An Improved Spatial Transformer Network based on Lightweight Localization Net (L-STN)

Tzu-Han Hsu[1][0000−0002−2612−5499], Shin-Jye Lee[2][0000−0003−4265−5016], and Hong-Yu Hsu[3][0009−0009−9858−2566]

[1] Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
hankshyu@gmail.com
https://github.com/hankshyu
[2] Institute of Management of Technology, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
camhero@gmail.com
https://en.mot.nycu.edu.tw
[3] Department of Computer Science, National Chengchi University, Taipei, Taiwan
110703056@nccu.edu.tw
https://github.com/KaidenHsu

Abstract. The technology of computer vision based on deep learning has been prevalently applied to a variety of fields in the industry and the real world. At present, developing an either high performance or lightweight structure network for computer vision is a primary trend for this research work. Meanwhile, Spatial transformer network (STN) is a powerful module that improves the spatial invariance of convolutional neural networks, and which can effectively decrease the computational cost. Due to carry out the above purpose, this work aims to develop a modern localization network architecture that boosts the performance of STN transformation with less parameters and computational overhead. Simulation are conducted through different network architecture designs and examines them with several prominent models on various datasets. As for the practical evidence, the experimental results validate the proposed network structure (L-STN) excels in accuracy while takes only one third FLOPS compared to previous works.

Keywords: Computer Vision · Deep Learning · Lightweight Network.

1 Introduction

Convolutional layers are feature extractors that scan through each pixel with dot product operations, which 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. However, Reproducing training images leads to potential hazards like inflating the dataset causing heavy computation overhead.

Specialized layers are designed to transform the input image before the image was sent into the main convolutional layers. Network structures, like [1], offers neural network the ability to learn reshaping and rotation through the learning process. 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. However, most of the components in such structure are out of date, thereby could not reflect the advancement in convolutional architectural by modern research works. Thus, this work aims to design a renewed structure of such layer to improve its capabilities. We will then then evaluates such re-designed structure on various models to validate the improvements.

2 Related Work

A series of high-performance convolutional neural networks have been recognized as the state-of-the-art machine learning approaches since AlexNet [2] won the ImageNet Challenge: ILSVRC 2012 [3]. This technology has been successfully applied to address a broad range of machine learning tasks, such as object detection, tracking [4], super-resolution [5] and document analysis [6]. Recent works include combining machine learning with swarm intelligence approaches to obtain outstanding results [7, 8]. While network structures [9] have reached hundreds of layers to boost performance, many advanced works focus on finer interconnection methods. [10] renewed convolution procedures to reduce the computational overhead, or develop specialized layers targeting a subset of training problems [11]. 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.

2.1 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 [1] 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.

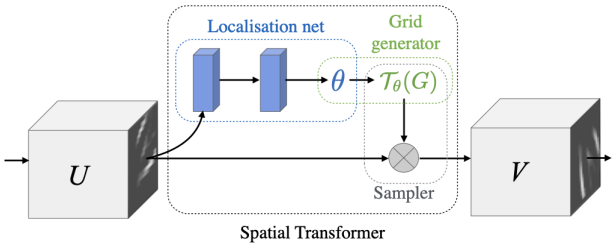

Fig. 1. The Architecture of a Spatial Transformer Module [1]

2.2 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. VGG [12] put forward the concept of unified sized convolutional layers, pooling layers and strides, and proved Local Response Normalization useless. Inception Net [13] 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 1×1 kernel are added to reduce the number of channels.

2.3 Skip Connections

Figure 2 describes a residual learning block Network. Residual neural network (ResNet) [9] 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. The design solved the saturation issue of deep models, so models like DenseNet [10] or [14,15] could build their structures with denser convolution blocks targeting better performance.

2.4 Lightweight Models and EfficientNet

While some researches like [11,16,17] endeavor to make concession between performance and computational overhead. EfficientNet [18] 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. [18] Suggests a simple but effective compound scaling method which uniformly scales three dimensions. Furthermore, they designed a baseline network through neural architecture search [19] then compound scale to an entire family of network called EfficientNet. The

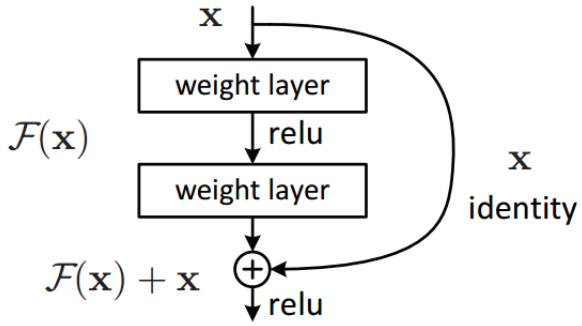

Fig. 2. The Architecture of a Skip Connection [9]

baseline network is based on the inverted bottleneck residual blocks coming from MobileNetV2 [15] in addition to squeeze-and-excitation blocks [20] and Swish activation function [21].

3 Localization Net Architecture

Affine transformation on two-dimensional space requires 6 parameters which are capable of resizing, rotating, displacing and shearing images. According to [1] 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 $H \times W \times C$ in advance and connects with two fully connected layers. One with $H \times W \times C$ as input and 32 as output, another with 32 as input and 6 (for affine transformation) as output. ReLU [22] 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 $(H \times W \times C) \times 32 + 192$ parameters in total.

3.1 Regular STN Structure (STN)

Most commonly used spatial transform network (STN) localization architecture is made out of two wide convolution layers, one 7×7 in the front followed by a 5×5 and with 2×2 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 [23]. Further, Table 1 shows the regular localization architecture.

Table 1. Regular STN Localization Architecture

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

3.2 VGG-like Structure (V-STN)

Breaking down large size convolution filers into stacked small size convolutions are one of the main contributions of VGG [12]. In theory, two 3×3 convolutions would share the same receptive field as one 5×5 convolution as well as three stacked 3×3 convolutions could replace a wide 7×7 convolution layer. Whereas three 3×3 convolutions take 27 parameters but a 7×7 takes 49, over 40% 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 7×7 wide convolution layer into three stacked 3×3 convolution layers and the second 5×5 wide convolution would turn into two 3×3 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	$3 \times 3 \times 3 \times 8$	$222 \times 222 \times 8$	224
Conv	$3 \times 3 \times 8 \times 8$	$220 \times 220 \times 8$	584
Conv	$3 \times 3 \times 8 \times 8$	$218 \times 218 \times 8$	584
Max pool, stride 2	$2 \times 2 \times 8 \times 8$	$109 \times 109 \times 8$	0
Conv	$3 \times 3 \times 8 \times 10$	$107 \times 107 \times 10$	730
Conv	$3 \times 3 \times 10 \times 10$	$105 \times 105 \times 10$	910
Max pool, stride 2	$2 \times 2 \times 10 \times 10$	$52 \times 52 \times 10$	0
Total			3032

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

Convolution layers are three dimensional ($width \times length \times height$), 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 [13] popularizes the use of 1×1 kernels between channels to increase or decrease the feature map count, concentrating the useful ones and discard the wastes. There are two types of 1×1 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 [24] 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 1×1 kernels to concentrate channels and to reduce parameters. Batch normalization were added after the convolution and after the activation function. 3×3 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.

4 Simulation Experiments

4.1 Datasets Descriptions

Caltech 101 The dataset [25] 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,

Table 3. The Proposed Architecture (L-STN)

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

most category has about 50 images with resolution of about 300×200. 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).

Cifar-10 The dataset [26] 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 (32×32) 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:

Preprocessing The datasets are first scaled into the input size of the model using bilinear interpolation, then normalized. Other data preprocessing methods like data augmentation are not performed. Since the work's main goal is to experiment with different model architectures, it's fair since the same dataset is used to train and test our models, so techniques alike are omitted.

Batch Size The batch size is set around 10 to 20. Since this work adopt stochastic gradient descent, batch size does affect the model training process but not much to the final result. In this work, larger training epochs are chosen for models to converge in order to minimize the effect of different batch sizes.

Loss Function 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 [27] 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.

Early Stopping Early stopping is adopted to reduce training resources and avoid overfitting. Stopping conditions are carefully set to avoid prematurely terminating the training process. 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, the training procedure is then terminated.

Hardware Environment 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

Caltech 101 From Table 4, 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 456×456, 4 times the size of the ordinary 224×224 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 4, 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.

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.

Owing the fact that L-STN is imitating an entire CNN architecture at the localization architecture. It 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.

Cifar-10 From Table 5 could illustrate that all of the models have higher accuracy when spatial transform layers were added even on larger dataset like Cifar-10. 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.

Table 5 strongly suggests that the model's accuracy before and after adding the spatial transformer layer are strongly related. 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.

4.4 Computational Cost of STN Architectures

Computational overhead is also taken into consideration. The number of parameters and the FLOPS take to compute them are in Table 6. The size of the layer

Table 4. Parameters, GFLOPS, Validation and Test Accuracy of Different models on Caltech 101

Model	Original			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	14,163,937	1.730	73.33	72.33
EfficientNet-b1	27,357,540	3.558	72.83	71.58
EfficientNet-b2	29,615,577	4.837	66.83	66.83
EfficientNet-b3	40,789,556	8.260	63.33	60.83
EfficientNet-b4	56,613,262	18.432	68.67	69.25
EfficientNet-b5	95,571,391	41.653	70.00	68.75
ResNet-50	23,714,981	4.110	75.17	73.58
Model	STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	15,032,641	1.809	75.67	73.92
EfficientNet-b1	28,364,484	3.650	73.33	71.92
EfficientNet-b2	30,809,721	4.945	68.50	68.42
EfficientNet-b3	42,406,100	8.405	69.83	69.83
EfficientNet-b4	59,266,606	18.668	69.42	71.83
EfficientNet-b5	99,446,815	41.994	72.75	71.83
ResNet-50	24,583,685	4.189	76.42	76.92
Model	V-STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	15,032,479	1.816	71.83	74.25
EfficientNet-b1	28,364,322	3.658	71.42	72.25
EfficientNet-b2	30,809,559	4.954	69.58	68.50
EfficientNet-b3	42,405,938	8.417	74.00	73.67
EfficientNet-b4	59,266,444	18.687	73.25	72.08
EfficientNet-b5	99,446,653	42.021	73.25	72.08
ResNet-50	24,583,523	4.196	78.75	77.00
Model	The proposed structure, L-STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	15,033,137	1.756	74.00	74.83
EfficientNet-b1	28,364,980	3.588	72.00	73.00
EfficientNet-b2	30,810,217	4.872	73.17	75.00
EfficientNet-b3	42,406,596	8.307	75.58	75.42
EfficientNet-b4	59,267,102	18.508	73.92	74.83
EfficientNet-b5	99,447,311	41.763	79.83	78.33
ResNet-50	24,584,181	4.135	77.42	77.33

Table 5. Parameters, GFLOPS, Validation and Test Accuracy of Different models on Cifar-10

Model	Original			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	14,047,366	1.7299	86.83	86.05
EfficientNet-b1	27,235,236	3.5579	86.90	86.89
EfficientNet-b2	29,487,358	4.8370	86.65	86.38
EfficientNet-b3	40,648,506	8.2597	87.11	86.79
ResNet-18	11,181,642	1.8185	85.60	84.96
ResNet-50	23,528,522	4.1094	85.07	84.84
ResNet-152	58,164,298	11.5567	85.15	84.65
DenseNet-121	6,964,106	2.8647	88.91	88.61
Model	STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	14,916,070	1.8092	87.48	86.40
EfficientNet-b1	28,242,180	3.6494	87.01	86.95
EfficientNet-b2	30,681,502	4.9450	87.81	87.30
EfficientNet-b3	42,265,050	8.4046	87.12	86.96
ResNet-18	12,050,346	1.8979	85.44	85.53
ResNet-50	24,397,226	4.1888	85.43	85.01
ResNet-152	59,033,002	11.6361	85.09	84.81
DenseNet-121	7,832,810	2.9440	89.42	89.29
Model	V-STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	14,915,908	1.8162	87.88	86.81
EfficientNet-b1	28,242,018	3.6574	87.62	86.98
EfficientNet-b2	30,681,340	4.9542	87.57	87.36
EfficientNet-b3	42,264,888	8.4167	87.19	87.02
ResNet-18	12,050,184	1.9049	85.79	85.83
ResNet-50	24,397,064	4.1958	85.63	85.02
ResNet-152	59,032,840	11.6431	85.40	85.16
DenseNet-121	7,832,648	2.9510	89.88	89.51
Model	The Proposed Structure, L-STN			
	Parameters	GFLOPS	Top-5 val.	Top-5 test
EfficientNet-b0	14,916,566	1.7556	87.71	87.45
EfficientNet-b1	28,242,676	3.5876	87.71	87.03
EfficientNet-b2	30,681,998	4.8720	87.77	87.52
EfficientNet-b3	42,265,546	8.3064	87.64	87.22
ResNet-18	12,050,842	1.8443	86.37	86.41
ResNet-50	24,397,722	4.1352	86.69	86.19
ResNet-152	59,033,498	11.5825	86.86	85.17
DenseNet-121	7,833,306	2.8904	86.17	90.28

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 6. Parameters & GFLOPS of Different Spatial Transformer Architecture

Model	Original		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.

6 Acknowledgments

This research is partially supported by the Ministry of Science and Technology research grant in Taiwan (NSTC 112-2410-H-A49 -033 -).

References

1. Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in neural information processing systems 28 (2015).
2. Krizhevsky, Alex, Ilya Sutskever and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks.” Communications of the ACM 60 (2012): 84 - 90.
3. 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.
4. 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.
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. 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.
7. Bacanin, Nebojsa, et al. "Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application." Scientific Reports 12.1 (2022): 6302.
8. Bacanin, Nebojsa, et al. "Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization." Mathematics 9.21 (2021): 2705.
9. 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.
10. 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.
11. Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).
12. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).
13. 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.
14. Szegedy, Christian, Sergey Ioffe, Vincent Vanhoucke and Alexander Amir Alemi. “Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.” AAAI (2017).
15. 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.
16. 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.

17. Iandola, Forrest N., et al. "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size." arXiv preprint arXiv:1602.07360 (2016).
18. Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." International conference on machine learning. PMLR, 2019.
19. Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).
20. 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.
21. Ramachandran, Prajit, Barret Zoph, and Quoc V. Le. "Searching for activation functions." arXiv preprint arXiv:1710.05941 (2017).
22. Nair, Vinod, and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." Proceedings of the 27th international conference on machine learning (ICML-10). 2010.
23. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324
24. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. pmlr, 2015.
25. 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.
26. Krizhevsky, Alex, and Geoffrey Hinton. "Learning multiple layers of features from tiny images." (2009): 7.
27. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).