Rotation Invariance Neural Network

Shiyuan Li

Abstract

Rotation invariance and translate invariance have great values in image recognition. In this paper, we bring a new architecture in convolutional neural network (CNN) to achieve rotation invariance and translate invariance in 2-D symbol recognition. We can also get the position and orientation of the 2-D symbol by the network to achieve detection purpose for multiple non-overlap target. Human being have the ability look at an object by one glance and remember it, we also can use this architecture to achieve this one shot learning.

1 Introduction

Convolutional neural networks have recently shown outstanding performance on image classification tasks [14]. Have rotation and translate invariance are important goals for model design in image recognition task [2, 11, 13]. Although convolutional neural network can achieve some kind of rotation and translate invariance in image recognition, yet that is based on large train data and not good enough for real world applications [3], which need almost same performance for same target in different position and orientation, like satellite pictures[3, 4] or microscope pictures [5, 6].

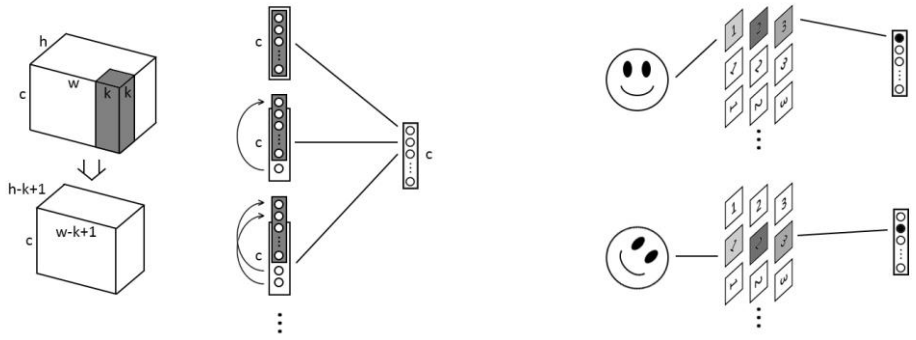
The structure of convolutional neural networks have naturally achieved some level of translate invariance, could this architecture works on rotation invariance? The answer is yes. We design a new layer in deep convolutional neural networks models called cyclic convolutional layer to solve this problem, it is mainly a simple 3-D convolutional layer, between this cyclic convolutional layer and common 3-D convolutional layer are two difference. 1. The kernel size in the 3th dimension is same to feature map size in that dimension. 2. The padding in the 3th dimension is not zeros but the other side of the feature map (Figure 1). And the output is also a 4-D array with 3 dimensions for translate and rotate variance and 1 dimension for different kernels.

This layers can convert rotate variation of a symbol into a translate variation which is easy to deal with common convolution operation, by this, it means that the different orientation will lead to same activation in different position in the feature map of a convolutional neural network. To make this work, every kernel in the front layers should be rotatable. For example assume there are k kernels in layer l, and the rotation resolution is r degree, so there are $n = 360/r$ different orientation, than every kernel in layer l should have n duplicate, each for every orientation. So the different orientation of the symbol will active the same kernel of the different orientation, and the followed cyclic convolutional layer will convert the different orientation into a new dimensions of translate (Figure 2).

One the other hand, the position and orientation of the object is also important information for real world image process tasks, common convolutional neural networks can only recognize the

class of the object and ignore the position and orientation of the object. By add and fully convolutional layer after the cyclic convolutional layer, we can get the position and orientation of the object by a 3-D heat map produce by the last layer of the network.

Last but not the least, human being have an ability to look at an object once and remember it. You look at someone's picture, and you can recognize the person when you meet him, although the picture only include one angle and one position of the person. By use the output of the last but one layer of the network as a feature, we can train a linear classify, which use one sample in one position and one orientation as train data and can recognize it in different positions and orientations. By this we achieve some kind of one shot learning.



(a) (b)

Figure 1: (a) structure of the cyclic convolutional layer, w, h and c stand for the width, height and channels of the feature maps. (b) sketch of an image in different angle to active the different filter of a layer.

2 Previous Work

Scale-Invariant feature is well studied in computer vision, hand craft feature have widely used in many practical applications. D.Lowe's SIFT [1] use key points extracted from images and stored in a database, transforms an image into a large collection of feature vectors, each of which is invariant to image translation, scaling, and rotation. Deformable part models [10] with designed features assumes an object is constructed by its parts, the detector first found a match of its whole, and then using its part models to fine-tune the result. Which is limited to a small set of sub-structure, and destroy information that could be used to whole model object.

Neural network-based architectures recently had great success in significantly advancing the state of the art on challenging image classification and object detection datasets. Robert Gens and Pedro Domingos's Deep Symmetry Networks [8] use symmetry group to describe any form transformation into a set, and use kernel-based interpolation to compute features through a pooled map to achieve rotation invariance. J. Bruna and S. Mallat's Scattering networks [12] are cascades of wavelet decompositions designed to be invariant to demonstrate translation and rotation invariance. Sander Dieleman, Kyle W. Willett and Joni Dambre's Rotation-invariant convolutional neural networks [9] rotate the input images in different angles, than compute different images with the same convolutional filters, the output feature maps of those are

concatenated together, and one or more dense layers are stacked on top to achieve rotation invariance. Well this don's fully use the different rotated filters.

Our approach for neural network-based rotation invariance is to directly rotate the filter of the convolutional neural networks by affine transformation, and stack the filters in the order of rotated angles, and apply new convolutional layer on top of it, so we can use all of the benefit of rotated filters. All the previous work has some operate pool through the rotate dimension, this will lost the orientation information of the object, instead of pooling, we use convolutional to maintain the orientation information of the object.

3 Model

We first train a very simple Lenet-like convolutional neural network use very little samples (only 15 image of 15 different symbols, to avoid the "6" rotate 180 degree and become a "9" problem, all the symbols are not rotationally symmetric, show in figure 2) and rotate the filter of convolutional layers to build the rotation invariance neural network. Than test it with a dataset generated by the training set use random rotation and translation.

Figure 2: Some of the symbols we used

The training process is a greedy per layer way. We first train the network use standard back propagation gradient descent method, than rotate the filter of the first convolutional later, stack all rotated filters and use it as the initial of a new network. We than train the new network and fix the first layer's weights, after the network is converged we rotate the filter of the second convolutional layer and use the first and second filter as the initial of another new network. When there is enough convolutional layers, we put the cyclic convolutional layers between the convolutional layers and fully connected layers, initial it (with an identity matrix in this paper) and add the fully convolutional layers after that, use the same weights of the fully connected layers. Than we can fine-turning the whole network use back propagation. But in our case it is a very simple network with very little samples, so after the last but one step, we already yield a 100% accuracy in the test dataset so we didn't actually do the last step.

Rotate the Filter

The rotate process is a little tricky here, only rotate the filter will lead to the wrong feature of the previous layer because the structure of convolutional neural network. The solution we use here has two part:

1. we arrange the filter by the rotate angles, it mean the first channel of the feature map which produce by the convolution of the first filter and the previous feature map is rotate by 0 degree, the second channel of the feature map which produce by the convolution of the second filter and the previous feature map is rotate by 360/n degree (n is the cyclic numbers, stand for how much different angles we process)… and the last channel of the feature map which produce by the convolution of the last filter and the previous feature map is rotate by 360*(n-1)/n degree.

2. When we rotate the filter, we also cycle the filter channels, it mean we will put the first channel of the filter to the second channel, put the second channel of the filter to the third channel… put the last channel of the filter to the first channel. This will solve the problem. We also fine-turning the network after the rotate.

Background Depress

We don't want the background to active any output of our classifier, so we have to depress the background. The usual way to do this is to add another class for background, but this may lead to unexpected result.

If the input of the network is empty (all zero vector), the reasonable output of the network is also empty (all zeros before the softmax), but it is usually not. How to make this reasonable result happen? The all zero input will make the inner product zero, after subtract the bias, if the result is negative, the output will be zero after the ReLu, so the key to this problem is to have a negative bias. To achieve this, we add some background samples in the train samples and have all zero labels, the derivation operate of the softmax will direct put all negative gradient to all the former layers, it directly minus the bias. As this is a kind of regularization to the network, add too much of the background sample will make the network not converge. The result heat map is shown in figure 3.

4 Experiment

We use 15 32x32 images for training sample to train a network, and use those images to random rotate and translate to generate 1000 64x64 images for test sets, after the rotate filter step, we achieve 100% accuracy rate, so we accomplish the rotate invariance and it also can consider as a way to achieve one shot learning.

Implement detail

The network we use is a lenet-like network, it has two layer of convolution and two layer of fully connection. The first layer has 6 7x7 filter of6 orientation for edge detect, it arranged by the angle. After a pooling layer of size 2, is another convolution layer for 36 14x14 filters, and two fully connected layer for kernel number as 36 and 24. We didn't use the second pooling layer because the pooling may influence the rotation invariance.

We use 12 for orientation resolution, so there are 36*12 = 432 filters for the second convolution layer, after the cyclic convolution layer, there will be a 3-D feature map for size of 16*16*12, and 36 channels. Than after two fully connected layers the final output of the network is a probability map for 15 classes of 16*16*12 pixels. Some of the class's probability map is shown in figure 3.

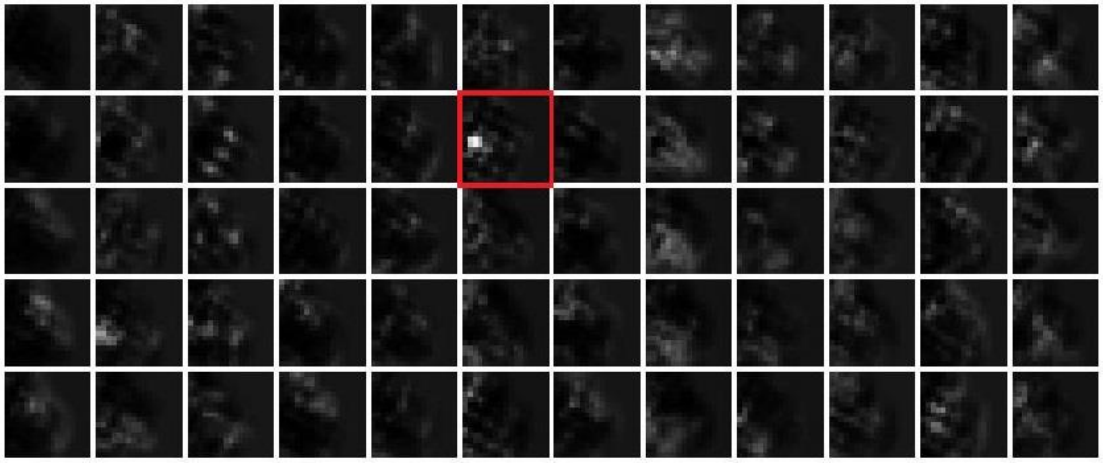


Figure 3: out map of the network, each row for one class, each column for one orientation, each sub picture for 16x16 different position. The red square is the final result, for this example the class of the sample is 2, angle is 150^{o} and the symbol is in the middle left of the picture.

Detection

Because the output of the network is for each class of each angle and each position, it can output all the different symbol in the same picture, which can use as detection when we put a threshold to distinguish if there is a symbol.

The PR curve of different threshold is shown in figure 4. 0.8 is the best threshold in this paper.

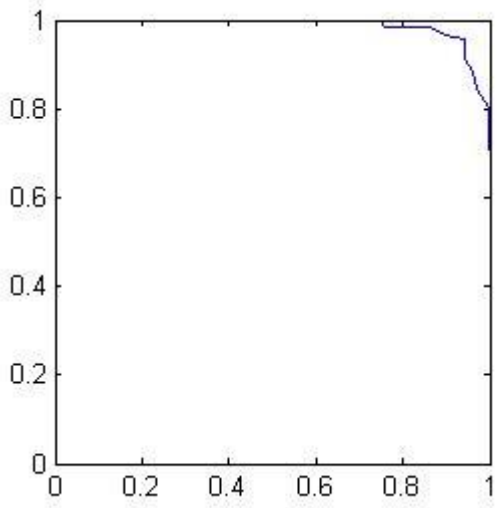


Figure 4 PR curve of detection

One Shot Learning

One Shot Learning Can use very little sample to train a classify to recognize new objects [15, 16]. By use the output of the last but one layer as a feature, we can train a linear classify to recognize new samples and add new classes to the network. The positive training data is the new sample of new classes, with the known position and orientation in the 3-D feature map of the last but one layer, and use features near the right position and orientation as negative training data.

5 Conclusion

In this paper we bring a method to convert rotate variance into translate variance, and apply rotate invariance through convolution operator, and achieve transform invariance in high accuracy rate. There is other kind of variances like scale and skew, it can also covert to translate variance use the same method in this paper, and it will be a 5-D convolution instead of 3-D convolution. In real world there are 6 free dimensions (3 translate dimensions and 3 rotate dimensions). So it is possible to apply to nature pictures use a 6-D cyclic convolutional layer to achieve transform invariance in real world applications.

The method in this paper will make the network more width not more deep, which is not same to the current results in other papers, but shallow network will be fast to compute when the parallel computing resource is abundant, which is almost infinite when computer will be way more cheaper in the future.

References

[1] David G.Lowe. Distinctive Image Features from Scale-Invariant Keypoints. January 5, 2004.

[2] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998

[3] Neal Jean et al. Combining satellite imagery and machine learning to predict poverty. Science : Vol. 353, Issue 6301, pp. 790-794 DOI: 10.1126/science.aaf7894

[4] Super-Resolution on Satellite Imagery using Deep Learning, Part 1

[5] Image processing for accurate cell recognition and count on histologic slides

[6] Josef Madl, Sebastian Rhode, Herbert Stangl A combined optical and atomic force microscope for live cell investigations

[7] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[8] Robert Gens Pedro Domingos. Deep Symmetry Networks. Neural Information Processing Systems 2014.

[9] Sander Dieleman1, Kyle W. Willett2, Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. May 24, 2015.

[10] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[11] G. E. Hinton, A. Krizhevsky, and S. D.Wang. Transforming auto-encoders. In Proceedings of the Twenty-First International Conference on Artificial Neural Networks, 2011.

[12] J. Bruna and S. Mallat. Invariant scattering convolution networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1872–1886, 2013.

[13] D. G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1999.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, 2012.

[15] A.Santoro, S.Bartunov, One-shot Learning with Memory-Augmented Neural Networks. arXiv:1605.06065
[16] BM Lake，R Salakhutdinov. One shot learning of simple visual concepts.