GEOMETRIC OPERATOR CONVOLUTIONAL NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

The Convolutional Neural Network (CNN) has been successfully applied in many fields during recent decades; however it lacks the ability to utilize prior domain knowledge when dealing with many realistic problems. We present a framework called Geometric Operator Convolutional Neural Network (GO-CNN) that uses domain knowledge, wherein the kernel of the first convolutional layer is replaced with a kernel generated by a geometric operator function. This framework integrates many conventional geometric operators, which allows it to adapt to a diverse range of problems. Under certain conditions, we theoretically analyze the convergence and the bound of the generalization errors between GO-CNNs and common CNNs. Although the geometric operator convolution kernels have fewer trainable parameters than common convolution kernels, the experimental results indicate that GO-CNN performs more accurately than common CNN on CIFAR-10/100. Furthermore, GO-CNN reduces dependence on the amount of training examples and enhances adversarial stability.

1 INTRODUCTION

Convolutional Neural Networks have been successfully applied in many fields during recent decades, but the theoretical understanding of the deep neural network is still in the preliminary stages. Although CNNs have strong expressive abilities, they have two clear deficiencies. First, as complex functional mappings, CNNs, like black boxes, cannot take full advantage of domain knowledge and prior information. Second, when little data is available for a certain task, CNNs' generalization ability weakens. This is due to overfitting, which may occur due to the large number of parameters and the large model size. Stemming from these two defects, a great deal of research has been done to modify CNNs (Dai et al., 2017; Wang et al., 2018; Sarwar et al., 2017).

Before CNNs were applied, traditional geometric operators had developed quite well. Each geometric operator represents the precipitation of domain knowledge and prior information. For example, the Sobel operator (Works) is a discrete difference operator, which can extract image edge information for edge detection. The Schmid operator (Schmid, 2001) is an isotropic circular operator, which extracts texture information from images for face recognition. The Histogram of Oriented Gradients (HOG) (Dalal & Triggs, 2005) is a statistic operator of gradient direction, which extracts edge direction distributions from images for pedestrian detection and other uses.

Many computer vision tasks require domain knowledge and prior information. For example, in Cao et al. (2015), the texture information from the image is used for an auxiliary diagnosis of a fracture. Geometric operators can make use of domain knowledge and prior information, but cannot automatically change parameter values by learning from data. Convolutional Neural Networks have strong data expression abilities and learning abilities, but they struggle to make use of domain knowledge. For better data learning, we have combined the two. It is natural to directly use geometric operators for pre-processing, and then classify the data through a Convolutional Neural Network (Yao et al., 2016). However, this method uses human experience to select geometric operator parameter values, and then carries out the Convolutional Neural Network learning separately. This method is a kind of two-stage technique, and without reducing parameter redundancy in a Convolutional Neural Network, it is difficult to achieve global optimization. The method proposed in this paper directly constructs geometric operator convolution and then integrates geometric operator convolution into a Convolutional Neural Network to form a new framework - the Geometric Operator Convolutional

Neural Network. This method achieves global optimizations and utilizes the properties of geometric operators.

In summary, the contributions of this work are as follows:

- This framework can integrates many conventional geometric operators, which reveals its broad customization capabilities when handling diverse problems.
- In theory, the same approximation accuracy and generalization error bounds are achieved when geometric operators meet certain conditions.
- The Geometric Operator Convolutional Neural Network not only reduces the redundancy of the parameters, but also reduces the dependence on the amount of the training samples.
- The Geometric Operator Convolutional Neural Network enhances adversarial stability.

2 Related Work

In recent years, Convolutional Neural Networks have been widely used in various classification and recognition applications (Krizhevsky et al., 2012; Hu et al., 2014). Convolutional Neural Networks have achieved advanced success in various problems. All CNNs adopt an end-to-end approach to learning; however, each unique task is associated with its own distinctive domain knowledge and prior information. Thus, to improve classification accuracy, researchers use priori information that is tailored to each specific task and each specific Convolutional Neural Network. One way to do this is to use the traditional image processing algorithm as a preprocessing step. Another way is to use the traditional image processing algorithm to initialize convolution kernels.

Classification accuracy is a primary concern for researchers in the machine-learning community. Different pre-processing models, such as filters or feature detectors, have been employed to improve the accuracy of CNNs. One example of this is the Gabor filter with CNN (Daugman, 1988). The Gabor filter is a feature extractor based on human vision. Besides the Gabor filter, some people also use Fisher vectors (Cimpoi et al., 2014), sparse filter Banks (Pfister & Bresler, 2015), and the HOG algorithm (Lu et al., 2018) combined with a CNN to improve accuracy. Based on the human visual system, these filters are found to be remarkably well-suited for texture representation and discrimination. In the works by Kwolek (2005) and Mounika et al. (2012), the Gabor filter is used to extract features from the input image in a pre-processing step. However, these methods require a kind of two-stage procedure that may not reach the optimal global solution.

In addition, some scholars use traditional image processing algorithms to initialize convolutional kernels, such as building a Feature Pyramid Network with an image pyramid for multi-scale feature extraction (Lin et al., 2017). Geometric operators are widely used in traditional image processing algorithms. Many researchers use the Gabor filter to fix the first convolution layer, while other layers, which are common convolution layers, can be trained to improve their accuracy (Yao et al., 2016; Sarwar et al., 2017). John et al. simultaneously adopted the weight of the first layer convolution with the Gershgorin circle theorem and the Gabor filter constraint to improve the classification accuracy when Convolutional Neural Networks propagated backward. In Calderón et al. (2003) and Chang & Morgan (2014), the authors have attempted to get rid of the pre-processing overhead by introducing Gabor filters in the first convolutional layer of a CNN. In addition, some researchers use filters to initialize multiple convolutional kernels. Lu et al. (2018) only used the Gabor function to create kernels in four directions to initialize the convolutional kernels from a Convolutional Neural Network. These methods change the initialization weight and use domain knowledge, but they do not reduce the redundancy of model parameters, and they do not enhance the transformation ability of the model.

In this paper, a new network, the Geometric Operator Convolutional Neural Network, is proposed. This method integrates geometric operators, namely the filters, into a convolutional neural network. This network can not only make use of domain knowledge and prior information, but also reduce the redundancy of network parameters and enhance the ability of model transformation. This network's construction is described in detail in the following section.

3 THE FRAMEWORK OF THE GEOMETRIC OPERATOR CONVOLUTIONAL NEURAL NETWORK

3.1 GEOMETRIC OPERATORS

Before the development of deep Convolutional Neural Networks, traditional image feature extraction methods were based on traditional image processing algorithms, primarily geometric operators. At present, a large number of geometric operators have been applied, such as the Scale Invariant Feature Transform (SIFT) (Lowe, 1999), the Roberts operator (Rosenfeld, 1981), the Laplace operator (van Vliet et al., 1989), the Gabor operator (Han & Ma, 2007), and so on. Each operator has different characteristics. Therefore, different geometric operators are used in different application scenarios, according to the characteristics of each unique problem. For example, SIFT looks for feature points in different scale spaces for pattern recognition and image matching. The Roberts operator uses local differences to find edges for edge detection, and the Laplace operator uses isotropic differentials to retain details for image enhancement.

Geometric operators represent the precipitation of domain knowledge and prior knowledge. The GO-CNN is proposed in this paper, which uses the characteristics of geometric operators. The first step in this framework is to convolve geometric operators. In this paper, the Gabor operator and the Schmid operator are mainly used as examples to illustrate how to carry out convolutions and integrate these convolutions into CNNs. Other geometric operators in subsequent studies employ similar concepts.

3.2 CONVOLUTION OF GEOMETRIC OPERATORS

3.2.1 GABOR OPERATOR

In order to study the frequency characteristics of local range signals, Gabor (1946) proposed the famous "Window" Fourier transform (also called the short-time Fourier transform, STFT) in the paper "Theory of communication" in 1946. This is now known as the Gabor operator; when combined with images, it is referred to as the Gabor filter. Until now, the Gabor filter has undergone many developments, and its primary characteristics are listed below. First, the Gabor filter has the advantages of both spatial and frequency signal processing. As shown in Eqn. 1.0, the Gabor operator is essentially a Fourier transform with a gaussian window. For an image, the window function determines its locality in the spatial domain, so the spatial domain information from different positions can be obtained by moving the center of the window. In addition, since the gaussian function remains the same after the Fourier transform, the Gabor filter can extract local information in the frequency domain. Second, the Gabor filter's response to biological visual cells may be an optimal feature extraction method. In 1985, Daugman (1985) extended the Gabor function to a 2-dimensional form and constructed a 2D Gabor filter on this basis. It was surprising to find that the 2D Gabor filter was also able to maintain consistency with the mammalian model of retinal nerve cell reception. Third, the Gabor kernels are similar to the convolution kernels from the first convolutional layer in the CNN. From the visualization of the first convolutional layer in AlexNet, which was proposed by Krizhevsky et al. (2012). Some convolution kernels present geometric properties, as in the kernel function from the Gabor filter. From this feature, it can also be explained that there are parameter redundancies in the Convolutional Neural Network, and the Gabor operator can be convoluted and integrated into CNN. Lastly, the Gabor filter can extract directional correlation texture features from an image.

$$
\begin{aligned}
g_{\theta,\phi,\gamma,\sigma,\lambda}(x,y) &= exp\left(-\frac{x'^2+y'^2}{2\sigma^2}\right) cos\left(\frac{2\pi x'}{\lambda}+\phi\right) \\
x' &= xcos\theta + ysin\theta \\
y' &= -xsin\theta + ycos\theta
\end{aligned}
\tag{1.0}
$$

Since the Gabor operator combines with the CNN in the image, better feature expressions can be obtained. There are two main binding methods. First, the image is preprocessed by the Gabor operator, and then its features are extracted by the CNN. Next, the Gabor operator is convoluted

to form a convolution layer, and then we integrate this convolution into the common Convolutional Neural Network. The second approach is used in this article. As shown in Eqn. 1.0, the Gabor kernel function has 5 parameters, which are obtained by learning and then regenerated into an $m \times m$ kernel. We replace the common convolution kernels with these Gabor kernels to form a convolutional layer. However, for the common convolutional layer, an $m \times m$ convolution kernel is generated by an identity mapping, which requires m^2 parameters. So, our method reduces the number of trainable parameters in the convolutional layer.

3.2.2 SCHMID OPERATOR

In 2001, Schmid (2001) proposed a Gabor-like image filter, namely the Schmid operator. As shown in Eqn. 2.0, its composition is similar to the kernel function of the Gabor operator, so it retains the properties of the Gabor operator. In addition, the Schmid operator has rotation invariance. So, the Schmid operator is convoluted, and we integrate this convolution into common Convolutional Neural Network. This network improves the model's adversarial stability to rotation and improve the image feature extraction effect. Similar to the convolution of the Gabor operator, as shown in Eqn. 2.0, the Schmid kernel function has two parameters, which are obtained by learning and then generated by the Schmid kernel. Finally, we replace common convolution kernels with Schmid kernels to form a convolutional layer.

$$\begin{aligned} F_{\sigma,\tau}(r) &= exp\left(-\frac{r^2}{2\sigma^2}\right) cos\left(\frac{2\pi\tau r}{\sigma}\right) \\ r &= \sqrt{x^2+y^2} \end{aligned} \tag{2.0}$$

In this paper, only two geometric operator convolutions are explained. Similarly, for other geometric operators, operator kernels are generated by operator kernel functions, which replace common convolution kernels to form a convolutional layer. Due to the diversity of geometric operators, different geometric operators can be replaced with geometric operator convolutions, so the geometric operator convolution is customizable. There is a kind of geometric operator to form any kind of geometric operator convolution. Consequently, a question that must be addressed is how we combine multiple geometric operators with common CNNs to form GO-CNNs.

3.3 GEOMETRIC OPERATOR CONVOLUTIONAL NEURAL NETWORK

Since the visualization of the first layer of convolution kernels maintains some geometric characteristics, we replace the convolution kernel in the first convolutional layer by kernel generated from geometric operators, and denote this kind of CNN as Geometric Operator Convolutional Neural Network (GO-CNN). In GO-CNN, kernels from the first convolutional layer are calculated by parameters of various geometric operator functions, and we call these operator functions as generator functions. Then, we concatenate all the calculated convolutional kernels in the last dimension to obtain a complete convolutional kernel. The full procedure is illustrated in Fig. 1. The generated convolution kernel is used as the weight of the first convolution layer in the Geometric Operator Convolutional Neural Network, and then the common convolution layer and output layer are connected. In this way, we have defined the forward propagation of the whole Geometric Operator Convolutional Neural Network. So, in backward propagation, the gradient of loss is transferred to the convolution kernel; this process is different from the usual convolution. Here, the convolution kernel generated by the geometric operator needs to further use the chain derivative rule (i. e., Eqn. 3.0, where L is the loss function, w is each convolution kernel, and p_i is the parameter to generate each convolution kernel) to transfer the gradient to the parameters of each convolution kernel. Then, trainable parameters are updated by gradient descent algorithms, and the whole GO-CNN is complete.

$$\frac{\partial L}{\partial p_i} = \frac{\partial L}{\partial w}\frac{\partial w}{\partial p_i} \tag{3.0}$$

4 THEORETICAL ANALYSES

The whole framework of the Geometric Operator Convolutional Neural Network has been introduced above. Next, we describe how to theoretically analyze the GO-CNN. It is theoretically proved

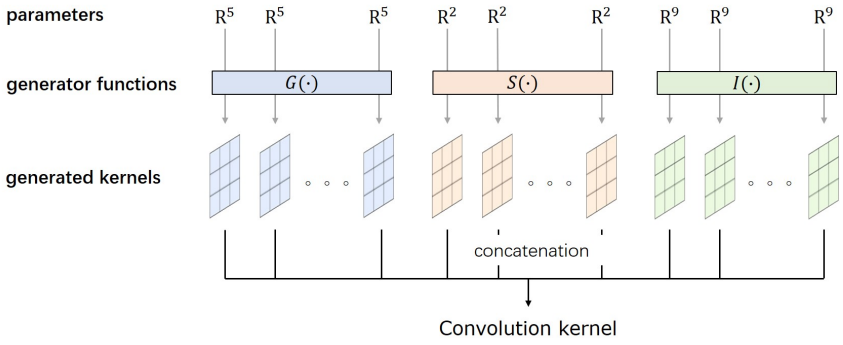


Figure 1: For each out channel, parameters (the first row) are fed into a generator function (the second row), and generate a kernel (the third row). Then, kernels are concatenated to get the convolution kernel in the first convolutional layer. In the second row, G(·) is Gabor kernel function, S(·) is Schmid kernel function, and I(·) is identity mapping.

that although the number of trainable parameters in the GO-CNN decreases, the effectiveness for computer vision tasks does not decrease. All detailed proofs are expanded in Appendix B.

4.1 Definition of data and loss function

- We denote the input by $\mathcal{S} = \{I_i\}_{i=1}^N$, the corresponding label is $\{y_i | y_i = 0 \text{ or } 1\}_{i=1}^N$.
- The loss function is *Mean Square Error*.
- The output of the neural network is $\tilde{y}_i$ for each input I_i, and the empirical loss function is defined as follows:

$$\hat{\mathbb{E}}_S[h] = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2 \tag{4.0}$$

Definition 1 (**Parametric Convolutional Kernel Space**). *Let f be a function that maps vector from $\mathbb{R}^n$ to matrix in $\mathbb{R}^{m\times m}$, $n, m \in \mathbb{N}^+$, and we call this function as convolution kernel generator function. Then we define Parametric Convolutional Kernel Space $\mathcal{K}_f$ as:*

$$\begin{aligned}\mathcal{K}_f = \{[f(p^1), f(p^2), \cdots, f(p^{od})], p^i \in D \subset \mathbb{R}^n, \\ i = 1, 2, \cdots, od\}.\end{aligned} \tag{5.0}$$

We call n the parameter number, m the kernel size, od (short for output dimension) the output dimension. Since a convolutional kernel in a parametric convolutional kernel spaces is generated by function f, we call f as the generator function, and $f_{i,j}(p) = f(p)[i, j]$ as the pixel generator function.

Since kernel can be generated from a generator function by fewer parameters than ordinary kernel, the amount of trainable parameters of GO-CNN can be much smaller. However, reduction in parameters often causes loss of performance as the hypothesis space is smaller. In the simplest situation, we replace the ordinary kernel in the first convolutional layer by the parameter kernel generated from a parametric convolutional kernel space, and study on it.

Definition 2 (**GO-CNN**). *Assume that $\mathcal{K}_f$ is a parametric convolutional kernel space. If the kernel in the first convolutional layer of a convolutional neural network is generated from $\mathcal{K}_f$, we call this network GO-CNN. We denote the set of GO-CNN by $\mathcal{G}_f$.*

GO-CNN is almost exactly the same as common CNN, except for the kernel in the first convolutional layer. We treat the first convolutional layer as a function from images to outputs, which then act as input of the following layer. If this function is not an injective function, meaning that different inputs can be mapped to identical outputs, then the network takes these identical outputs as the input of the following layers, meaning that the final outputs are still the same. However, the image inputs of the first convolutional layer are different, and corresponding labels can also be different. Thus, when the final outputs are the same, errors must occur.

Therefore, we need to choose kernel carefully to make the function be an injective function. Since the convolution operator is a linear operator, we have the following proposition.

Proposition 1. *If the kernel of a convolutional layer, denoted by w, satisfies the following:*

$$I * w = 0 \Leftrightarrow I = 0, \quad \forall I \tag{6.0}$$

where I is the layer input and $*$ is the convolution operation, then this convolutional layer is an injective function.

We find a necessary and sufficient condition for a convolutional layer to be an injective function. But which kernel satisfies this condition? In the proposition below, we show that 3×3 kernel generated by Gabor kernel function satisfies this condition.

Proposition 2. *Let f be the Gabor kernel function, that is $f_{x,y}(\theta, \sigma, \gamma, \lambda, \psi) = exp(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2})cos(2\pi\frac{x'}{\lambda}+\psi)$, where $x' = xcos\theta + ysin\theta, y' = -xsin\theta + ycos\theta$. Let $\mathcal{K}_f$ be the corresponding parametric convolutional kernel space with kernel size m equal to 3 and sufficient output dimension od. Then, there exists kernel in $\mathcal{K}_f$ satisfies the condition (6.0).*

As the kernel generated from $\mathcal{K}_f$ could not meet the (6.0), we have the following definition:

Definition 3 (Well-Defined GO-CNN). *Let $G \in \mathcal{G}_f$, if there is a kernel generated by K_f that satisfies (6.0), we call G a well-defined GO-CNN. We denote the set of all well-defined GO-CNNs as $\mathcal{G}_f^*$.*

Corollary 1. *If the generator function f is Gabor kernel function, the GO-CNN is well-defined.*

Now, let us consider a Convolutional Neural Network with one convolutional layer and two fully connected layers, and we will study the convergency of common CNN and GO-CNN. The detailed mathematical expression is expanded in Appendix B.

Theorem 1. *For any $F \in \mathcal{F}$, where $\mathcal{F}$ is the set of common CNN, if the first fully connected layer is wide enough, the empirical loss of a well-defined GO-CNN can be that of common CNN controls. That is, for an arbitrary $\epsilon > 0$, there exists $d^* \in \mathbb{N}^+$ and $G \in \mathcal{G}_f^*$, such that when $d_1 \geq d^*$, the following inequality holds:*

$$|\hat{\mathbb{E}}_{\mathcal{S}}[G] - \hat{\mathbb{E}}_{\mathcal{S}}[F]| \leq \epsilon \tag{7.0}$$

Theorem 2. *For any $F \in \mathcal{F}$, where $\mathcal{F}$ is the set of common CNN, if the first fully connected layer is wide enough, the generalization error of a well-defined GO-CNN can be that of common Convolutional Neural Network controlled. That is, for an arbitrary $\epsilon > 0$, there exists $d^* \in \mathbb{N}^+$ and $G \in \mathcal{G}_f^*$, such that when $d_1 \geq d^*$, the following inequality holds:*

$$\hat{\mathbb{E}}_{\mathcal{D}}[G] \leq \hat{\mathbb{E}}_{\mathcal{S}}[F] + 2\hat{\mathfrak{R}}_S^a(\mathcal{F}) + \sqrt{\frac{log(1/\delta)}{2N}} + \epsilon \tag{8.0}$$

In Theorem. 2, we know that well defined GO-CNNs have almost the same generalization error as common CNNs. Therefore, we need to find which GO-CNNs are well defined.

As GO-CNN with Gabor kernel function as the *generator function* is well defined, we have the following corollary.

Corollary 2. *Let f be Gabor kernel function, for any $F \in \mathcal{F}$, if the first fully connected layer is wide enough, the generalization error GO-CNN G, which applies f as the generator function can be that of F controlled. That is, for an arbitrary $\epsilon > 0$, there exists $d^* \in \mathbb{N}^+$ and $G \in \mathcal{G}_f^*$, such that when $d_1 \geq d^*$, the following inequality holds:*

$$\hat{\mathbb{E}}_{\mathcal{D}}[G] \leq \hat{\mathbb{E}}_{\mathcal{S}}[F] + 2\hat{\mathfrak{R}}_S^a(\mathcal{F}) + \sqrt{\frac{log(1/\delta)}{2N}} + \epsilon \tag{9.0}$$

More generally, if there are many *generator functions* in the first convolutional layer of a GO-CNN, when the number of kernels generated by Gabor kernel function is sufficient enough, this GO-CNN is also well defined. Therefore, we have the following corollary.

Corollary 3. *Let $\{f_1, f_2, \cdots, f_T\}$ be the set of generator functions. Suppose that there are od convolution kernels $\{k_1, k_2, \cdots, k_{od}\}$ in the first convolutional layer of a GO-CNN, denoted by G, and each k_j is generated by function f_{t_j}, where $1 \leq j \leq od, 1 \leq t_j \leq T$. If there exists $t^* \in \{1, 2, \cdots, T\}$ such that f_{t^*} is Gabor kernel function, and the number of kernels generated by f_{t^*}, denoted by n_{t^*}, is sufficient big enough, then G is well defined, so that (8.0) holds.*

5 EXPERIMENTS

All experiments are performed on a single machine with CPU Intel Core i7-7700 CPU @ 3.60GHz $\times$ 8, GPU TITAN X (Pascal), and RAM 32G. Experimental details and more experiments are given in Appendix C, respectively.

5.1 APPROXIMATION ACCURACY AND GENERALIZATION ERROR

Theoretical analyses ensures that the GO-CNN has the same approximation accuracy and the same upper bound for generalization error as the common CNN. We verify this using two kinds of experiments on CIFAR-10/100 datasets. For the GO-CNN, the convolution kernels from the first layer are half trainable Schmid kernels and half trainable Gabor kernels. The basic network frameworks used

Table 1: The model's accuracy rates averaged over five experiments on the test set

–	CIFAR-10	CIFAR-100
common ResNet18	94.79%	77.06%
GO-ResNet18	**95.17%**	**77.59%**
common ResNet34	95.27%	78.26%
GO-ResNet34	**95.77%**	**78.72%**
common ResNet50	94.44%	78.45%
GO-ResNet50	**94.72%**	**79.50%**

in these experiments are ResNet18, ResNet34, and ResNet50 (He et al., 2016).

According to the cross-entropy curve of CIFAR-10/100 train sets, GO-CNN's value initially fell faster than the common CNN's, eventually almost reaching the same value. It is verified that GO-CNN achieves the same approximation accuracy as the common Convolutional Neural Network. According to the error rate curve of CIFAR-10/100 verification sets, the value of GO-CNN is lower than that of the common CNN. In addition, as shown in Table 1, the GO-CNN on the CIFAR-10 test set was 0.4% more accurate than the common CNN. On the CIFAR-100 test set, the GO-CNN was 0.5% more accurate than the common CNN. It is verified that the GO-CNN achieves the same generalization error bound as the common CNN.

5.2 GENERALIZATION

In many practical applications, such as the military, medical care, and so on, annotated data are often insufficient. Thus, a model's generalization ability for small data sets is of great importance. For CIFAR-10/100 and MNIST datasets, their train sets are large and their test sets are small. So, in these numerical experiments, the test set is directly used to train the model, and the train set is used to evaluate the model. For numerical experiments with CIFAR-10/100 datasets, the techniques and models used are the same as in Sec. 5.1. For numerical experiments with MNIST dataset, the basic network structure used in the experiment is LeNet (LeCun et al., 1998). Similarly, in the GO-CNN, the first convolutional layer is replaced by the operator convolutional layer. The convolution kernels from the first layer are composed of trainable Gabor kernels and Schmid kernels. The other convolutional layers are the common convolutional layers.

As shown in Table 2, from the perspective of the accuracy of MNIST and CIFAR-10/100, after the train set drops to one-fifth of the original train set, the accuracy of the common CNN falls faster than the GO-CNN. Moreover, the GO-CNN more accurate than the common CNN on the original train set. That is to say, the GO-CNN is better at predicting unknown data than the common CNN. The GO-CNN not only reduces the redundancy of the parameters, but also reduces the dependence on the amount of training samples.

5.3 ADVERSARIAL STABILITY

The current machine learning model, including the neural network and other models, is vulnerable to attacks from adversarial samples. In addition, CNNs show instability under attacks against adversarial samples (Goodfellow et al., 2014). So, it is very important to study the stability of adversarial

Table 2: The accuracy of the test set for the small train set and the large train set (in brackets)

–	CIFAR-10	CIFAR-100	MNIST
common ResNet18	84.96%(94.79%)	44.97%(77.06%)	–
GO-ResNet18	**86.21%(95.17%)**	**47.03%(77.59%)**	–
common ResNet34	82.33%(95.27%)	44.74%(78.26%)	–
GO-ResNet34	**86.36%(95.77%)**	**49.00%(78.72%)**	–
common ResNet50	83.86%(94.44%)	45.93%(78.45%)	–
GO-ResNet50	**85.64%(94.72%)**	**47.09%(79.50%)**	–
common LeNet	–	–	97.75%(99.22%)
GO-LeNet	–	–	**97.97%(99.24%)**

Table 3: The adversarial stability of randomly rotated samples and Gaussian disturbance samples. The loss compared to the original accuracy is in brackets, the smaller value is marked in green.

-	**original**	**randomly rotated**	**Gaussian disturbance**
common LeNet	99.22%	58.97%(-40.25%)	95.69%(-3.53%)
GO-LeNet	99.24%	60.20%(-39.04%)	96.31%(-2.93%)

samples in practice. The stability of the model is measured by the difference between the accuracy of the original test set and the adversarial sample generated by the test set. The open handwriting recognition dataset (MNIST) is the primary dataset used in these experiments. The techniques and models are the same as those used for MNIST in Sec. 5.2. Both models are trained on the MNIST train set. Original images, adversarial samples of gaussian interference, and adversarial samples from random rotation are used to evaluate the two models.

It can be seen from Table 3 that when the test set is randomly rotated within 90 degrees, the difference of the GO-CNN is 1.21% lower than that of the common CNN. This verifies that the GO-CNN enhances the adversarial stability of rotated samples. When the small Gaussian disturbance (the mean is 0, the standard deviation is 0.3) is applied to the test set, the difference of the GO-CNN is 0.6% lower than that of the common CNN. This indicates that the GO-CNN enhances the adversarial stability of Gaussian disturbance adversarial samples. In sum, the GO-CNN enhances the adversarial stability of certain adversarial samples.

In the above experiments, the Geometric Operator Convolutional Neural Network uses a priori knowledge from the field of medicine and provides a better recognition effect. Experiments about intelligent medical diagnoses of bone fractures is given in Appendix C. Although the trainable parameters decrease, GO-CNN still reaches the same approximation accuracy and a slightly lower generalization error upper bound when compared with the common CNN. And the GO-CNN reduces the dependence on training samples and enhances the adversarial stability of certain adversarial samples.

6 Conclusion and Future Research

In this paper, we present a novel framework named the Geometric Operator Convolution Neural Network, where the kernel in the first convolutional layer is replaced with kernels generated by geometric operator functions. This new network boasts several contributions. Firstly, the GO-CNN is customizable for diverse situations. Secondly, there is a theoretical guarantee in the learning framework of the GO-CNN. Thirdly, the GO-CNN reduces the dependence on training samples. Lastly, the GO-CNN enhances adversarial stability. In the future, we can explore a more appropriate geometric operator convolution block.

References

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Andrés Calderón, Sergio Roa, and Jorge Victorino. Handwritten digit recognition using convolutional neural networks and gabor filters. *Proc. Int. Congr. Comput. Intell*, 2003.

Yu Cao, Hongzhi Wang, Mehdi Moradi, Prasanth Prasanna, and Tanveer F Syeda-Mahmood. Fracture detection in x-ray images through stacked random forests feature fusion. In *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, pp. 801–805. IEEE, 2015.

Shuo-Yiin Chang and Nelson Morgan. Robust cnn-based speech recognition with gabor filter kernels. In *Fifteenth annual conference of the international speech communication association*, 2014.

Seok Won Chung, Seung Seog Han, Ji Whan Lee, Kyung-Soo Oh, Na Ra Kim, Jong Pil Yoon, Joon Yub Kim, Sung Hoon Moon, Jieun Kwon, Hyo-Jin Lee, et al. Automated detection and classification of the proximal humerus fracture by using deep learning algorithm. *Acta orthopaedica*, pp. 1–6, 2018.

Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep convolutional filter banks for texture recognition and segmentation. *arXiv preprint arXiv:1411.6836*, 2014.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *CoRR, abs/1703.06211*, 1(2):3, 2017.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pp. 886–893. IEEE, 2005.

John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.

John G Daugman. Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on acoustics, speech, and signal processing*, 36(7):1169–1179, 1988.

Dennis Gabor. Theory of communication. part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, 93(26):429–441, 1946.

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

Ju Han and Kai-Kuang Ma. Rotation-invariant and scale-invariant gabor features for texture image retrieval. *Image and vision computing*, 25(9):1474–1481, 2007.

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.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pp. 2042–2050, 2014.

Vijay John, Ali Boyali, and Seiichi Mita. Gabor filter and gershgorin disk-based convolutional filter constraining for image classification.

Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pp. 1094–1096. Springer, 2011.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Bogdan Kwolek. Face detection using convolutional neural networks and gabor filters. In *International Conference on Artificial Neural Networks*, pp. 551–556. Springer, 2005.

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.

Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, pp. 4, 2017.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pp. 1150–1157. Ieee, 1999.

Tongwei Lu, Dandan Wang, and Yanduo Zhang. Fast object detection algorithm based on hog and cnn. In *Ninth International Conference on Graphic and Image Processing (ICGIP 2017)*, volume 10615, pp. 1061509. International Society for Optics and Photonics, 2018.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012.

BR Mounika, NJ Reddy, and VBD Reddy. A neural network based face detection using gabor filter response. *International Journal of Neural Networks (ISSN: 2249-2763 & E-ISSN: 2249-2771)*, 2 (1):06–09, 2012.

Luke Pfister and Yoram Bresler. Learning sparsifying filter banks. In *Wavelets and Sparsity XVI*, volume 9597, pp. 959703. International Society for Optics and Photonics, 2015.

Azriel Rosenfeld. The max roberts operator is a hueckel-type edge detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):101–103, 1981.

Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. Gabor filter assisted energy efficient fast learning convolutional neural networks. *arXiv preprint arXiv:1705.04748*, 2017.

Cordelia Schmid. Constructing models for content-based image retrieval. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pp. II–II. IEEE, 2001.

Lucas J van Vliet, Ian T Young, and Guus L Beckers. A nonlinear laplace operator as edge detector in noisy images. *Computer vision, graphics, and image processing*, 45(2):167–195, 1989.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

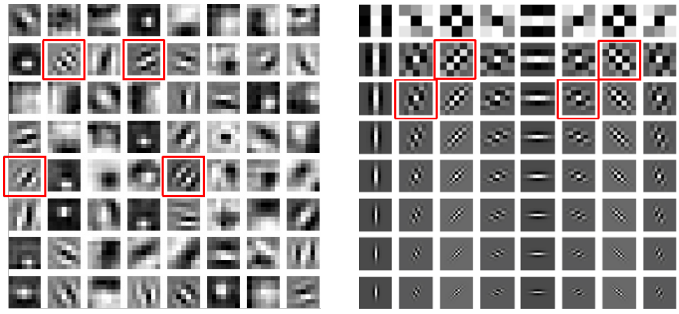
How It Works. Sobel edge detector. *cse. secs. oakland. edu*.

Hu Yao, Li Chuyi, Hu Dan, and Yu Weiyu. Gabor feature based convolutional neural network for object recognition in natural scene. In *Information Science and Control Engineering (ICISCE), 2016 3rd International Conference on*, pp. 386–390. IEEE, 2016.

A Appendix

A supplementary explanation about the Gabor and Schmid operators.

The Gabor kernels are similar to the convolution kernels from the first convolutional layer in the CNN. An illustration of this similarity is shown in Figure 2. In addition, the Gabor filter can extract directional correlation texture features from an image. As shown in Figure 3, there are 40 Gabor



(a) Convolution kernels from the first layer of ResNet50

(b) Gabor kernels

Figure 2: The similarities between the CNN's first convolutional kernels and Gabor kernels.

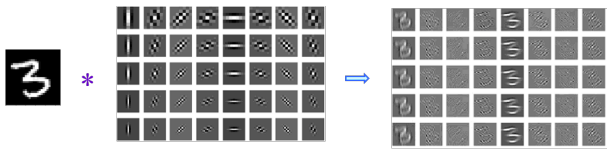


Figure 3: The results of the Gabor operator on an image

kernels from five scales and eight directions convolving with an image. Texture feature maps in different directions can be obtained from the original image.

As shown in Figure 4, when the original image and a version of that image that has been rotated 90 degrees are both convolved with the same Schmid kernel, the resulting characteristic graph exhibits only 90 degrees of rotation. So, the Schmid operator has rotation invariance.

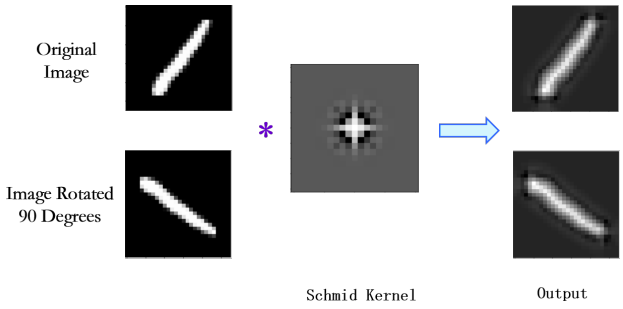


Figure 4: The results of the Schmid operator on an image

B APPENDIX

Lemma 1. *(Cybenko, 1989) Define a functional class* $\Pi \subset \{f \mid f : \mathbb{R}^d \mapsto [0,1]\}$*, where each* $f \in \Pi$ *can be approximated with error at most* ϵ *by a one hidden-layer neural network* N*, that is:*

$$|f(x) - N(x)| \leq \epsilon, \quad \forall x \tag{10.0}$$

Lemma 2. *(Bartlett & Mendelson, 2002) let* $\mathcal{F}$ *and* $\mathcal{G}$ *be two hypothesis classes and let* $a \in \mathbb{R}$ *be a constant, we have:*

$$\begin{aligned} \mathcal{F} \subseteq \mathcal{G} \Rightarrow \hat{\mathfrak{R}}_S^a(\mathcal{F}) \leq \hat{\mathfrak{R}}_S^a(\mathcal{G}) \\ \hat{\mathfrak{R}}_S^a(\mathcal{F}+\mathcal{G}) \leq \hat{\mathfrak{R}}_S^a(\mathcal{F}) + \hat{\mathfrak{R}}_S^a(\mathcal{G}) \end{aligned} \tag{11.0}$$

Lemma 3. *(Mohri et al., 2012) Let z be a random variable of support $\mathcal{Z}$ and distribution $\mathcal{D}$. Let $S = \{z_1, z_2, \cdots, z_N\}$ be a data set of N i.i.d. samples drawn from $\mathcal{D}$. Let $\mathcal{F}$ be a hypothesis class satisfying $\mathcal{F} \subseteq \{f \mid f : \mathcal{Z} \to [a, a+1]\}$. Fix $\delta \in (0,1)$. With probability at least $1-\delta$ over the choice of S, the following holds for all $h \in \mathcal{F}$:*

$$\mathbb{E}_{\mathcal{D}}[h] \leq \hat{\mathbb{E}}_S[h] + 2\hat{\mathfrak{R}}_S^a(\mathcal{F}) + \sqrt{\frac{log(1/\delta)}{2N}} \tag{12.0}$$

Proof of Proposition1.
Assume that the proposition is not true, then there exist $I_1 \neq I_2$, such that $I_1 * w = I_2 * w$. Thus, if we set $I = I_1 - I_2$, we have $I * w = (I_1 - I_2) * w = 0$, since $*$ is a linear operator, which means that $I = 0$ according to the condition. Therefore, the assumption is not true, and the conclusion is proved. □

Proof of Proposition2.
Assume that there exists $I \in \mathbb{R}^{3\times 3}, I \neq 0$, such that $I * k = 0$ holds for $\forall k \in \mathcal{K}_f$.

We write I in the following matrix way:

$$I = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \tag{13.1}$$

We define the *pixel generator function* f_{ij} to be $f_{x-1,y-1}(\theta, \sigma, \gamma, \lambda, \psi)$. Then, we have the following equivalence:

$$I * k = 0 \iff \sum_{i=0}^{2}\sum_{j=0}^{2} a_{ij} f_{ij} = 0. \tag{13.2}$$

We will choose a variety of different parameters to discuss.

I. $\theta = 0,\ \lambda = 1,\ \psi = 0$.

Since $\theta = 0$, we have $x' = x, y' = y$, and the following:

$$\begin{cases} f_{00} = f_{02} = f_{20} = f_{22} = \exp\left(-\frac{1+\gamma^2}{2\sigma^2}\right) \\ f_{01} = f_{21} = \exp\left(-\frac{1}{2\sigma^2}\right) \\ f_{10} = f_{12} = \exp\left(-\frac{\gamma^2}{2\sigma^2}\right) \\ f_{11} = 1 \end{cases} \tag{13.3}$$

We make the following shorthands for conveniency:

$$\begin{aligned} h_1 &= \exp\left(-\frac{1+\gamma^2}{2\sigma^2}\right) \\ h_2 &= \exp\left(-\frac{1}{2\sigma^2}\right) \\ h_3 &= \exp\left(-\frac{\gamma^2}{2\sigma^2}\right) \end{aligned} \tag{13.4}$$

From Eqn.13.2, we can get:

$$(a_{00} + a_{02} + a_{20} + a_{22})h_1 + (a_{01} + a_{21})h_2 + (a_{10} + a_{12})h_3 + a_{11} = 0. \tag{13.5}$$

The equation above means that, $\exists\ b_0, b_1, b_2, b_3 \in \mathbb{R}$, such that

$$b_0 + b_1 h_1 + b_2 h_2 + b_3 h_3 = 0. \tag{13.6}$$

Differentiate on both sides of parameter γ and get:

$$\begin{aligned} &-\frac{\gamma}{\sigma^2} b_1 h_1 - \frac{\gamma}{\sigma^2} b_3 h_3 = 0 \\ \Longrightarrow &b_1 + \exp\left(\frac{\gamma^2}{\sigma^2}\right) b_3 = 0 \end{aligned} \tag{13.7}$$

Since Eqn.13.7 holds for $\forall \gamma, \sigma$, which indicates that:

$$b_1 = b_3 = 0 \tag{13.8}$$

In the same way, we can get the following equation from Eqn.13.8:

$$\begin{aligned} &b_2 h_2 + b_0 = 0 \\ &b_0 = b_2 = 0 \end{aligned} \tag{13.9}$$

Therefore, we have the following equations:

$$\begin{cases} a_{00} + a_{02} + a_{20} + a_{22} = 0 \\ a_{01} + a_{21} = 0 \\ a_{10} + a_{12} = 0 \\ a_{11} = 0 \end{cases} \tag{13.10}$$

II. $\theta = 0,\ \lambda = 3,\ \psi = \pi/3$.

In the same way, we have the following equations:

$$\begin{cases} f_{20} = f_{21} = f_{22} = 0 \\ f_{00} = f_{02} = \frac{1}{2} h_1 \\ f_{10} = f_{12} = \frac{1}{2} h_3 \\ f_{01} = \frac{1}{2} h_2 \\ f_{11} = \frac{1}{2} \end{cases} \tag{13.11}$$

And we can get:

$$(a_{00} + a_{02})h_1 + a_{01} h_2 + (a_{10} + a_{12})h_3 + a_{11} = 0, \tag{13.12}$$

which indicates that:

$$\begin{cases} a_{00} + a_{02} = 0 \\ a_{01} = 0 \end{cases} \tag{13.13}$$

From Eqn.13.10, we can get:

$$\begin{cases} a_{00} + a_{02} = 0 \\ a_{01} = a_{21} = 0 \end{cases} \tag{13.14}$$

III. $\theta = 0,\ \lambda = 3,\ \psi = -\pi/3$.

We can get the following equations in the way just the same as discussed in situation II:

$$a_{20} + a_{22} = 0 \tag{13.15}$$

IV. $\theta = \pi/2,\ \lambda = 3,\ \psi = \pm\pi/3$.

We have $x' = y, y' = x$ this time, and we can get the following equations as the way discussed in situation II III, :

$$\begin{cases} a_{00} + a_{20} = 0 \\ a_{02} + a_{22} = 0 \\ a_{10} = a_{12} = 0 \end{cases} \tag{13.16}$$

Combine equations 13.14 13.15 13.16, we can get:

$$\begin{cases} a_{00} = -a_{02} = a_{22} = -a_{20} = v \\ a_{01} = a_{10} = a_{12} = a_{21} = 0 \end{cases} \tag{13.17}$$

V. $\theta = \pi/4,\ \lambda = 2,\ \psi = \sqrt{2}\pi$.

We have $x' = \frac{\sqrt{2}}{2}(x + y), y' = \frac{\sqrt{2}}{2}(y - x)$ and the following:

$$\begin{cases} f_{00} &= \exp\left(-\frac{1}{\sigma^2}\right) \\ f_{02} &= f_{20} = \exp\left(-\frac{\gamma^2}{\sigma^2}\right) \\ f_{22} &= \exp\left(-\frac{1}{\sigma^2}\right)\cos(2\sqrt{2}\pi) \end{cases} \tag{13.18}$$

Therefore, we have

$$\begin{aligned} &(1 + \cos(2\sqrt{\pi}))\exp(-\frac{1}{\sigma^2})v + 2\cos(2\sqrt{2}\pi)\exp(-\frac{\gamma^2}{\sigma^2})v = 0 \\ &\implies v = 0 \\ &\implies a_{00} = a_{02} = a_{20} = a_{22} = 0 \end{aligned} \tag{13.19}$$

Combine equations 13.10 13.17 13.19, we can find that $a_{ij} = 0, fori, j = 0, 1, 2$, which means that $I = 0$. Therefore, the assumption that $I \neq 0$ is not true.

For an arbitrary sized input I, we can focus on the 3×3 sized submatrix that will do inner product with the convolution kernel and get the same conclusion.

□

Proof of Corollary1.
From Prop.2, the conclusion is obvious. □

For the common CNN, denoted by F, we define the convolution kernel as k_F. The weights of the rest of fully connected layers are $\{a_{F,1}, a_{F,2}\}$, and the biases of three layers are $\{b_{F,0}, b_{F,1}, b_{F,2}\}$. Let σ stand for *sigmoid activation function*, then the convolutional layer C_F and the fully connected layer FC_F can be defined as follows:

$$\begin{aligned} C_F(x) &= x * k_F + b_{F,0} \\ FC_{F,k}(x) &= a_{F,k}x + b_{F,k}, \quad k = 1, 2 \end{aligned} \tag{14.1}$$

Then, the last two fully connected layers can be defined as:

$$D_F(x) = FC_{F,2} \circ \sigma \circ FC_{F,1}(x) \tag{14.2}$$

Therefore, the output before activation, denoted by $F(x)$, and after activation, denoted by $\tilde{F}(x)$, are defined as:

$$\begin{aligned} F(x) &= D_F \circ \sigma \circ C_F(x) \\ \tilde{F}(x) &= \sigma \circ F(x) \end{aligned} \tag{14.3}$$

We denote the set of common CNN as $\mathcal{F}$, that is, $\mathcal{F} = \{F\}$, and the output before activation and after activation of input I_i as $F_i, \tilde{F}_i$.

For a GO-CNN G, we similarly define the convolutional kernel to be k_g, and the weights and biases are $\{a_{G,1}, a_{G,2}, b_{G,0}, b_{G,1}, b_{G,2}\}$. Then, we have the following shorthand when the input is x:

$$\begin{aligned} C_G(x) &= x * k_g + b_{G,0} \\ FC_{G,k} &= a_{G,k}x + b_{G,k}, \quad k = 1, 2 \\ D_G(x) &= FC_{G,2} \circ \sigma \circ FC_{G,1}(x) \\ G(x) &= D_G \circ \sigma \circ C_G(x) \\ \tilde{G}(x) &= \sigma \circ G(x) \end{aligned} \tag{14.4}$$

We denote the output before activation and after activation of input I_i as $G_i, \tilde{G}_i$ as well.

We maintain the same neuron number for each corresponding layer in common CNN and GO-CNN, that is to say, $dim(b_{F,k}) = dim(b_{G,k}), k = 0, 1, 2$, since the approximation ability is different when the neuron number is different. We define the width of each layer as $d_k = dim(b_{F,k}) = dim(b_{G,k}), k = 0, 1, 2$.

For the common CNN, denoted by F, we define the convolution kernel as k_F. The weights of the rest of fully connected layers are $\{a_{F,1}, a_{F,2}\}$, and the biases of three layers are $\{b_{F,0}, b_{F,1}, b_{F,2}\}$. Let σ stand for *sigmoid activation function*, then the convolutional layer C_F and the fully connected layer FC_F can be defined as follows:

$$\begin{aligned} C_F(x) &= x * k_F + b_{F,0} \\ FC_{F,k}(x) &= a_{F,k}x + b_{F,k}, \quad k = 1, 2 \end{aligned} \tag{15.1}$$

Then, the last two fully connected layers can be defined as:

$$D_F(x) = FC_{F,2} \circ \sigma \circ FC_{F,1}(x) \tag{15.2}$$

Therefore, the output before activation, denoted by $F(x)$, and after activation, denoted by $\tilde{F}(x)$, are defined as:

$$\begin{aligned} F(x) &= D_F \circ \sigma \circ C_F(x) \\ \tilde{F}(x) &= \sigma \circ F(x) \end{aligned} \tag{15.3}$$

We denote the set of common CNN as $\mathcal{F}$, that is, $\mathcal{F} = \{F\}$, and the output before activation and after activation of input I_i as $F_i, \tilde{F}_i$.

For a GO-CNN G, we similarly define the convolutional kernel to be k_g, and the weights and biases are $\{a_{G,1}, a_{G,2}, b_{G,0}, b_{G,1}, b_{G,2}\}$. Then, we have the following shorthand when the input is x:

$$\begin{aligned} C_G(x) &= x * k_g + b_{G,0} \\ FC_{G,k} &= a_{G,k}x + b_{G,k}, \quad k = 1, 2 \\ D_G(x) &= FC_{G,2} \circ \sigma \circ FC_{G,1}(x) \\ G(x) &= D_G \circ \sigma \circ C_G(x) \\ \tilde{G}(x) &= \sigma \circ G(x) \end{aligned} \tag{15.4}$$

We denote the output before activation and after activation of input I_i as $G_i, \tilde{G}_i$ as well.

We maintain the same neuron number for each corresponding layer in common CNN and GO-CNN, that is to say, $dim(b_{F,k}) = dim(b_{G,k}), k = 0, 1, 2$, since the approximation ability is different when the neuron number is different. We define the width of each layer as $d_k = dim(b_{F,k}) = dim(b_{G,k}), k = 0, 1, 2$.

Proof of Theorem1.
Notice that

$$\begin{aligned} N(\hat{\mathbb{E}}_{\mathcal{S}}[G] - \hat{\mathbb{E}}_{\mathcal{S}}[F]) &= \sum_i \tilde{G}_i^2 - \tilde{F}_i^2 - 2y_i(\tilde{G}_i - \tilde{F}_i) \\ &= \sum_i (\tilde{G}_i - \tilde{F}_i)(\tilde{G}_i + \tilde{F}_i - 2y_i) \end{aligned} \tag{16.1}$$

Apply absolute value on both sides

$$\begin{aligned} N|\hat{\mathbb{E}}_{\mathcal{S}}[G] - \hat{\mathbb{E}}_{\mathcal{S}}[F]| &\leq \sum_i |\tilde{G}_i - \tilde{F}_i||\tilde{G}_i + \tilde{F}_i - 2y_i| \\ &\leq 4 \sum_i |\tilde{G}_i - \tilde{F}_i| \end{aligned} \tag{16.2}$$

The last inequality holds as $|\tilde{F}_i|, |\tilde{G}_i|, |y_i| \leq 1$.

We can fix parameters of ordinary CNN, so that there is a mapping between input I_i and output $\tilde{F}_i$, and the mapping function is $\tilde{F}$ as we have defined.

We can also fix parameters of C_G, and choose the convolution kernel of C_G that satisfies (**??**) since G is a well-defined GO-CNN, so that C_G is an injective function, which means that C_G^{-1} exists. In the same time, $D_G = FC_{G,2} \circ \sigma \circ FC_{G,1}$ can be treated as a one hidden layer neural network.

Define a new hypothesis $\tilde{h} = \sigma \circ F \circ C_G^{-1} \circ \sigma^{-1}$ ranges in $[0, 1]$, according to Lemma.1, we can find parameters $\{a_{G,k}, b_{G,k}\}, k = 1, 2$, such that

$$|\sigma \circ D_G(x) - \tilde{h}(x)| \leq \epsilon/4, \quad \forall x. \tag{16.3}$$

Replace x by $\sigma \circ C_G(I_i)$ we can get

$$\begin{aligned} &|\tilde{G}_i - \tilde{F}_i| \\ &= |\tilde{G}(I_i) - \tilde{F}(I_i)| \\ &= |\sigma \circ D_G \circ \sigma \circ C_G(I_i) - \sigma \circ F(I_i)| \\ &= |\sigma \circ D_G(\sigma \circ C_G(I_i)) - \sigma \circ F \circ C_G^{-1} \circ \sigma^{-1}(\sigma \circ C_G(I_i))| \\ &\leq \epsilon/4 \end{aligned} \tag{16.4}$$

Combine with (16.2), we can get

$$|\hat{\mathbb{E}}_{\mathcal{S}}[G] - \hat{\mathbb{E}}_{\mathcal{S}}[F]| \leq \frac{4}{N} \sum_i |\tilde{G}_i - \tilde{F}_i| \leq \frac{4}{N} \frac{N}{4} \epsilon = \epsilon \tag{16.5}$$

□

Proof of Theorem2.
From Theorem.1, we know that G satisfies the following inequality:

$$|\hat{\mathbb{E}}_{\mathcal{S}}[G] - \hat{\mathbb{E}}_{\mathcal{S}}[F]| \leq \epsilon \tag{17.1}$$

From Lemma.3, we know that

$$\begin{aligned} \hat{\mathbb{E}}_{\mathcal{D}}[G] \leq & \hat{\mathbb{E}}_{\mathcal{S}}[G] + 2\hat{\mathfrak{R}}^a_S(\mathcal{G}^*_f) + \sqrt{\frac{log(1/\delta)}{2N}} \\ \leq & \hat{\mathbb{E}}_{\mathcal{S}}[F] + 2\hat{\mathfrak{R}}^a_{\mathcal{S}}(\mathcal{G}^*_f) + \sqrt{\frac{log(1/\delta)}{2N}} + \epsilon \end{aligned} \tag{17.2}$$

Since $\mathcal{G}^*_f \subset \mathcal{F}$, we have the following inequality from Lemma.2:

$$\hat{\mathfrak{R}}^a_{\mathcal{S}}(\mathcal{G}^*_f) \leq \hat{\mathfrak{R}}^a_{\mathcal{S}}(\mathcal{F}) \tag{17.3}$$

Combined with (17.2), we have

$$\hat{\mathbb{E}}_{\mathcal{D}}[G] \leq \ \hat{\mathbb{E}}_{\mathcal{S}}[F] + 2\hat{\mathfrak{R}}^a_S(\mathcal{F}) + \sqrt{\frac{log(1/\delta)}{2N}} + \epsilon \tag{17.4}$$

The conclusion is proved!

□

Proof of Corollary2.
From Theorem.2 and Corollary.1, this conclusion is obvious. □

Proof of Corollary3.
Let $\mathcal{K}$ be the set of k_i such that the generator function of k_i is $f_{t_i} = f_{t^*}$ and denote the concatenation of all these k_i as $\hat{k}$.

Suppose that there exists an input x, satisfies that $x * k_i = 0, i = 1, 2, \cdots, od$, then $x * \tilde{k} = 0, \forall \tilde{k} \in \mathcal{K}$. Therefore, $x * \hat{k}$ holds for any parameters. However, it is conflict with Prop.2.

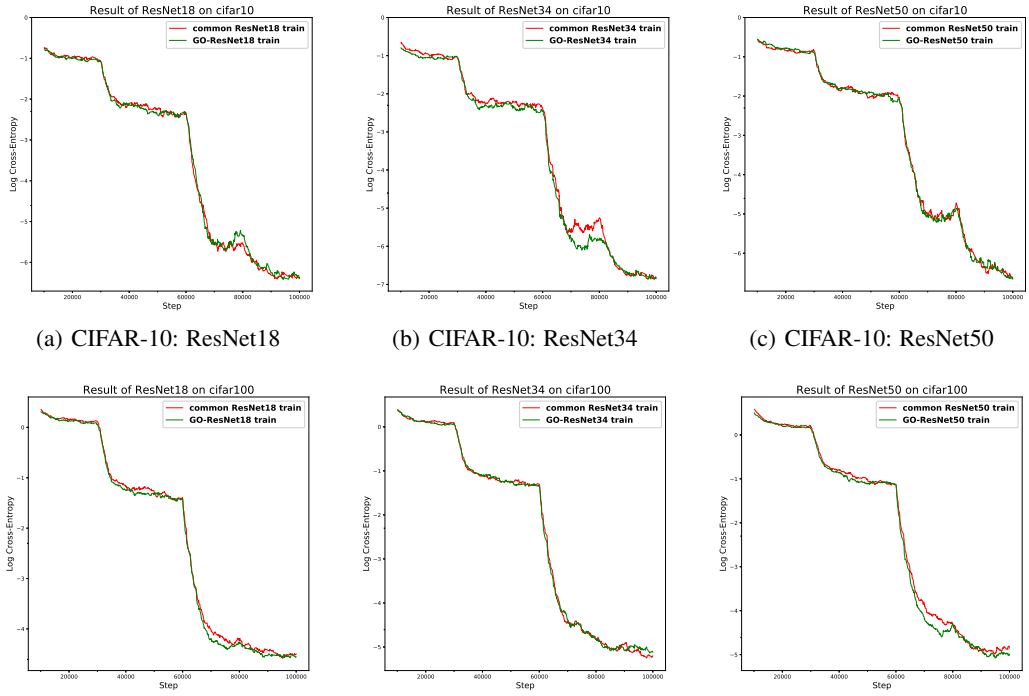
Therefore, the conclusion is proved! □

C APPENDIX

Approximation accuracy and generalization error Recognizing objects in an actual scene is not dependent on corresponding domain knowledge but on humans' prior information. For object recognition tasks, the Geometric Operator Convolutional Neural Network's recognition effect is worth exploring. The commonly used public data sets for common object recognition are CIFAR-10 (ten categories) and CIFAR-100 (100 categories). They are all three-channel color images with a resolution of 32×32. The train set contains 50,000 images and the test set contains 10,000 images. The basic network frameworks used in these experiments are ResNet18, ResNet34, and ResNet50 (He et al., 2016), which mainly consist of a new residual structure unit. In these experiments, four paddings are added on the four edges. Then, a random 32×32 cropping is performed, and a data enhancement method is carried out, which involves turning the image up and down. For both testing and training, the images' pixels are normalized to a 0-1 distribution. The Stochastic gradient descent optimization algorithm with 0.9 the momentum (Loshchilov & Hutter, 2016) is used during the training process. The batch size is 100, the initial learning rate is 0.1, and the weight decay is 0.0005. The learning rate is reduced by one fifth per 60, 120, and 160 epochs. We report the performance of our algorithm on a test set after 200 epochs based on the average over five runs.

As shown in Figure 5, according to the cross-entropy curve of the CIFAR-10 and CIFAR-100 train sets, GO-CNN's value initially fell faster than the common CNN's, eventually almost reaching the same value. It is verified that Geometric Operator Convolutional Neural Network achieves the same approximation accuracy as the common Convolutional Neural Network. According to the error rate

curve of the CIFAR-10 and CIFAR-100 verification set (Figure 6), the value of Geometric Operator Convolutional Neural Network is lower than that of the common Convolutional Neural Network.



(a) CIFAR-10: ResNet18 (b) CIFAR-10: ResNet34 (c) CIFAR-10: ResNet50

(d) CIFAR-100: ResNet18 (e) CIFAR-100: ResNet34 (f) CIFAR-100: ResNet50

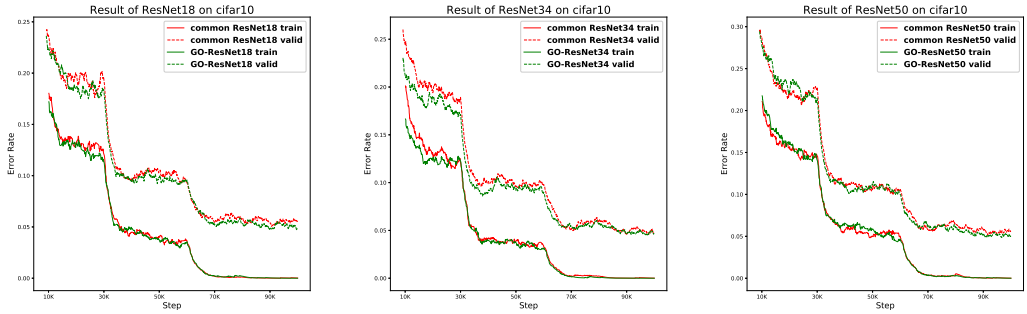
Figure 5: Log of cross entropy curve during training in common ResNet 18-34-50 and GO-ResNet 18-34-50.

Feature visualization One way to evaluate a model is through visualizing the features that the model extracts; this is called feature visualization. T-SNE (Maaten & Hinton, 2008) or PCA (Jolliffe, 2011) are generally used for visualization. The T-SNE visualization maps data points to a two-dimensional or three-dimensional probability distribution through affinitie transformation. Then, the data points are displayed with a two-dimensional or three-dimensional plane.

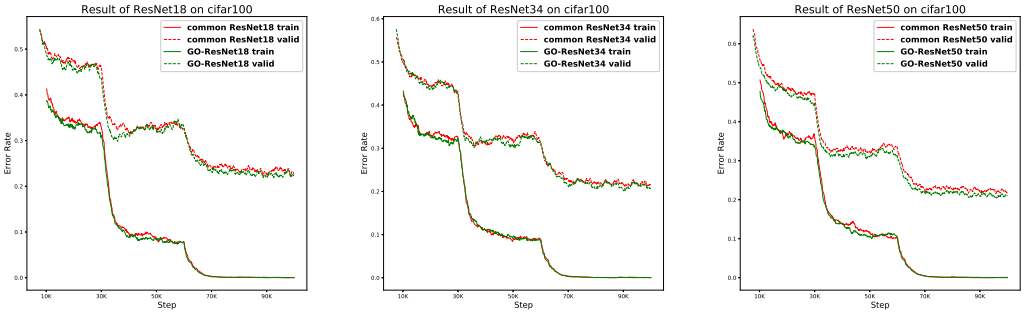
In this paper, a two-dimensional T-SNE visualization is adopted to display the CIFAR-10 features extracted by the model. As shown in Figure 7, the CIFAR-10 features extracted by the Geometric Operator Convolutional Neural Network are evenly separated from each other in the two-dimensional visualization of T-SNE, while the features extracted from the common Convolutional Neural Network are mixed. It is apparent that the features extracted by the GO-CNN are more separable; in other words, the features learned by the GO-CNN are more distinguishable and easy to classify with the last fully connected layer.

Generalization MNIST is a public, handwritten recognition dataset with a total of ten classes. This dataset is a channel image with 28×28 resolution and a clean background. The train set contains 50,000 images and the test set contains 10,000 images. For numerical experiments with the MNIST data set, the adaptive moment estimation (Kingma & Ba, 2014) optimization algorithm is used. In addition, as an image enhancement strategy, the image padding is increased to 32×32 during the training process. The batch size is 100, the initial learning rate is 0.001, and the weight decay is 0.0005. The learning rate stays the same until reaching 20,000 iterations. Consequently, we complete 20,000 iterations on one test set and average the performance over five runs in order to report the final performance evaluation of our algorithm.

Application Medical images in China are developing rapidly, but specialist doctors are short of resources, and they are mainly concentrated in big cities and big hospitals. Many small and medium-sized cities do not have sufficient diagnostic imaging capacities, so many patients have to go to big cities in order to access better medical resources and obtain better treatment. Similarly, there are few

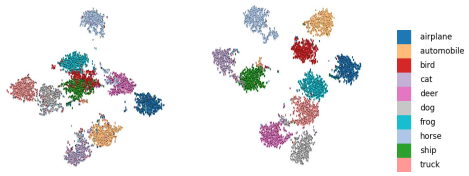


(a) CIFAR-10: ResNet18 (b) CIFAR-10: ResNet34 (c) CIFAR-10: ResNet50



(d) CIFAR-100: ResNet18 (e) CIFAR-100: ResNet34 (f) CIFAR-100: ResNet50

Figure 6: Error rate curve during training in common ResNet 18-34-50 and GO-ResNet 18-34-50.



(a) Common CNN (b) GO-CNN

Figure 7: T-SNE two-dimensional visualization of CIFAR-10

orthopaedic surgeons in China. Fractures often occur in real life due to accidents, such as falls and car accidents. Orthopedists usually use X-ray images to diagnose fractures. With the development of artificial intelligence technology, many scholars use Convolutional Neural Networks to assist doctors in determining whether a bone image reveals a fracture (Chung et al., 2018).

Doctors usually judge whether a fracture has occurred based on whether there is a fracture line (texture) in the image. In Cao et al. (2015), the texture information from the image is used for an auxiliary diagnosis of a fracture. With prior information from the Schmid operator, we do pre-processing by Schmid operators to enhance the texture information from an image. Then, we use the CNN to conduct classification. However, the parameters of geometric operators in this two-stage method are preset by human experience. At this point, it is difficult for the local parameters obtained by the respective optimization to reach the global optimum. Thus, one may consider integrating the preprocessing of geometric operators into the deep network for global parameter learning without prior artificial empirical design parameters. In other words, this would mean using the GO-CNN proposed in this paper, wherein the convolution kernels from the first layer are all trainable Schmid kernels.

Around 2,000 samples from X-rays taken at the Hainan People's Hospital were used as the data for the three kinds of intelligent fracture diagnosis models. Each sample was manually divided into bone

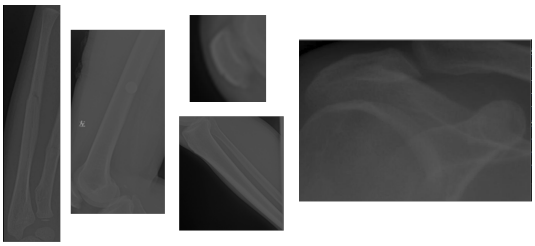


Figure 8: Bone images

regions, as shown in Figure 8, with a total of 5,743 bone regions, including 723 bone fracture regions. The above three models are used for numerical experiments. The basic network framework used in the experiment is ResNet50 (He et al., 2016), which mainly consists of a new residual structure unit. To balance the data during training, the number of fracture patches is increased to 4,016 by rotating the images and changing the background of the images. In the test set, there are 145 fracture patches and 1,004 non-fracture patches. Then, five experiments are conducted to evaluate each model. The SGD algorithm and the finetune strategy are used during the training process, with a batch size of 50. The initial learning rate is 0.001 and the weight decay is 0.0005. The learning rate is reduced by one fifth every 4,000 iterations. Each data class is queued, and the data from each batch is averaged out of each data class during training. We report the performance of our algorithm on the test set after 12,000 iterations based on the average over five runs.

According to Table 4, the Geometric Operator Convolutional Neural Network is the most accurate. Moreover, the fracture recall of the two-stage method is 0.77% higher than that of the Convolutional Neural Network, indicating that domain knowledge from the field of medicine is important for intelligent diagnosis. The fracture recall of the Geometric Operator Convolutional Neural Network is 2.21% higher than that of the two-stage method, which indicates that the Geometric Operator Convolutional Neural Network does make use of medical knowledge for fracture diagnosis. The integration of geometric operator into the deep neural network indeed achieve global optimization.

Table 4: Experimental results of intelligent diagnosis

TestSet	CNN	two-stage	GO-CNN
accuracy	92.38%	93.05%	**93.98%**
fracture recall	87.97%	88.74%	**90.95%**
non-fracture recall	96.57%	96.17%	**96.87%**