.0116	0.9643 ± 0.0124	-	0.9540 ± 0.0158	-
	Case I	0.7515 ± 0.0184	0.7450 ± 0.0166	0.9398 ± 0.0125	**25.05%** $\uparrow$	0.9417 ± 0.0104	**26.40%** $\uparrow$
	Case II	0.7824 ± 0.0186	0.7784 ± 0.0118	0.9130 ± 0.0174	**16.69%** $\uparrow$	0.9363 ± 0.0174	**20.28%** $\uparrow$
	Case III	0.7575 ± 0.0146	0.7341 ± 0.0132	0.9027 ± 0.0178	**19.16%** $\uparrow$	0.9228 ± 0.0169	**25.70%** $\uparrow$
CIFAR-10	Clean	0.7210 ± 0.0106	0.7003 ± 0.0192	0.7047 ± 0.0189	-	0.7124 ± 0.0177	-
	Case I	0.3445 ± 0.0196	0.3211 ± 0.0246	0.5078 ± 0.0191	**47.40%** $\uparrow$	0.5133 ± 0.0247	**59.85%** $\uparrow$
	Case II	0.3390 ± 0.0243	0.3674 ± 0.0285	0.5062 ± 0.0233	**49.32%** $\uparrow$	0.5128 ± 0.0182	**39.57%** $\uparrow$
	Case III	0.3375 ± 0.0214	0.3036 ± 0.0197	0.4816 ± 0.0243	**42.69%** $\uparrow$	0.5112 ± 0.0260	**68.37%** $\uparrow$

Case II. As we expected, taking location information of the noisy features as a priori in opinion initialization helps the model operate on reliable features and achieve higher trustworthiness and accuracy. The benefit comes from assigning disbelief mass to noisy features, which greatly affects the feature propagation in *max-trust-pooling* layer.

Case III. Finally, with the least amount of information known about the noise, TrustCNets manage to improve accuracy and trustworthiness compared to its non-trust-aware variants by assigning uncertainty mass to all input features. The results show that TrustCNets in case III have lower numbers than in case I and II, which is reasonable because both location and intensity of noise are unknown in case III.

Furthermore, we also find that TrustCNets have certain noise-tolerant ability when used alone or placed towards the beginning of the neural network (close to the input layer). When placed after non-trust-aware layers, TrustCNets do not possess this ability anymore. This is because that the noise is processed by the proceeding layers and TrustCNet does not have direct access to noise information anymore; hence, it could not improve the results.

Discussion. In this section, we demonstrated that TrustCNets with max-trust-pooling layers are useful in cases involving attribute noise. Besides those CNN architectures containing max-pooling layers, there are many modern network designs do not contain a max-pooling. We would like to briefly discuss some potential future directions on how to utilize trust maps without max-pooling. Max-trust-pooling is a straight-forward way to leverage the quantified trust values in neural network training or design by pooling with trust values. We can also combine trust with dropout layer, to make a trust-guided dropout; or to use it for model architecture selection or tuning, e.g., keep only the essential and trustworthy parts of the architecture to save resource; or we can infuse trust evaluation in continual learning tasks, to identify parameters that are more important and trustworthy during training to reduce catastrophic forgetting and improve model trustworthiness. We believe the trust map and the whole trust quantification framework are worth exploring and useful in many future directions.

8 Conclusions and Discussion

In this work, we propose a trustworthiness quantification framework for CNNs and use it to evaluate popular CNN building blocks and deeper CNN architectures. We design a trust-based *max-pooling* layer and develop TrustCNet, a building block for trust-aware CNN construction. When compared with their non-trust-aware variants, TrustCNets achieves higher accuracy and trust values in cases involving noise in the data. We anticipate this approach will inspire more works on trustworthy DL model design. As part of our future work, we plan to apply our trust quantification framework to state-of-the-art deep learning models employed in cyber-physical systems. We plan to explore the impact of trustworthiness on continual learning. A great deal of effort has been focused on solving catastrophic forgetting, such as EWC (elastic weight consolidation) Kirkpatrick et al. (2017). One of the ideas of such methods is to analyze the impact of various parameters, e.g., the Fisher information matrix is used in EWC to identify which parameters are more important to task A than task B. We believe our trust measure could be adaptive in such cases. For example, when learning new tasks, we identify which parameters are more trustworthy and apply higher weights to them. This could further link to developing trust-aware models and methods for OOD generalization. In summary, we believe this work provides an innovative aspect and is useful in DL model evaluation and design. We hope to inspire more works to use trust in continual learning applications and scenarios involving imperfect data, such as OOD detection and generalization, adversarial attack defense, and noise-resistant and robust deep learning architecture design.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the support by the National Science Foundation under the Career Award CPS/CNS-1453860, the NSF award under Grant Numbers CCF-1837131, MCB-1936775, CNS-1932620, and CMMI 1936624 and the DARPA Young Faculty Award and DARPA Director's Fellowship Award, under Grant Number N66001-17-1-4044, an Air Force Research Laboratory and Innovare Advancement Center Trusted Artificial Intelligence (TAI) Challenge award, a 2021 USC Stevens Center Technology Advancement Grant (TAG) award, an Intel faculty award, and a Northrop Grumman grant. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied by the Defense Advanced Research Projects Agency, the Department of Defense or the National Science Foundation.

CONTRIBUTION

M.C., P.B. and S.N. contributed to the design of the research including simulations and experiments. M.C., S.N. and P.B. contributed to the writing and revision of the manuscript. M.C. and T.S. contributed to the implementation of simulator, running experiments, the preparing the figures, tables, and their captions, and revision of the manuscript. M.C. contributed to organizing the information in the main section and the appendix section.

REFERENCES

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

Chirag Agarwal, Daniel D'souza, and Sara Hooker. Estimating example difficulty using variance of gradients. *arXiv preprint arXiv:2008.11600*, 2020.

Ramón Fernandez Astudillo and João Paulo da Silva Neto. Propagation of uncertainty through multilayer perceptrons for robust automatic speech recognition. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

David Barber and Christopher M Bishop. Ensemble learning in bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, 168:215–238, 1998.

Jorge Bernal Bernabe, Jose Luis Hernandez Ramos, and Antonio F Skarmeta Gomez. Taciot: multidimensional trust-aware access control system for the internet of things. *Soft Computing*, 20(5):1763–1779, 2016.

Jesús Bobadilla, Raúl Lara-Cabrera, Ángel González-Prieto, and Fernando Ortega. Deepfair: deep learning for improving fairness in recommender systems. *arXiv preprint arXiv:2006.05255*, 2020.

Sijia Cai, Wangmeng Zuo, and Lei Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 511–520, 2017.

Mingxi Cheng, Shahin Nazarian, and Paul Bogdan. There is hope after all: Quantifying opinion and trustworthiness in neural networks. *Frontiers in Artificial Intelligence*, 3:54, 2020.

Mingxi Cheng, Chenzhong Yin, Shahin Nazarian, and Paul Bogdan. Deciphering the laws of social network-transcendent covid-19 misinformation dynamics and implications for combating misinformation phenomena. *Scientific Reports*, 11(1):1–14, 2021a.

Mingxi Cheng, Chenzhong Yin, Junyao Zhang, Shahin Nazarian, Jyotirmoy Deshmukh, and Paul Bogdan. A general trust framework for multi-agent systems. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 332–340, 2021b.

Mingxi Cheng, Junyao Zhang, Shahin Nazarian, Jyotirmoy Deshmukh, and Paul Bogdan. Trust-aware control for intelligent transportation systems. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 377–384. IEEE, 2021c.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Charles Corbière, Marc Lafon, Nicolas Thome, Matthieu Cord, and Patrick Pérez. Beyond first-order uncertainty estimation with evidential models for open-world recognition. In *ICML 2021 Workshop on Uncertainty and Robustness in Deep Learning*, 2021.

Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pp. 2189–2200. PMLR, 2021.

Shuiguang Deng, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. On deep learning for trust-aware recommendations in social networks. *IEEE Transactions on Neural Networks and Learning Systems*, 28(5):1164–1177, 2017. doi: 10.1109/TNNLS.2016.2514368.

Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 2020.

Birhanu Eshete. Making machine learning trustworthy. *Science*, 373(6556):743–744, 2021.

Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 317–326, 2016.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.

Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9785–9795, 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pp. 403–412, 2004.

Caner Hazirbas, Joanna Bitton, Brian Dolhansky, Jacqueline Pan, Albert Gordo, and Cristian Canton Ferrer. Towards measuring fairness in ai: the casual conversations dataset. *arXiv preprint arXiv:2104.02821*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

Heinrich Jiang, Been Kim, Melody Y Guan, and Maya Gupta. To trust or not to trust a classifier. *arXiv preprint arXiv:1805.11783*, 2018.

Audun Jøsang. *Subjective logic*. Springer, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Anna-Kathrin Kopetzki, Bertrand Charpentier, Daniel Zügner, Sandhya Giri, and Stephan Günnemann. Evaluating robustness of predictive uncertainty estimation: Are dirichlet-based models reliable? In *International Conference on Machine Learning*, pp. 5707–5718. PMLR, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pp. 99–112. Chapman and Hall/CRC, 2018.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 3569–3577, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

Youngmin Oh, Beomjun Kim, and Bumsub Ham. Background-aware pooling and noise-aware loss for weakly-supervised semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6913–6922, 2021.

Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 464–479, 2018.

John Phillips, Julieta Martinez, Ioan Andrei Bârsan, Sergio Casas, Abbas Sadat, and Raquel Urtasun. Deep multi-task learning for joint localization, perception, and prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4679–4689, 2021.

Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *arXiv preprint arXiv:1806.01768*, 2018.

Glenn Shafer. *A mathematical theory of evidence*, volume 42. Princeton university press, 1976.

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021.

Weishi Shi, Xujiang Zhao, Feng Chen, and Qi Yu. Multifaceted uncertainty estimation for label-efficient deep learning. *Advances in Neural Information Processing Systems*, 33:17247–17257, 2020.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Florentin Smarandache. *An in-depth look at information fusion rules and the unification of fusion theories*. Infinite Study, 2004.

Zhiyuan Su, Mingchu Li, Xinxin Fan, Xing Jin, and Zhen Wang. Research on trust propagation models in reputation management systems. *Mathematical Problems in Engineering*, 2014, 2014.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

Jessica S Titensky, Hayden Jananthan, and Jeremy Kepner. Uncertainty propagation in deep neural networks using extended kalman filtering. In *2018 IEEE MIT Undergraduate Research Technology Conference (URTC)*, pp. 1–4. IEEE, 2018.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Raquel Urena, Gang Kou, Yucheng Dong, Francisco Chiclana, and Enrique Herrera-Viedma. A review on trust propagation and opinion dynamics in social networks and group decision making frameworks. *Information Sciences*, 478:461–475, 2019.

Anisie Uwimana and Ransalu Senanayake. Out of distribution detection and adversarial attacks on deep neural networks for robust medical image analysis. *arXiv preprint arXiv:2107.04882*, 2021.

A"a ron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *CoRR*, abs/1601.06759, 2016. URL http://arxiv.org/abs/1601.06759.

Dongxia Wang, Jie Zhang, et al. Multi-source fusion in subjective logic. In *2017 20th International Conference on Information Fusion (Fusion)*, pp. 1–8. IEEE, 2017.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. *Advances in neural information processing systems*, 33:8588–8601, 2020.

Zikun Yang, Paul Bogdan, and Shahin Nazarian. An in silico deep learning approach to multi-epitope vaccine design: a sars-cov-2 case study. *Scientific reports*, 11(1):1–21, 2021.

Haotian Ye, Chuanlong Xie, Yue Liu, and Zhenguo Li. Out-of-distribution generalization analysis via influence function. *arXiv preprint arXiv:2101.08521*, 2021.

Kai Zhang, Wangmeng Zuo, Shuhang Gu, and Lei Zhang. Learning deep cnn denoiser prior for image restoration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3929–3938, 2017.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European conference on computer vision*, pp. 286–301. Springer, 2016.

Jun-Yan Zhu, Philipp Krahenbuhl, Eli Shechtman, and Alexei A Efros. Learning a discriminative model for the perception of realism in composite images. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3943–3951, 2015.

A Appendix

This section includes introduction the concepts of related subjective logic (SL) knowledge we used in the main text, such as beta distribution (Sec. A.1) and related SL operators (Sec. A.2), related prior works for readability of the manuscript (Sec. A.3), and additional experimental results (Sec. A.4.2). Code to our experiments is in the Supplementary Materials zip file, we will release the Github link once this work get published.

A.1 Beta Distribution and Trustworthiness

A binomial opinion in SL Jøsang (2016) can be interpreted as Beta probability density function. Considering a random variable X which drawn from binary domain $\{x, \bar{x}\}$ has a continuous probability function p. Let $p : X \to [0, 1]$ and $p(x) + p(\bar{x}) = 1$. The corresponding Beta probability density function $(p(x), \alpha, \beta)$ shows as:

$$\begin{cases} \text{Beta}(p(x), \alpha, \beta) = \dfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}(p(x))^{\alpha-1}(1-p(x))^{\beta-1}, \\ \alpha = r_x + a_x W, \\ \beta = s_x + (1-a_x)\,W, \end{cases}$$

where α, β represent evidence of $X = x, X = \bar{x}$. We can get the projected probability of a binomial opinion, trustworthiness value, as $p_x = b_x + u_x a_x$, which is equal to expected probability of Beta probability density function $E[x] = \frac{\alpha}{\alpha+\beta} = \frac{r_x + a_x W}{r_x + s_x + W} = p_x$.

A.2 Related SL Opinion Operators

The following 2 parts introduce main operators for constructing trust model. Notice that the average fusion operator takes multiple source opinions as inputs.

A.2.1 Multiplication Operator

To obtain conjunction of two opinions $x \wedge y$, the following equation is used Jøsang (2016):

$$\overline{W}_{x \cdot y} = \overline{W}_x \cdot \overline{W}_y = \begin{cases} b_{x\wedge y} = b_x b_y + \frac{(1-a_x)a_y b_x u_y + a_x(1-a_y)b_y u_x}{1-a_x a_y} \\ d_{x\wedge y} = d_x + d_y - d_x d_y, \\ u_{x\wedge y} = u_x u_y + \frac{(1-a_y)b_x u_y + (1-a_x)b_y u_x}{1-a_x a_y}, \\ a_{x\wedge y} = a_x a_y. \end{cases}$$

A.2.2 Averaging Fusion Operator

Let $\mathbb{C} = \{C_1, C_2, \ldots C_N\}$ be a frame of N sources with the respective opinions $\omega_X^{C_1}, \omega_X^{C_2}, \ldots \omega_X^{C_N}$ over the same variable X. Let C denote a specific source $C \in \mathbb{C}$. The averaging merger of all the sources in the source frame $\mathbb{C}$ is denoted $\underline{\diamond}(\mathbb{C})$. The opinion $\omega_X^{\diamond(\mathbb{C})} \equiv \left(b_X^{\diamond(\mathbb{C})}, u_X^{\diamond(\mathbb{C})}, a_X^{\diamond(\mathbb{C})}\right)$ is the averaging fused opinion expressed as Wang et al. (2017).

Case I: $\exists u_X^C \neq 0$,

$$\begin{cases} b_X^{\diamond(\mathbb{C})}(x) = \dfrac{\sum_{C\in\mathbb{C}}\left(\boldsymbol{b}_X^C(x)\prod_{C_j \neq C} u_X^{C_j}\right)}{\sum_{C\in\mathbb{C}}\left(\prod_{C_j\neq C} u_X^{C_j}\right)} \\ u_X^{\diamond(\mathbb{C})} = \dfrac{N\prod_{C\in\mathbb{C}} u_X^C}{\sum_{C\in\mathbb{C}}\left(\prod_{C_j\neq C} u_X^{C_j}\right)} \end{cases}$$

Case II: $u_X^C = 0, \quad \forall C \in \mathbb{C}$,

$$\begin{cases} b_X^{\diamond(\mathbb{C})}(x) = \sum_{C\in\mathbb{C}} \gamma_X^C b_X^C(x) \\ u_X^{\diamond(\mathbb{C})} = 0 \\ \gamma_X^C = \lim_{u_X^{\mathbb{C}} \to 0} \dfrac{u_X^C}{\sum_{C_j\in\mathbb{C}} u_X^{C_j}} \end{cases}$$

A.3 Opinion Evaluation of a Single Neuron

For a single neuron in neural networks, during the forward propagation, it takes inputs and weights and produces an output, and in the backward propagation, weights are updated based on gradients. Inspired by this process, the opinion calculation of a single neuron is quantified in a procedure called *opinion propagation* Cheng et al. (2020). Similarly in classical neural networks, opinion forward propagation is nothing but another "feature" propagation, where the feature is an opinion. Each weight is attached with an opinion as well. In opinion backward propagation, the opinions of weights are updated. For an output neuron, the opinion $(\overline{W}_{neuron})$ combines the forward passing opinion $(\overline{W}_N^F)$ and backward updating opinion $(\overline{W}_N^B)$:

$$\overline{W}_{neuron} = fusion(\overline{W}_N^F, \overline{W}_N^B), \tag{7}$$

where N is short for $neuron$. For a hidden neuron, the opinion calculation is simpler as it only contains the forward pass opinion.

A.4 Additional Experimental Results

A.4.1 Trust Quantification Results for Deeper CNNs

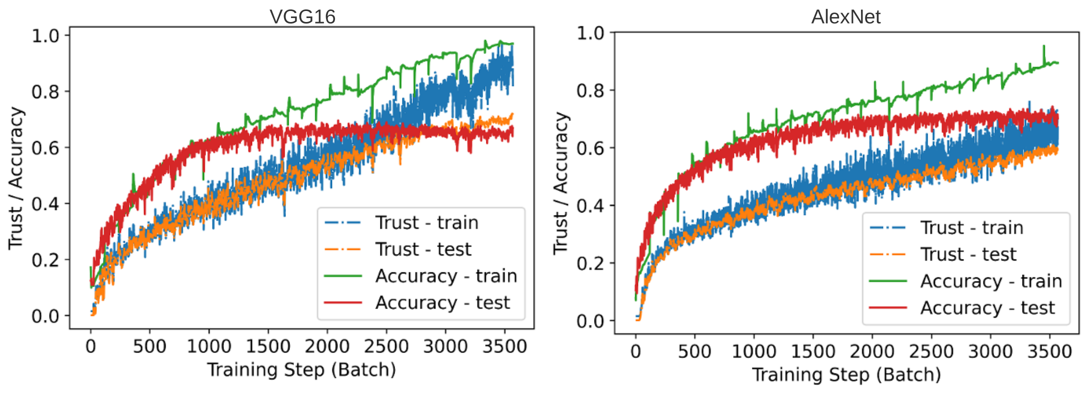

Figure 7: Accuracy and trustworthiness evaluation of VGG16 and AlexNet.

We follow the experiment setup in Sec. 6.1 and perform the same experiment on VGG16 and AlexNet, the results of training / test accuracy and trustworthiness are shown in Fig. 7. We find that with similar accuracy results, trust values are different. For example, VGG16 and AlexNet both achieve similar training accuracy, but their trustworthiness in training is very different. Furthermore, VGG16 results in lower test accuracy compared to AlexNet, but the test trust values of VGG16 is clearly higher. From these observations we find that trust and accuracy are not necessarily positively correlated.

A.4.2 Additional Results for TrustCNet Evaluation

Table 4: Comparison between TrustCNets and their non-trust-aware variants under 4 different types of noise.

	Max-pooling		Max-trust-pooling (TrustCNet)			
Noise	Accuracy	Trustworthiness	Accuracy	Improvement	Trustworthiness	Improvement
MNIST (Gaussian)	0.6577 ± 0.0146	0.6638 ± 0.0132	0.7659 ± 0.0178	**16.45%** $\uparrow$	0.7543 ± 0.0169	**13.63%** $\uparrow$
MNIST (localvar)	0.6477 ± 0.0387	0.6453 ± 0.0279	0.9337 ± 0.0340	**44.15%** $\uparrow$	0.9228 ± 0.0173	**43.00%** $\uparrow$
MNIST (salt)	0.5548 ± 0.0147	0.5276 ± 0.0234	0.6948 ± 0.0226	**25.23%** $\uparrow$	0.6871 ± 0.0154	**30.23%** $\uparrow$
MNIST (s&p)	0.6901 ± 0.0236	0.6890 ± 0.0165	0.7658 ± 0.0171	**10.96%** $\uparrow$	0.7748 ± 0.0235	**12.45%** $\uparrow$
CIFAR10 (Gaussian)	0.3068 ± 0.0165	0.2898 ± 0.0143	0.4290 ± 0.0167	**39.83%** $\uparrow$	0.4235 ± 0.0134	**46.13%** $\uparrow$
CIFAR10 (localvar)	0.5484 ± 0.0324	0.5214 ± 0.0314	0.5770 ± 0.0161	**8.47%** $\uparrow$	0.5634 ± 0.0247	**8.05%** $\uparrow$
CIFAR10 (salt)	0.3985 ± 0.0174	0.3437 ± 0.0194	0.4437 ± 0.0139	**11.34%** $\uparrow$	0.4561 ± 0.0177	**32.70%** $\uparrow$
CIFAR10 (s&p)	0.3597 ± 0.0289	0.3137 ± 0.0127	0.4317 ± 0.0116	**20.01%** $\uparrow$	0.4431 ± 0.0168	**41.24%** $\uparrow$

Evaluation of TrustCNet with different noises. From the previous experiments, we see that TrustCNets are able to improve trust and accuracy when there is Gaussian noise. In this section, we further study the noise-tolerant ability of TrustCNets and include a variety of noise interference. We follow Zhang et al. (2017) for noise setup and add four types of noise to the whole image: (i) Gaussian with 0 mean and 0.2 variance, (ii) Gaussian-distributed additive noise (localvar), (iii) salt noise with 0.2 intensity, and (iv) salt and pepper (s&p) noise with 0.2 intensity. We consider the hardest case III in this experiment and the results are shown in Tab. 4. From the results, we see that TrustCNets achieves better results in all cases, and this demonstrates the noise-tolerant ability of TrustNets when facing with simple noises. TrustCNets with *max-trust-pooling* managed to learn useful information from noisy features, and taking both feature map and trust map into consideration in learning proves to be beneficial in terms of accuracy and trust.