Week4 Pre-Report: Limitation - Spatial Transformer Networks (STN)

Euijin Hong[1]
[1]Department of Electrical and Electronic Engineering, Yonsei University.

1 Introduction and Background

In this paper, we will discuss about the key theory and configuration of Spatial Transformer and Spatial Transformer Networks(STN)[11], and think about the contribution of Spatial Transformer Networks in terms of image recognition. And in this section, we will explore the background of this research and overview the core concepts of the theory. One of the problems of Convolutional Neural Networks(CNN)[2] that existed at that time was the lack of ability to be spatially invariant to the input data, whether in a computational and parameter-efficient manner.

The general method to obtain spatial invariance was to apply pooling mechanism such as introducing local max-pooling layer. However, such method has few disadvantages. One problem is having small spatial support, and this makes spatial invariance only for deep hierarchy of pooling and convolution layers, which is another problem. In other words, intermediate feature maps are actually not invariant to large transformations of the input data[4][13]. These problems occur because of having a pre-defined, limited pooling mechanism for dealing with variations in the spatial arrangement of data.

In order to deal with the problems that pooling mechanisms face, *Spatial Transformer* module is introduced, which can be included into a normal neural network architecture. The strengths of spatial transformer are that its action is conditioned on individual data samples, having the proper behavior learnt during training. The transformer is a dynamic mechanism which can take spatial transform of the image in an active way, by producing an appropriate transform for each inputs. This transformation takes place on the entire feature map, which includes cropping, scaling, rotations, and even non-rigid deformations. By performing this transformation, it allows the network to focus on learning the most relevant region of the image, and the transformed region is set as expected, canonical pose to make following layers easy to recognize. Spatial transformers can be trained by standard back-propagation, thus enables end-to- end training of the model.

CNNs with spatial transformers can be beneficial when performing 1) image classification, by simplifying the subsequent task via output feature map, 2) co-localization, by localizing different instance of the same class in each image, and 3) spatial attention, being more flexible and trained by pure back- propagation, thus increase computational efficiency by giving transformed lower resolution inputs.

2 Related Works

In this section, we will briefly cover the central ideas of the prior works in modeling transformations with neural networks, learning and analysing transformation-invariant representations, with attention and detection mechanisms for feature selection.

In works [10] and [9], assigned canonical frames as a reference of object parts, in which 2D affine transformations were modeled to create a generative model configured of transformed parts.

By estimating linear relationships between representations of original and transformed images, invariance and equivariance of CNN representations to input image has been studied. Several attempts have been made, such as in [4] by analyzing the behavior in relation to symmetry groups, or scattering networks[3], and CNNs that construct filter banks of transformed filters[12] [15]. However, by using spatial transformers, we can manipulate the input data rather than using feature extractors, which was done for clustering in [6], thus achieve invariant representations.

Selective attention-based neural networks manipulate the data via crops, thus can learn translation invariance. In works [1] [14], to avoid requirements of differentiable attention mechanism, trained with reinforcement learning, while [8] utilizes a Gaussian Kernel in a generative model to use a differentiable attention mechanism. In other works such as [7], region proposal algorithm is used as a form of attention, and regress salient regions with a CNN used in [5]. Here, the spatial transformer framework can be considered as a generalization of differentiable attention mechanism within any spatial transformation.

3 Theory and Architecture of Spatial Transformers

In this section we will discover the mechanisms and formulation of the *spatial transformer* model. The model is differentiable that applies a feature map a spatial transformation for a single forward pass, conditioned on a particular input and generating a single output feature map. When it comes to multi-channel inputs, the identical warping is implemented to every channel.

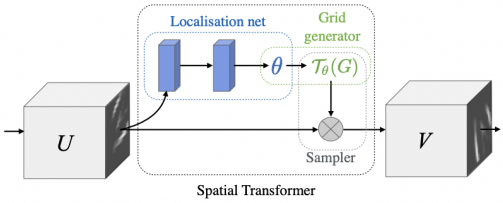

Figure 1: The architecture of spatial transformer module. This module is comprised of localization network, sampling grid, and grid generator. First, the input feature map U passes through the localization network which gives parameters θ. Then, regular space grid G over V is transformed into the sampling grid $\mathcal{T}_\theta(G)$. The grid is then applied to U, thus produces a warped output feature map V.

The mechanism of spatial transformer can be categorized into three parts: which are *localization network*, *sampling grid*, and *grid generator*, which will be further reviewed. The architecture is shown in Figure 1.

3.1 Localization Network

The localization network takes the input feature map with width W, height H, and C channels($U \in \mathbb{R}^{H\times W\times C}$) and outputs θ, the parameters of the transformation $\mathcal{T}_\theta$ are applied to the feature map: $\theta = f_{loc}(U)$. The size of θ varies by the transformation type, and the localization network $f_{loc}(U)$ can take any form, including a final regression layer that produce the output transformation parameters θ.

3.2 Parameterised Sampling Grid

In order to warp the input feature map, every output pixel is computed by using a sampling kernel, refering an element of generic feature map *pixel-wise*. In normal cases, the output pixels locate on a regular grid $G = G_i$ with pixels $G_i = (x_i^t, y_i^t)$, thus forms an output feature map $V \in \mathbb{R}^{H'\times W'\times C}$, in which H' and W' are height and width of the output grid while number of channel remains still as C.

If $\mathcal{T}_\theta$ is a 2D affine transformation, the pointwise transformation is shown below.

$$\begin{pmatrix} x_i^s \\ y_i^s \end{pmatrix} = \mathcal{T}_\theta(G_i) = \mathbf{A}_\theta \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11}\theta_{12}\theta_{13} \\ \theta_{21}\theta_{22}\theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} \tag{1}$$

Here, (x_i^t, y_i^t) are target coordinates of the regular grid in the output feature map, and (x_i^s, y_i^s) are the source coordinates in the input feature map

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.

that define sample points, while $\mathbf{A}_\theta$ is the affine transformation matrix. The transformation allows cropping, translation, rotation, skew and scale, which generates a mapped regular grid that lies in a parallelogram inbound the range of x_i^t, y_i^t.

The transform can have any parameterized form, but has to be differentiable with respect to parameters, which guarantees gradients to be back-propagated through from the sample points $\mathcal{T}_\theta(G_i)$ to the localisation network output θ. This is expected to reduce the complexity of the task when the transformation is parameterised in a structural, low-dimensional way.

3.3 Differentiable Image Sampling

A sampler must take the set of sampling points $\mathcal{T}_\theta(G_i)$, along with the input feature map U and produce the sampled output feature map V, to perform a spatial transformation of the input feature map. In order to get the value at a particular pixel in the output V, sampling kernel is applied to each (x_i^s, y_i^s) coordinates in $\mathcal{T}_\theta(G_i)$ defines the spatial location in the input. The process can be written as the following equation.

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c k(x_i^s - m; \phi_x) k(y_i^s - n; \phi_y) \quad \forall i \in [1...H'W'] \forall c \in [1...C] \quad (2)$$

Generic sampling kernel $k()$ has parameters ϕ_x and ϕ_y that define the image interpolation, while U_{nm}^c is the value at location (n,m) in channel c of the input feature map, and V_i^c is the output value for pixel i at location (x_i^t, y_i^t) in channel c.

Theoretically, any sampling kernels on which the (sub-)gradients can be defined with respect to x_i^s and y_i^s are able to be used. For example, a bilinear sampling kernel can be used, which can be written as follows.

$$V_i^c = \sum_n^H \sum_m^W U_{nm}^c max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \quad (3)$$

In order to allow the backpropagation of the loss through the sampling mechanism, we can define the gradients with respect to U and G. In bilinear sampling, the partial derivatives is as follows.

$$\frac{\partial V_i^c}{\partial U_{nm}^c} = \sum_n^H \sum_m^W max(0, 1 - |x_i^s - m|) max(0, 1 - |y_i^s - n|) \quad (4)$$

$$\frac{\partial V_i^c}{\partial x_i^s} = \sum_n^H \sum_m^W U_{nm}^c max(0, 1 - |y_i^s - n|) \begin{cases} 0 & if |m - x_i^s| \geq 1 \\ 1 & if m \geq x_i^s \\ -1 & if m < x_i^s \end{cases} \quad (5)$$

Equation (5) can be applied similarly for $\frac{\partial V_i^c}{\partial y_i^s}$. This shows that differentiable sampling mechanism allows loss gradients to flow back not only to the input feature map(4) but also to the sampling grid coordinates(5), therefore back to the transformation parameters θ and localisation network because $\frac{\partial x_i^s}{\partial \theta}$ which can easily be derived. The sampling mechanism allows us to ignore the sum over all locations and look at the kernel support region for each output pixel.

3.4 Spatial Transformer Networks (STN)

The spatial transformer module consisting localization network, sampling grid, and grid generator, can be inserted into a CNN architecture at any point and in any number, which makes *spatial transformer networks*. Inserting the module does not impede the speed, and even leads to faster speed due to subsequent downsampling. Therefore, STN can actively learn how to transform the feature maps to help minimize the overall cost function of the network while training. In the localization network, weights of it contains information of how to transform each training sample. In some tasks, feeding the output θ forward can be useful, because it encodes the transformation of a region or object. Using spatial transformers to downsample or oversample a feature map is also possible, and we can have multiple transformers in a CNN, which allows transformation of increasingly abstract representations and gives localization network more informative information to base the predicted transformation parameters on. Parallel spatial transformers can also be implemented, especially when multiple objects are of interest in an output feature map.

4 Evaluation

In order to evaluate the performance of STN, researchers performed several experiments in several supervised learning tasks. In this section, we will briefly review the process and results of each experiment.

First, the team used distorted MNIST data to explore the range of transformation that a network can learn invariance via spatial transformer. Researchers measured the efficiency of STN by comparing the baseline neural networks (FCN and CNN) and STNs (ST-FCN and ST-CNN), where all STNs used bilinear sampling and variants are distorted differently. The result shows that STN outperforms its counterpart base network, for every type of distortion, shown in Figure 2-(left).

Then, researchers tested the network on a Street View House Numbers (SVHN) dataset, by comapring the baseline CNN model comprised of 11 layers and 5 softmax classifiers, predicting the digit at a particular position in the sequence. Then, they performed an extension by putting a spatial transformer in the baseline CNN right after the input (ST-CNN single), where localization network is a 4-layered CNN. Also, they defined another extension where spatial transformers are inserted for every first four conv. layers of the baseline CNN, where localization networks are all 2-layered FCN with 32 units per layer, which thus allows to predict a transformation based on richer features than raw image. The result of this experiment is shown in Figure 2-(mid), where the spatial transformer models obtain the state-of-art performance, reaching minimum error as 3.6%.

Finally, the team used STN with multiple transformers connected parallel to perform fine-grained bird classification, especially in the CUB-200-2011 birds dataset[16], using only image class labels for training. They considered to have a strong baseline CNN model that already achieves state-of-the-art accuracy by 82.3%. Then, they trained ST-CNN which contains 2 or 4 parallel spatial transformers, parameterised for attention and acting on the input image. The transformers captured discriminative image parts, which are passed to the part description sub-nets. The resulting part representations are then concatenated and classified with a single softmax layer, while the model is trained on image class labels by backpropagation, end-to-end. The results are shown in Figure 2-(right), which achieves the accuracy of 84.1%, exceeding that of the baseline by 1.8%. In the visualization of the transforms predicted, the spatial transformer seems to split their jobs to predict different features of an object, without additional supervision. Also, by using spatial transformers, the downsampled output image can be obtained when using 448px resolution input images.

Model		MNIST Distortion R	RTS	P	E
FCN		2.1	5.2	3.1	3.2
CNN		1.2	0.8	1.5	1.4
ST-FCN	Aff	1.2	0.8	1.5	2.7
	Proj	1.3	0.9	1.4	2.6
	TPS	1.1	0.8	1.4	2.4
ST-CNN	Aff	0.7	0.5	0.8	1.2
	Proj	0.8	0.6	0.8	1.3
	TPS	0.7	0.5	0.8	1.1

Model		Size 64px	128px
Maxout CNN [13]		4.0	-
CNN (ours)		4.0	5.6
DRAM* [1]		3.9	4.5
ST-CNN	Single	3.7	**3.9**
	Multi	**3.6**	**3.9**

Model		
Cimpoi '15 [5]		66.7
Zhang '14 [40]		74.9
Branson '14 [3]		75.7
Lin '15 [23]		80.9
Simon '15 [30]		81.0
CNN (ours)	224px	82.3
2×ST-CNN	224px	83.1
2×ST-CNN	448px	83.9
4×ST-CNN	448px	**84.1**

Figure 2: (left): Test result for MNIST data, (mid): Test result for SVHN, (right): Test result for fine-grained bird classification

5 Conclusion

By introducing a neural network module that internally contains the spatial transformer, the module could perform explicit spatial transformations of features, which opened up new ways for modeling data, with end-to-end learning and without making changes to the loss function. Although CNN alone provides a strong baseline, inserting spatial transformers in the module could improve accuracy, thus gaining state-of-the-art performance. Plus, the regressed transformation parameters are available as an output and could be used for subsequent tasks.

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple object recognition with visual attention. *arXiv preprint arXiv:1412.7755*, 2014.

[2] Jake Bouvrie. Notes on convolutional neural networks. 2006.

[3] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[4] Taco S Cohen and Max Welling. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659*, 2014.

[5] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014.

[6] Brendan J Frey and Nebojsa Jojic. Fast, large-scale transformation-invariant clustering. *Advances in neural information processing systems*, 14, 2001.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[8] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International conference on machine learning*, pages 1462–1471. PMLR, 2015.

[9] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pages 44–51. Springer, 2011.

[10] Geoffrey F Hinton. A parallel computation that assigns canonical object-based frames of reference. In *Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2*, pages 683–685, 1981.

[11] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[12] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. Locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1412.5104*, 2014.

[13] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 991–999, 2015.

[14] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[15] Kihyuk Sohn and Honglak Lee. Learning invariant representations with local transformations. *arXiv preprint arXiv:1206.6418*, 2012.

[16] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.