An Improved Spatial Transformer Network based on Lightweight Localization Net and its Application to Computer Vision

Tzu-Han Hsu
dept. Computer Science
National Yang Ming Chiao Tung University
Hsinchu, Taiwan
hankshyu@gmail.com

Abstract

Spatial transformer network (STN) is a powerful module that improves the spatial invariance of convolutional neural networks. Amid all components, the localization net serves as the backbone as it intakes the feature map and generates the affine transformation parameters, deciding how the transformation to be operated. Meanwhile, the design of the localization net is closely correlated to the outcome of transformation consequently affecting the system performance. This work simulates through different localization net designs and examines them with several prominent models on various datasets, and which aims to discover a modern localization network architecture that boosts the performance of STN transformation with less parameters and computational overhead.

1. Introduction

A series of high-performance convolutional neural networks have been recognized as the state-of-the-art machine learning approaches since AlexNet [1] won the ImageNet Challenge: ILSVRC 2012 [2]. This technology has been successfully applied to address a broad range of machine learning tasks, such as object detection, tracking [3, 4], super-resolution [5, 6] and document analysis [7]. Although they were introduced roughly in the 90s [8], it was not until the development of computational hardware along with the improvement of model architectures in this decade that enabled a smooth training of deep neural networks. Meanwhile, the pioneering LeNet5 [9] introduced a 5 layered structure in 1998, but nowadays deep neural networks (DNN) have reached a hundred layers or more with different performances through more specialized design, such as Residual Networks [10] or Highway Networks [11]. In addition, many advanced works focus on finer interconnection methods [12], renewed convolution procedures to reduce the computational overhead [13, 14] or specialized layers targeting a subset of training problems. These approaches increase the model performance based on elaborated mechanisms to gather potential explainable information from the feature map and learn more detail from them. However, how to effectively transform the input data by the model itself for the purpose of attaining a better performance draws few concerns. Moreover, applying the attention mechanism to the input image is one of popular studies in the deep learning, figuring out where the model should put emphasis on; howbeit, it was not until recently that the relevant works start to design the corresponding model performance. Specialized layers are designed to transform the input image before the image was sent into the main convolutional layers. However, some of the components in such layers are out of date, thereby could not reflect the improvements in convolutional architectural by modern research works. Thus, this work aims to design a renewed structure of such layers to improve its capabilities, and then evaluates such re-designed structure on various models to validate its improvements.

My sincere thanks go to Professor Hero S.J Lee
for guiding this work.

2. Related Work

Convolutional layers are feature extractors that scan through each pixel with dot product operations. They mechanism is very effective despite the fact that such operation is not spatially invariant. Features rotated, flipped or scaled may not be captured by the convolution layers thus lead to a false recognition. The introduction of pooling layers is capable of spatially transforming a feature map, but the effect is still limited nevertheless due to the prevalent small pooling size. On the other hand, it is customary to train the model with transforming conducted images, a technique performed during the data preprocessing stage. Reproducing training images by cropping, rotating or focusing the input image to enlarge the dataset and train the model on them, but this could lead to potential hazards: techniques like so usually work by hand-crafted rules and execute according to the choices of designers. Designers are limited to come up with details such as the rotational angle or the cropping area of the raw image. The process of reproducing new images from raw images inflates the size of the dataset, further prolongs the training time. When the dataset is very large, such measure comes with a heavy cost.

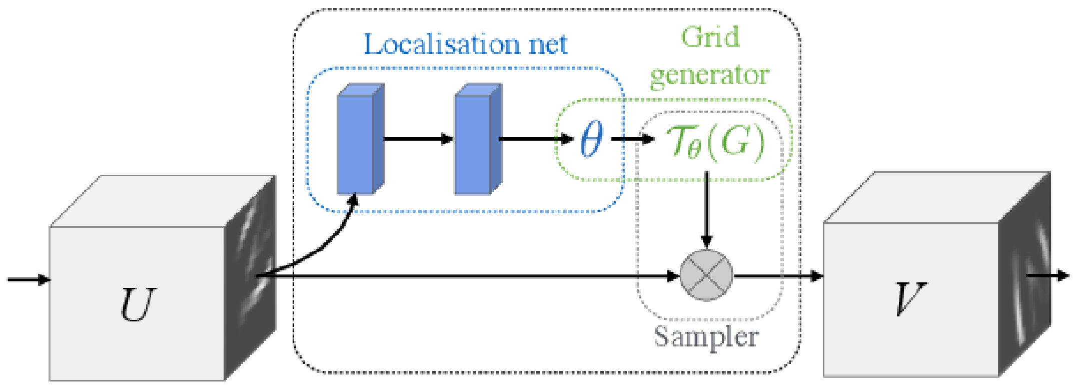

Figure 1: The Architecture of a Spatial Transformer Module [15]

A. Spatial Transformer Network

In order to obtain models with the ability to disentangle the pose and size of object, the best way is to incorporate attention mechanism into the model. Spatial transformer networks [15] designed a whole new architecture targeting the work. In figure 1 shows that the architecture has three main components: the localization network, the grid generator and the sampler. The localization network inputs feature map directly and outputs predicted transformation parameters to the gird generator, who creates sample grids. Sample grids are set of points marking where the input feature map should be sampled. The sampler takes the two outputs as input and produces the output feature map, a well transformed image.

B. VGG and Inception Net

The mechanism of the transformer network heavily depends on the localization network, how well the localization network understands the input feature map directly affects the quality of its output, affine transform parameters. The developments in convolution network architectures are highly related to premium localization network designs. After the success of AlexNet, VGG [16] put forward the concept of unified sized convolutional layers, pooling layers and strides, and proved Local Response Normalization useless. The largest VGG structure has 19 layers, with 138 million parameters. Neural network models tend to perform better with deeper or wider architectural design, so Inception Net [17] designed a wider model, with different size of convolutions to extract different features from the feature map. However, such design would lead to explosive increase of parameters, so 1 x 1 kernel are added to reduce the number of channels.

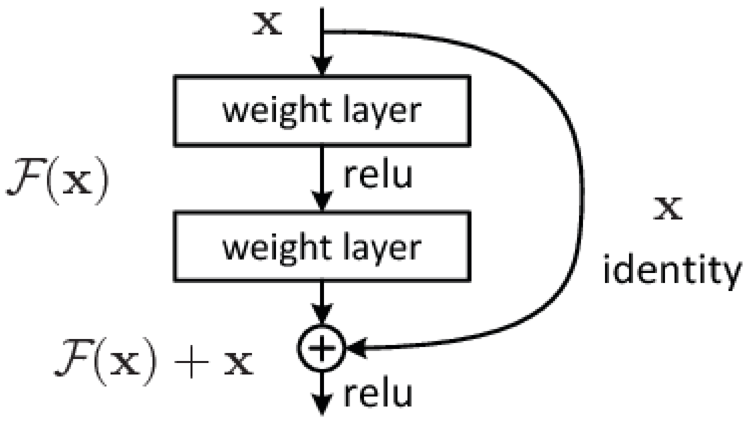
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2: Residual Learning: A Building Block [10]

C. Skip Connections

As the model layers grow, the accuracy soon saturates and then degrades rapidly. Residual neural network (ResNet) [10] popularized the use of skip connections, a direct path for latter layers to access feature maps from previous layers. Identity could easily be learnt from the latter layers to prevent disruption of the former features, albeit learning anything new. Such design opened the door for training neural network over 100 layers without concerning the overfitting problem. Figure 2 describes a residual learning block Networks like DenseNet [12] is also based on skip connections but with denser interconnect in the convolution block targeting better performance. Some network architectures [18, 19] further incorporate skip connections into their original structure.

D. Lightweight Models

Some researchers are interested in finding a more efficient convolution model. MobileNet [14] decomposes the convolutional operation into depth-wise and point-wise convolution, and it greatly reduces the parameters and computational cost. ShuffleNet [20] adopts point wise group convolution and channel shuffle whereas SqueezeNet [21] reduces parameter by designing a fire model, which is somewhat similar to a ResNet structure. Although these models may not reach state-of-the art performance in benchmarks, they endeavor to make concession between performance and computational overhead.

E. EfficientNet

EfficientNet [22] tries to balance network depth, width and resolution under a certain computational budget. Convolutional models often focus on scaling only one of the three dimensions with an arbitrary scale without further investigation. [22] Suggests a simple but effective compound scaling method which uniformly scales three dimensions. Furthermore, they designed a baseline network through neural architecture search [23] then compound scale to an entire family of network called EfficientNet. The baseline network is based on the inverted bottleneck residual blocks coming from MobileNetV2 [18] in addition to squeeze-and-excitation blocks [24] and Swish activation function [25].

3. Localization Net Architecture

Affine transformation on 2D space requires 6 parameters which are capable of resizing, rotating, displacing and shearing images. According to [15] and Figure 1, the localization net inputs the feature map U and output θ, and the parameters of the transformation are applied to the feature map. Meanwhile, the output θ is a 6-element array carrying the affine transformation specifications. To validate the performance of the proposed work, this work will be further simulated with different localization designs. Also, the localization net is capable of being trained through standard back-propagation, so the network architectures like convolutions or fully connected layers are fully supported. In addition, all classification layers of the localization net will be the same in the simulation experiment. Assume that the last hidden layer outputs feature map with height H, width W and Channels C, the feature map shall be flattened to one dimension of length $\mathrm{H} \times \mathrm{W} \times \mathrm{C}$ in advance and connects with two fully connected layers. One with $\mathrm{H} \times \mathrm{W} \times \mathrm{C}$ as input and 32 as output, another with 32 as input and 6 (for affine transformation) as output. ReLU [27] is selected as the activation function since it avoids vanishing gradient problem with lightweight cost of computational power. Hence, the classification layers of localization net would have $(\mathrm{H} \times \mathrm{W} \times \mathrm{C}) \times 32 + 192$ parameters in total.

A. Regular STN Structure (STN)

Most commonly used spatial transform network (STN) localization architecture is made out of two wide convolution layers, one 7 x 7 in the front followed by a 5 x 5 and with 2 x 2 max pooling layers interleaving between. Pooling layers contains no parameters and are of a certain degree spatial invariant to the position of features, despite the fact that they shrink the input size quickly, dropping gathered information brutally. Such composition resembles the LeNet5 architecture [9]. Further, Table 1 shows the regular localization architecture.

Table 1: Regular STN Localization Architecture

Type	Filter Shape	Output shape	Parameters
Conv	7 x 7 x 3 x 8	218 x 218 x 8	1184
Max pool, stride 2	2 x 2 x 8 x 8	109 x 109 x 8	0
Conv	5 x 5 x 8 x 10	105 x 105 x 10	2010
Max pool, stride 2	2 x 2 x 10 x 10	52 x 52 x 10	0
Total			3194

B. VGG-like Structure (V-STN)

Breaking down large size convolution filers into stacked small size convolutions are one of the main contributions of VGG [16]. In theory, two 3 x 3 convolutions would share the same receptive field as one 5 x 5 convolution as well as three stacked 3 x 3 convolutions could replace a wide 7 x 7 convolution layer. Whereas three 3 x 3 convolutions take 27 parameters but a 7 x 7 takes 49, over 40 percent of reduction. Smaller convolutions are also beneficial for increasing model linearity, becoming the mainstream of convolution layer architecture. Similar changes could be made to the STN localization architecture, by replacing the first 7 x 7 wide convolution layer into three stacked 3 x 3 convolution layers and the second 5 x 5 wide convolution would turn into two 3 x 3 convolution layers at the meantime. Max pooling layers remains unchanged. Further, Table 2 shows the VGG-like localization architecture.

Table 2: VGG - like Localization Architecture (V-STN)

Type	Filter Shape	Output shape	Parameters
Conv	3 x 3 x 3 x 8	222 x 222 x 8	224
Conv	3 x 3 x 8 x 8	220 x 220 x 8	584
Conv	3 x 3 x 8 x 8	218 x 218 x 8	584
Max pool, stride 2	2 x 2 x 8 x 8	109 x 109 x 8	0
Conv	3 x 3 x 8 x 10	107 x 107 x 10	730
Conv	3 x 3 x 10 x 10	105 x 105 x 10	910
Max pool, stride 2	2 x 2 x 10 x 10	52 x 52 x 10	0
Total			3032

C. The Proposed Network - An Improved Spatial Transformer Network based on Lightweight Localization Net (L-STN)

Convolution operation comes with heavy overheads, scalar product consumes a large portion of computational resources in the CNN network. According to Amdahl’s law, even by optimizing the convolution operation by an inch would lead to significant improvements. Convolution layers are three dimensional (width × length × height), replacing large convolution with smaller sizes reduces the two dimensions in width and length, which reduces computational cost and parameter counts by the factor of N^2. If there’s a way to further shrink the height dimension, it could also reduce the operation by a factor of N. Meanwhile, Inception Net [17] popularizes the use of 1 x 1 kernels between channels to increase or decrease the feature map count, concentrating the useful ones and discard the wastes. There are two types of 1 x 1 kernels that often come in pairs: the encoder and the decoder. The encoder kernel reduces the number of the feature map, in other words cherry pick the informative ones from different feature maps. The decoder kernel expands the compressed information and increases the count of feature maps. With a combination of an encoder and decoder, output channels could be reduced to save computational cost and parameter count. Also, Batch Normalization [26] is very popular in modern CNN Networks. It reduces the covariate shift so fewer exploding gradients would happen. Furthermore, the training will also be less affected by the initialization process. Batch Normalization is very computational efficient and effective, they could be place right after every convolution layer.

In order to realize the structure of the proposed network, Table 3 shows a localization architecture build by the above-mentioned structures (An Improved Spatial Transformer Network based on Lightweight Localization Net, L-STN). This work applies 1 x 1 kernels to concentrate channels and to reduce parameters. Batch normalizations were added after the convolution and after the activation function. 3 x 3 convolutional kernels with larger strides replacing the pooling layers, which leads to more parameter counts in exchange of versatility. Instead of constantly reducing the size of the feature map, some identity layers were introduced, which includes padding in the convolution operation. Such layers do nothing to shrink the feature into our target size, but focus on extracting spatial information from the feature map.

Although the parameters in this architecture lead to more parameters, the concentration mechanism would lead to better performances and with less computational overhead. The greatest reduction of parameters comes from the stem part of the network. Reducing the parameter by half via adopting a smaller output channel and expand the channel for the next convolutional layer by a decoder. Other layers remain the same I/O size but encoders and decoders were also added in between to concentrate the feature maps.

Table 3: The Proposed Architecture (L-STN)

Type	Filter Shape	Output shape	Parameters
Conv, stride 2 + BN	3 x 3 x 3 x 4	112 x 112 x 4	112+8
Conv + BN	1 x 1 x 4 x 8	112 x 112 x 8	40+16
Conv + BN	3 x 3 x 8 x 8	110 x 110 x 8	584+16
Conv + BN	1 x 1 x 8 x 4	110 x 110 x 4	36+16
Conv + BN	1 x 1 x 4 x 8	110 x 110 x 8	40+16
Conv + BN	3 x 3 x 8 x 8	108 x 108 x 8	584+16
Conv + BN	1 x 1 x 8 x 4	108 x 108 x 4	36+16
Conv + BN	1 x 1 x 4 x 8	108 x 108 x 8	40 + 16
Conv, stride 2 + BN	3 x 3 x 8 x 8	54 x 54 x 8	584+16
Conv + BN	1 x 1 x 8 x 4	54 x 54 x 4	36+8
Conv + BN	1 x 1 x 4 x 8	54 x 54 x 8	40 + 16
Conv, padding 1	3 x 3 x 8 x 8	54 x 54 x 8	584+16
Conv + BN	1 x 1 x 8 x 4	54 x 54 x 4	36+8
Conv + BN	1 x 1 x 4 x 8	54 x 54 x 8	40 + 16
Conv + BN	3 x 3 x 8 x 8	52 x 52 x 8	584+16
Conv + BN	1 x 1 x 8 x 4	52 x 52 x 4	36+8
Conv + BN	1 x 1 x 4 x 10	52 x 52 x10	50+20
Total			3690

4. Simulation Experiments

4.1 Datasets Description

A. Chinese Traffic Sign Database

The original dataset TSRD includes 6164 traffic sign images with 58 categories, they are collected by National Nature Science Foundation of China (NSFC) to serve as an open traffic sign recognition dataset. A test file containing 4170 images and a test set with 1994 images are open for download. Some duplicated images have been removed and readjustments have been made in this work. 4000 data were kept as training set, 1000 as testing set and the remaining 998 images are left as validation set.

B. Caltech 101

The dataset [28] is created by Fei-Fei Li, Marco Andreetto, Marc 'Aurelio Ranzato and Pietro Perona at the California Institute of Technology for computer vision experiments. Pictures of objects are classified into 101 classes, most category has about 50 images with resolution of about 300 x 200. The dataset has in total 8677 images, about 75% (6277) were kept as training set while other images were split evenly as validation (1200) and testing dataset (1200) .

C. Cifar-10

The dataset [29] was collected by Canadian Institute for Advanced Research and is the subset of the 80 million tiny images dataset. The dataset is a collection of 60000 low resolution images (32x32) classified into 10 classes. Each class has 6000 images. Due to the fact that all classes have a uniform distribution of images, data augmentation may not be critically necessary. 40000 is left as training set and the rest are split evenly as validation set (10000) and testing set (10000) in this work.

4.2 Training Procedure

In accordance with the above dataset, the training procedure of this work can be described in detail as follows:

1) The datasets are first scaled into the input size of the model using bilinear interpolation, then normalized (By assuming each pixel are even distributed between 0 and 1). Other data preprocessing methods like data augmentation are not performed. It is pretty obvious that by performing data preprocessing would lead to better performing models; nevertheless, the main goal is to test different model architectures, it's fair since the same dataset is used to train and test our models, so techniques alike are omitted.
2) The batch size is set to 20 for smaller models and 10 or smaller for larger models. Since this work adopt stochastic gradient descent, batch size does affect the model training process but not much to the final result. Setting large batch sizes leads to faster training speed yet smaller batch size saves memory space. (VRAM if hardware acceleration is turned on). In this work, larger training epochs are chosen for models to converge in order to minimize the effect of different batch sizes.
3) Loss function are set as cross-entropy, with loss averaged over each loss element in the batch and a weighted mean of the output is taken as the reduction to apply to the output. No smoothing measures were taken. Adam [30] is chosen as the optimizer, which is a combination of the adaptive gradient and the momentum optimizer. The learning initial learning rate (η) is 0.001, β_1 is 0.9 and β_2 is 0.999. objective and weight decay are set to 0.

4) Early stopping could reduce unnecessary training resources; furthermore, it helps with overfitting. Stopping conditions should be carefully set, not to stop the training before the model has converged thus a tolerance parameter is set up to judge whether the early stopping should take place. If the oscillation of the variation accuracy of a continuous 10 epochs does not exceed the tolerance parameter plus the accuracy is no longer raising, suggesting the model has already diverged, may then stop the training procedure. Note that a training epoch floor could be set in order to avoid the model from stop training at the start up stage.
5) All experiments are conducted on a workstation equipped with 8 cores 16 threads Intel Core i9-9900KF CPU with 3.6 GHz clock rate and could reach 5 GHz with Intel Turbo Boost Technology 2.0. Along with 32 GB DDR4 memory running at 2400 MHz A graphics card is installed for hardware acceleration: Nvidia TITAN RTX, built with Nvidia Turing architecture. It has 4608 CUDA cores running 1350 MHz base clock, 1770 MHz boost clock, also equipped with 24 GB GDDR6 memory running 14 Gbps memory data rate.

4.3 Classification Results

A. Chinese Traffic Sign Database

Chinese traffic sign dataset is modest in size. Due to the fact that most of the traffic signs are installed according to local regulations, a large number of images are of the same shape with slight rotations. Hence, the usage of spatial transform network might be as effective. From Table 4, the accuracy of all models is well above 98%, showing how well modern CNN architecture could deal with such real time tasks.

Table 4: TSRD Classification Cost and Results

Model	Parameters	GFLOPS	Training Error	Validation Error
AlexNet	57,044,810	0.7106	96.69	98.10
EfficientNet-b0	14,047,366	1.7299	99.50	99.40
EfficientNet-b1	27,235,236	3.5579	100.00	100.00
EfficientNet-b2	29,487,358	4.8370	99.90	99.30
EfficientNet-b3	40,648,506	8.2597	99.70	99.70
ResNet-18	11,181,642	1.8185	99.20	99.20
ResNet-34	21,289,802	3.6707	99.70	99.80
ResNet-50	23,528,522	4.1094	99.30	99.40
ResNet-101	42,520,650	7.8320	99.70	99.80
ResNet-150	58,164,298	11.5567	99.70	99.80
DenseNet-121	6,964,106	2.8647	99.90	99.90
DenseNet-120	12,501,130	3.3964	99.90	99.90
DenseNet-119	18,112,138	4.3391	99.80	99.90
DenseNet-118	30,675,594	5.8172	99.80	99.90

It is also noticeable that not model with the most parameters perform the best. The excessive parameters may cause overfitting to the training thus further harm the accuracy. That's when the residual structure comes in handy, by introducing feature maps form the previous layers, identity layers could be learnt by the layer that are close to the output layers to prevent model accuracy from falling when the overall parameters become large. In reality, models with appropriate size shall be chosen even if residual layers were added. If identity matrix were learnt at the last few layers, it will be better to remove them to levy the computational burden. And identify might not be learnt properly at the layers that are excessive. The chart also shows that some smaller models even have higher accuracy, due to the noise coming from redundant layers of a larger size model. The TSRD could be dealt with most of the modern CNN architectures pretty well, even with the obsolescent AlexNet could reach a high accuracy of 98.1%. If the CNN models could already deal with the datasets very well, it would be nice if the STN structure was kept from the model to save parameters and computational overhead.

B. Caltech 101

Caltech 101 with 101 classes of objects is in a different class if compared to TSRD. The objects with distinct shapes in all types of angels are nothing like traffic signs that approximately share a unified contour. That's when the spatial transformer network could take advantage to readjust the input image size further perform automatic rotations to the input image. This work would put more emphasis on the EfficientNet structure and add different spatial transform networks on top of them.

Table 5: Parameters & GFLOPS of Different Models on Caltech 101

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
	Parameters	GFLOPS	Parameters	GFLOPS	Parameters	GFLOPS	Parameters	GFLOPS
EfficientNet-b0	14,163,937	1.730	15,032,641	1.809	15,032,479	1.816	15,033,137	1.756
EfficientNet-b1	27,357,540	3.558	28,364,484	3.650	28,364,322	3.658	28,364,980	3.588
EfficientNet-b2	29,615,577	4.837	30,809,721	4.945	30,809,559	4.954	30,810,217	4.872
EfficientNet-b3	40,789,556	8.260	42,406,100	8.405	42,405,938	8.417	42,406,596	8.307
EfficientNet-b4	56,613,262	18.432	59,266,606	18.668	59,266,444	18.687	59,267,102	18.508
EfficientNet-b5	95,571,391	41.653	99,446,815	41.994	99,446,653	42.021	99,447,311	41.763
ResNet-50	23,714,981	4.110	24,583,685	4.189	24,583,523	4.196	24,584,181	4.135

Table 6: Validation and Test Accuracy of Different Models on Caltech 101

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
	Top-5 val.	Top-5 test	Top-5 val.	Top-5 test	Top-5 val.	Top-5 test	Top-5 val.	Top-5 test
EfficientNet-b0	73.33	72.33	75.67	73.92	71.83	74.25	74.00	74.83

Table 6: Validation and Test Accuracy of Different Models on Caltech 101

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
EfficientNet-b1	72.83	71.58	73.33	71.92	71.42	72.25	72.00	73.00
EfficientNet-b2	66.83	66.83	68.50	68.42	69.58	68.50	73.17	75.00
EfficientNet-b3	63.33	60.83	69.83	69.83	74.00	73.67	75.58	75.42
EfficientNet-b4	68.67	69.25	69.42	71.83	73.25	72.08	73.92	74.83
EfficientNet-b5	70.00	68.75	72.75	71.83	73.25	72.08	79.83	78.33
ResNet-50	75.17	73.58	76.42	76.92	78.75	77.00	77.42	77.33

From Table 5, ResNet-50 has parameters and FLOPS close to EfficientNet-b1 and EfficientNet-b2. Without the spatial transform networks, the accuracy of the EfficientNet models are inferior to ResNet. EfficientNet takes the measure of compound scaling to enlarge the entire model thus leading to large input size when the model expands. FLOPS expand quickly as when the input resolution increases. The EfficientNet-b5 has an input size of 456 x 456, 4 times the size of the ordinary 224 x 224 resolution, which leads to a scale of 2^4 times FLOPS to the model. Even with such an expensive overhead, it performs even worse compared to the cheap ResNet-50 model.

After adding regular STN network to the stem network, ahead of all model structures. From Table 6, the accuracy of the models increases. No single model performs worse after adding the spatial transformer network. Most models have 1% to 3% accuracy growth and the EfficientNet-b3 model increased its accuracy over 8 percent, from 60.83% to 69.25 %. The experiment demonstrates that models do benefit from rotating and resizing the input image in advance. By adjusting the input images to unify what is fed into the network, the hidden layers could do a better job discovering the features in the images and extracting them using the convolution operation. In other words, the network could focus on extracting features to do the classification, thus increases the test accuracy. Examples like EfficientNet-b3 may result from the original model have difficulties recognizing target images with diverge resolutions and spatial information.

By replacing the regular STN structure by the VGG-like structure (V-STN). All models decrease parameter counts but increase in FLOPS operations. Thanks to splitting multiple large convolutional layers to smaller ones, some parameters could be saved. The overall structure and overhead are very similar to the regular STN structure, in spite that accuracy further improved. All models performed better with V-STN than with regular STN structure, EfficientNet-b3 with 4.42% increase in performance. Other models increase by a bit, but not as much, this is because from changing from STN to V-STN, the overall architecture did not change much, which leads to only slight performance leap.

It is a completely different story when models are equipped with the proposed L-STN structure. Owing the fact that L-STN is imitating an entire CNN architecture at the localization architecture, seems to provide more accurate information about how to perform the affine transform. The accuracy of all models was boosted even more, models like EfficientNet-b3 and EfficientNet-b5 even boosted their accuracy over 6%. After adding L-STN in prior to the main CNN structure, EfficientNet-b5 finally outperforms the ResNet-50 model to become the best performing model.

C. Cifar-10

Will the improvements via adding spatial transformer network on a larger scale dataset still remain? This work applies similar setups on Cifar-10 and from Table 8 could illustrate that all of the models has higher accuracy when spatial transform layers were added. The greatest performance boost comes from EfficientNet-b2, a 0.92% accuracy advance. However, the accuracy scores are so close that does not make much of a difference.

By replacing regular STN layers with V-STN layers, similar results were also observed. All models once again outperform models with STN but only with minor performance boosts. 3 out of 7 models increased their performance lower than 0.1% and no model has a performance leap over 0.5%. When the localization architecture switched to L-STN, a much greater performance leap could be observed. Some models even performed 1% better overall on the testing set. The improvements of models equipped with L-STN could be 1 to 1.5 % higher than model without such architecture, which is a noticeable difference.

From the tables below strongly suggests that the model's accuracy before and after adding the spatial transformer layer are strongly related. For example, DenseNet-121 model performs the best among all models without STN layers. With whatever STN layers added, the model still tops all performances. Some models have powerful classification capabilities but perform poorly without the spatial transformer layers, this is due to the input images are difficult for the model to learn. By transforming each image by a specialized network could it standardize the shape of the input figure, allowing the neural network to learn more effortlessly, reaching its full potential.

Table7: Parameters & GFLOPS of Different models on Cifar-10

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
	Parameters	**GFLOPS**	**Parameters**	**GFLOPS**	**Parameters**	**GFLOPS**	**Parameters**	**GFLOPS**
EfficientNet-b0	14,047,366	1.7299	14,916,070	1.8092	14,915,908	1.8162	14,916,566	1.7556
EfficientNet-b1	27,235,236	3.5579	28,242,180	3.6494	28,242,018	3.6574	28,242,676	3.5876
EfficientNet-b2	29,487,358	4.8370	30,681,502	4.9450	30,681,340	4.9542	30,681,998	4.8720
EfficientNet-b3	40,648,506	8.2597	42,265,050	8.4046	42,264,888	8.4167	42,265,546	8.3064
ResNet-18	11,181,642	1.8185	12,050,346	1.8979	12,050,184	1.9049	12,050,842	1.8443
ResNet-50	23,528,522	4.1094	24,397,226	4.1888	24,397,064	4.1958	24,397,722	4.1352
ResNet-152	58,164,298	11.5567	59,033,002	11.6361	59,032,840	11.6431	59,033,498	11.5825
DenseNet-121	6,964,106	2.8647	7,832,810	2.9440	7,832,648	2.9510	7,833,306	2.8904

Table 8: Validation and Test Accuracy of Different models on Cifar-10

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
	Top-5 val.	**Top-5 test**	**Top-5 val.**	**Top-5 test**	**Top-5 val.**	**Top-5 test**	**Top-5 val.**	**Top-5 test**
EfficientNet-b0	86.83	86.05	87.48	86.40	87.88	86.81	87.71	87.45

Table 8: Validation and Test Accuracy of Different models on Cifar-10

Model	Original		STN		V-STN		The Proposed Structure, L-STN	
EfficientNet-b1	86.90	86.89	87.01	86.95	87.62	86.98	87.71	87.03
EfficientNet-b2	86.65	86.38	87.81	87.30	87.57	87.36	87.77	87.52
EfficientNet-b3	87.11	86.79	87.12	86.96	87.19	87.02	87.64	87.22
ResNet-18	85.60	84.96	85.44	85.53	85.79	85.83	86.37	86.41
ResNet-50	85.07	84.84	85.43	85.01	85.63	85.02	86.69	86.19
ResNet-152	85.15	84.65	85.09	84.81	85.40	85.16	86.86	85.17
DenseNet-121	88.91	88.61	89.42	89.29	89.88	89.51	86.17	90.28

D. Computational Cost of STN Architectures

This work is also interested in how much resource it takes for the spatial transformer network to apply. The number of parameters and the FLOPS takes to compute them are in Table 9. The size of the layer is proportional to the input resolution, thus models with identical input resolution would share the same spatial transformer structure. It is apparent that V-STN brings lower parameter counts but with slightly more FLOPS than the regular STN architecture. L-STN has slightly more parameters but the FLOPS it takes to compute is almost one third. This resulted from the widely use of decoder and encoder structure. EfficientNet adopts compound scaling method, so the input resolution as well as the spatial transformer structure grows as the model scales.

Table 9: Parameters of GFLOPS of Different Spatial Transformer Architecture

Model	STN		V-STN		The Proposed Structure, L-STN	
	Parameters	GFLOPS	Parameters	GFLOPS	Parameters	GFLOPS
EfficientNet-b0	868,704	0.0793	868,542	0.0863	869,200	0.0257
EfficientNet-b1	1,006,944	0.0915	1,006,782	0.0995	1,007,440	0.0297
EfficientNet-b2	1,194,144	0.1080	1,193,982	0.1172	1,194,640	0.0350
EfficientNet-b3	1,616,544	0.1449	1,616,382	0.1570	1,617,040	0.0469
EfficientNet-b4	2,653,344	0.2356	2,653,182	0.2544	2,653,840	0.0760
EfficientNet-b5	3,875,424	0.3418	3,875,262	0.3683	3,875,920	0.1101
ResNet-18	868,704	0.0793	868,542	0.0863	869,200	0.0257
ResNet-50	868,704	0.0793	868,542	0.0863	869,200	0.0257
ResNet-152	868,704	0.0793	868,542	0.0863	869,200	0.0257
DenseNet-121	868,704	0.0793	868,542	0.0863	869,200	0.0257

5. Conclusions

This work has proposed an improved spatial transformer network (L-STN) based on lightweight localization net. Simulation experiments with different spatial transformer network architecture on separate models have been made in this paper to prove the effectiveness of the proposed architecture. Throughout our experiment, L-STN leads in both validation and test accuracy, while manage to saving two thirds of the FLOPS with the cost of slightly more parameters. The proposed lightweight localization net design could seamlessly replace the regular structure with instant performance boost. Convolutional neural networks are being rapidly developed by researchers all around the world, any breakthrough architectural design would push the limit one step ahead. We plan to keep investigating the fittest localization architecture design to achieve the goal: Bring neural network the ability to be spatially invariant to input data with a focus to reach better performance.

References

[1] Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks.” Communications of the ACM 60 (2012): 84 - 90.

[2] Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg and Li Fei-Fei. “ImageNet Large Scale Visual Recognition Challenge.” International Journal of Computer Vision 115 (2015): 211-252.

[3] Girshick, Ross B., Jeff Donahue, Trevor Darrell and Jitendra Malik. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.” 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014): 580-587.

[4] Wang, Naiyan and Dit-Yan Yeung. “Learning a Deep Compact Image Representation for Visual Tracking.” NIPS (2013).

[5] Huang, Huaibo, Ran He, Zhenan Sun and Tieniu Tan. “Wavelet-SRNet: A Wavelet-Based CNN for Multi-scale Face Super Resolution.” 2017 IEEE International Conference on Computer Vision (ICCV) (2017): 1698-1706.

[6] Tai, Ying, Jian Yang and Xiaoming Liu. “Image Super-Resolution via Deep Recursive Residual Network.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): 2790-2798.

[7] Kang, Le, J. Kumar, Peng Ye, Yi Li and David S. Doermann. “Convolutional Neural Networks for Document Image Classification.” 2014 22nd International Conference on Pattern Recognition (2014): 3168-3172.

[8] LeCun, Yann, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard and Lawrence D. Jackel. “Backpropagation Applied to Handwritten Zip Code Recognition.” Neural Computation 1 (1989): 541-551.

[9] LeCun, Yann, Léon Bottou, Yoshua Bengio and Patrick Haffner. “Gradient-based learning applied to document recognition.” (1998).

[10] He, Kaiming, X. Zhang, Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

(2016): 770-778.

[11] Srivastava, Rupesh Kumar, Klaus Greff and Jürgen Schmidhuber. “Training Very Deep Networks.” NIPS (2015).

[12] Huang, Gao, Zhuang Liu and Kilian Q. Weinberger. “Densely Connected Convolutional Networks.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017): 2261-2269.

[13] Zhang, Xiangyu, Xinyu Zhou, Mengxiao Lin and Jian Sun. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 6848-6856.

[14] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam. “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.” ArXiv abs/1704.04861 (2017): n. pag.

[15] Jaderberg, Max, Karen Simonyan, Andrew Zisserman and Koray Kavukcuoglu. “Spatial Transformer Networks.” NIPS (2015).

[16] Simonyan, Karen and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” CoRR abs/1409.1556 (2015): n. pag.

[17] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, D. Erhan, Vincent Vanhoucke and Andrew Rabinovich. “Going deeper with convolutions.” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015): 1-9.

[18] Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke and Alexander Amir Alemi. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.” AAAI (2017).

[19] Sandler, Mark, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov and Liang-Chieh Chen. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 4510-4520.

[20] Zhang, Xiangyu, Xinyu Zhou, Mengxiao Lin and Jian Sun. “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 6848-6856.

[21] Iandola, Forrest N., Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally and Kurt Keutzer. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size.” ArXiv abs/1602.07360 (2016): n. pag.

[22] Tan, Mingxing and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.” ArXiv abs/1905.11946 (2019): n. pag.

[23] Zoph, Barret and Quoc V. Le. “Neural Architecture Search with Reinforcement Learning.” ArXiv abs/1611.01578 (2017): n. pag.

[24] Hu, Jie, Li Shen, Samuel Albanie, Gang Sun and Enhua Wu. “Squeeze-and-Excitation Networks.” IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (2020): 2011-2023.

[25] Ramachandran, Prajit, Barret Zoph and Quoc V. Le. “Searching for Activation Functions.” ArXiv abs/1710.05941 (2018): n. pag.

[26] Ioffe, Sergey and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” ArXiv abs/1502.03167 (2015): n. pag.

[27] Nair, Vinod and Geoffrey E. Hinton. “Rectified Linear Units Improve Restricted Boltzmann Machines.” ICML (2010).

[28] Fei-Fei, Li, Rob Fergus and Pietro Perona. “One-shot learning of object categories.” IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006): 594-611.

[29] Krizhevsky, Alex. “Learning Multiple Layers of Features from Tiny Images.” (2009).

[30] Kingma, Diederik P. and Jimmy Ba. “Adam: A Method for Stochastic Optimization.” CoRR abs/1412.6980 (2015): n. pag.