S3Pool: Pooling with Stochastic Spatial Sampling

Shuangfei Zhai[†] Hui Wu[‡] Abhishek Kumar[‡] Yu Cheng[‡] Yongxi Lu[§] Zhongfei Zhang[†] Rogerio Feris[‡]
Binghamton Univeristy[†] IBM Research[‡] UC San Diego[§]
{szhai2, zhongfei}@binghamton.edu {wuhu, abhishk, chengyu, rsferis}@us.ibm.com yol070@ucsd.edu

Abstract

Feature pooling layers (e.g., max pooling) in convolutional neural networks (CNNs) serve the dual purpose of providing increasingly abstract representations as well as yielding computational savings in subsequent convolutional layers. We view the pooling operation in CNNs as a two-step procedure: first, a pooling window (e.g., 2×2) slides over the feature map with stride one which leaves the spatial resolution intact, and second, downsampling is performed by selecting one pixel from each non-overlapping pooling window in an often uniform and deterministic (e.g., top-left) manner. Our starting point in this work is the observation that this regularly spaced downsampling arising from non-overlapping windows, although intuitive from a signal processing perspective (which has the goal of signal reconstruction), is not necessarily optimal for learning *(where the goal is to generalize). We study this aspect and propose a novel pooling strategy with stochastic spatial sampling (S3Pool), where the regular downsampling is replaced by a more general stochastic version. We observe that this general stochasticity acts as a strong regularizer, and can also be seen as doing implicit data augmentation by introducing distortions in the feature maps. We further introduce a mechanism to control the amount of distortion to suit different datasets and architectures. To demonstrate the effectiveness of the proposed approach, we perform extensive experiments on several popular image classification benchmarks, observing excellent improvements over baseline models.*

1. Introduction

The use of pooling layers (max pooling, in particular) in deep convolutional neural networks (CNNs) is critical for their success in modern object recognition systems. In most of the common implementations, each pooling layer downsamples the spatial dimensions of feature maps by a factor of s (e.g., 2). This not only reduces the amount of computation required by the time consuming convolution operation in subsequent layers of the network, it also facilitates the higher layers to learn more abstract representations by looking at larger receptive fields.

In this paper, we provide new insights into the design of the pooling operation by viewing it as a two-step procedure. In the first step, a pooling window slides over the feature map with stride size 1 producing the pooled output; in the second step, spatial downsampling is performed by extracting the top-left corner element of each disjoint $s \times s$ window, resulting in a feature map with s times smaller spatial dimensions. Our starting point in this work is the observation that although this uniformly spaced spatial downsampling is reasonable from a signal processing perspective which aims for signal reconstruction [19] and is also computationally friendly, it is not necessarily the optimal design for the purpose of *learning* which aims for generalization to unseen examples[1].

Motivated by this observation, we introduce and study a novel pooling scheme, named *S3Pool*, where the second step (downsampling) is modified to a stochastic version. For a feature map with spatial dimensions $h \times w$, S3Pool begins with partitioning it into p vertical and q horizontal strips, with $p = \frac{h}{g}$, $q = \frac{w}{g}$ and g being a hyperparameter named grid size. It then randomly selects $\frac{g}{s}$ rows and $\frac{g}{s}$ columns for each horizontal and vertical strip, respectively, with s being the stride, to obtain the final downsampled feature map of size $\frac{h}{s} \times \frac{w}{s}$. Compared to the downsampling used in standard pooling layers, S3Pool performs a spatial downsampling that is *stochastic* and hence is highly likely to be *non-uniform*. The stochastic nature of S3Pool enables it to produce different feature maps at each pass for the same training examples, which amounts to implicitly performing a sort of data augmentation [20], but at intermediate layers. Moreover, the non-uniform characteristics of S3Pool further extends the space of possible downsampled feature maps, which produces spatially distorted downsampled versions at each pass. The grid size g provides a handle for controlling the amount of distortion that S3Pool introduces, which can be used to adapt to CNNs with different designs,

[1]Uniform sampling has also been examined in the Signal Processing literature, e.g., J. R. Higgins writes [10]: *"What is special about equidistantly spaced sample points?"; and then finding that the answer is "Within certain limitations, nothing at all."*

Figure 1: Illustration of the effect of different downsampling strategies. Left panel: the image before downsampling. Right panel from top left to bottom right: uniform downsampling, stochastic spatial downsampling with the grid size equivalent to a quarter of the image width/height, half of the image width/height, and the image width/height, respectively.

and different datasets. Overall, S3Pool acts as a strong regularizer by performing 'virtual' data augmentation at each pooling layer, and greatly enhances a model's generalization ability as observed in our empirical study.

Practically, S3Pool does not introduce any additional parameters, and can be plugged in place of any existing pooling layers. We have also empirically verified that S3Pool only introduces marginal computational overheads during training time (evaluated by time per epoch). During test time, S3Pool can either be reduced to standard max pooling, or be combined with an additional average pooling layer for a slightly better approximation of the stochastic downsampling step. In our experiments, we show that S3Pool yields excellent results on three standard image classification benchmarks, with two state-of-the-art architectures, namely network in network [17], and residual networks [9]. We also extensively experiment with different data augmentation strategies, and show that under each setting, S3Pool is able to outperform other counterparts such as dropout [22] and stochastic pooling [26].

2. Related Work

The idea of spatial feature pooling dates back to the seminal work by Hubel and Wiesel [11] about complex cells in the mammalian visual cortex and the early CNN architectures developed by Yann Lecun *et al.* [15]. Prior to the re-emergence of deep neural networks in computer vision, different approaches based on bag-of-words and fisher vector coding also had spatial pooling as an essential component of the visual recognition pipeline, e.g., through orderless bag-of-features [2, 6], spatial pyramid aggregation [14], or task-driven feature pooling [23].

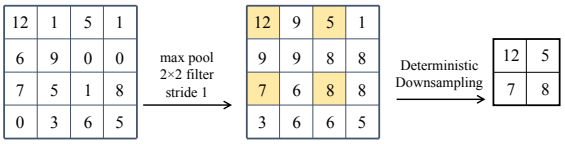
In modern CNN architectures, spatial pooling plays a fundamental role in achieving invariance (to some extent) to image transformations, and produces more compact representations for efficient processing in subsequent layers. Most existing methods rely on *max* or *average* pooling layers. Hybrid pooling [16, 18] combines different types of pooling into the same network architecture. Stochastic pooling [26] randomly picks the activation within each pooling region according to a multinomial distribution. Max-out networks [4, 21] perform pooling across different feature maps. Spatial pyramid pooling [8] aggregates features at multiple scales, and is usually applied to extract fixed-length feature vectors from region proposals for object detection. Fractional pooling [5] proposes to use pooling strides of less than 2 by applying mixed pooling strides of 1 and 2 at different locations. Learning-based methods for spatial feature pooling have also been proposed [7, 1].

As discussed previously, we view pooling as two distinct steps and propose stochastic spatial sampling as a novel solution that has not been investigated in previous work, to the best of our knowledge. Our approach is simple to implement, very efficient, and complementary to most of the techniques discussed above.

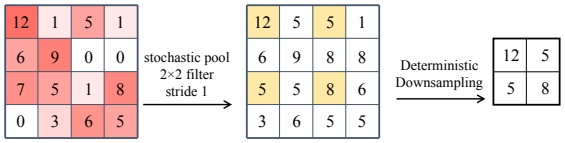
3. Model Description

3.1. A Two-Step View of Max Pooling

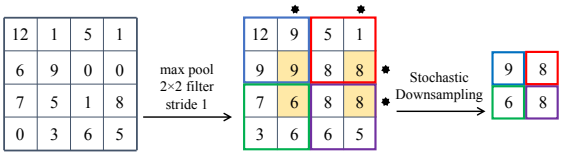
Max pooling is perhaps the most widely adopted pooling option in deep CNNs, which usually follows one or several convolutional layers to reduce the spatial dimensions of the feature maps. Let $\mathbf{x} \in R^{c \times h \times w}$ be the input feature map before a pooling layer, where c is the number of channels and h and w are the height and width, respectively. A max



(a) Max pooling, pooling window $k = 2$, stride $s = 2$



(b) Stochastic pooling [26], pooling window $k = 2$, stride $s = 2$



(c) S3Pool, pooling window $k = 2$, stride $s = 2$, grid size $g = 2$

Figure 2: Comparison of different pooling methods (best seen in color). Max pooling (a) consists of two steps, selecting the activation inside each pooling region and spatial downsampling, where both steps are deterministic. Stochastic pooling [26] adapts the first step by choosing the activation with a stochastic procedure (b). While our method modifies the second step by randomly selecting rows and columns from each spatial grid (c).

pooling layer with pooling window of size $k \times k$ and stride $s \times s$ is defined by the function $\mathbf{z} = \mathcal{P}_k^s(\mathbf{x})$, where $\mathbf{z} \in R^{c \times \frac{h}{s} \times \frac{w}{s}}$, and

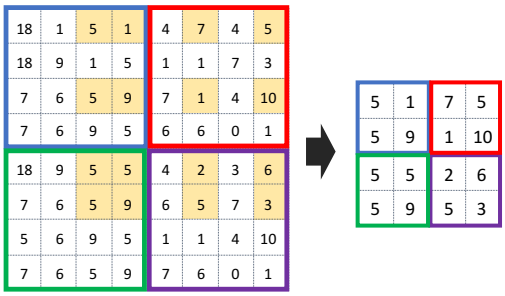
$$\mathbf{z}_{n,i,j} = \max_{\substack{i' \in [(i-1)s+1,(i-1)s+k], i' \leqslant h \\ j' \in [(j-1)s+1,(j-1)s+k], j' \leqslant w}} \mathbf{x}_{n,i',j'}, \quad n \in [1, c], i \in [1, \frac{h}{s}], j \in [1, \frac{w}{s}]. \tag{1}$$

Specifically, to obtain the value at each spatial location of the output feature map $\mathbf{z}$, $\mathcal{P}_k^s(\cdot)$ selects the maximum activation within the corresponding local region of size $k \times k$ in $\mathbf{x}$. While performed in a single step, conceptually, max pooling can be considered as two consecutive processes:

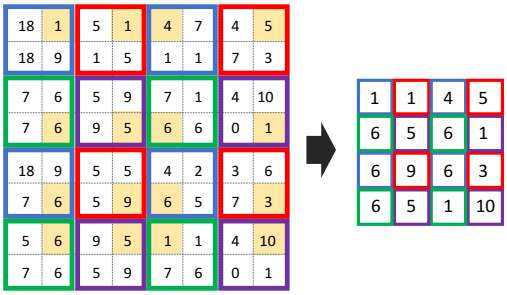
$$\mathbf{o} = \mathcal{P}_k^1(\mathbf{x}), \quad \mathbf{z} = \mathcal{D}^s(\mathbf{o}), \tag{2}$$

where $\mathbf{z}_{n,i,j} = \mathbf{o}_{n,(i-1)s+1,(j-1)s+1}$.

In the first step, max pooling with window size $k \times k$ and stride 1×1 is performed, producing an intermediate output $\mathbf{o}$, which has the same dimension as $\mathbf{x}$. In the second step, a



(a) stride $s = 2$, grid size $g = 4$



(b) stride $s = 2$, grid size $g = 2$

Figure 3: Controlling the amount of distortion/stochasticity by changing the grid size g in the stochastic downsampling step (best seen in color).

spatial downsampling step is performed, where the value at the top left corner of each disjoint $s \times s$ window is selected to produce the output feature map with the spatial dimension reduced by s times. The two-step view of max pooling allows us to investigate the differences of the effects of each step on learning. The first step $\mathcal{P}_k^1(\cdot)$ provides an additional level of nonlinearity to the CNN, as well as a certain degree of local (up to the scale of $k \times k$) distortion invariance. The second step $\mathcal{D}^s(\cdot)$, on the other hand, serves the purpose of reducing the amount of computation and weight parameters (given a fixed receptive field size) needed at upper layers of a deep CNN, as well as facilitating the model to learn more abstract representations by providing a more compact view of the input. We exploit this two-step view of the classical max pooling procedure and introduce a pooling algorithm which explicitly improves the downsampling step in order to learn models with better generalization ability.

3.2. Pooling with Stochastic Spatial Sampling

While the typical downsampling step of a max pooling layer intuitively reduces the spatial dimension of a feature map by always selecting the activations at fixed locations, this design choice is somewhat arbitrary and potentially suboptimal. For example, as specified in Equation 2, the downsampling function $\mathcal{D}^s(\cdot)$ selects only the activation at

the top left corner of each $s \times s$ disjoint window and discards the rest $s^2 - 1$ activations, which are equally informative for learning. Considering the total number of pooling layers present in a CNN, denoted by L, this deterministic downsampling approach discards $s^{2L} - 1$ possible sampling choices. Therefore, although a natural design choice, deterministic uniform spatial sampling may not be optimal for the purpose of learning where the goal is to generalize. On the other hand, if we allow the downsampling step to be performed in a non-uniform and non-deterministic way, where the sampled indices are not restricted to be at evenly distributed locations, we are able to produce many variations of downsampled feature maps. Motivated by this observation, we propose S3Pool, a variant of max pooling with a stochastic spatial downsampling procedure [2]. S3Pool, denoted by $\tilde{\mathcal{P}}^s_{k,g}(\cdot)$, works in a two-step fashion: the first step, $\mathcal{P}^1_k(\cdot)$, is identical to max pooling, however, the second step, $\mathcal{D}^s(\cdot)$, is replaced by a stochastic version $\tilde{\mathcal{D}}^s_g(\cdot)$.

Prior to the downsampling step of S3Pool, the feature map is divided into $\frac{h}{g}$ vertical and $\frac{w}{g}$ horizontal disjoint grids, indexed by $p \in [1, \frac{h}{g}]$ and $q \in [1, \frac{w}{g}]$, respectively, with g being the grid size. Within each vertical/horizontal grid, $\frac{g}{s}$ rows/columns are randomly chosen:

$$\mathbf{r}^p = \mathrm{C}^{\frac{g}{s}}_{[(p-1)g+1,pg]},\ \mathbf{c}^q = \mathrm{C}^{\frac{g}{s}}_{[(q-1)g+1,qg]}, \tag{3}$$

where $\mathrm{C}^m_{[a,b]}$ denotes a multinomial sampling function, which samples m sorted integers randomly from the interval $[a, b]$ without replacement. The indices drawn from each vertical/horizontal grid are then concatenated, producing a set of rows, $\mathbf{r} = [\mathbf{r}^1, \mathbf{r}^2, \cdots, \mathbf{r}^{\frac{h}{g}}]$ and a set of columns, $\mathbf{c} = [\mathbf{c}^1, \mathbf{c}^2, \cdots, \mathbf{c}^{\frac{w}{g}}]$, which leaves us the downsampled feature map being: $\mathbf{z} = \tilde{\mathcal{D}}^s_g(\mathbf{o})$, where $\mathbf{z}_{n,i,j} = \mathbf{o}_{n,\mathbf{r}_i,\mathbf{c}_j}$. To summarize, given the grid size g, the stride s and the pooling window size k, S3Pool is defined as:

$$\mathbf{z} = \tilde{\mathcal{D}}^s_g(\mathcal{P}^1_k(\mathbf{x})) \tag{4}$$

The grid size, g, is a hyperparameter of S3Pool which controls the level of stochasticity introduced. Figure 3 illustrates the effect of changing the grid size for the stochastic spatial downsampling $\mathcal{D}^2_g(\cdot)$. Larger grid sizes correspond to less uniformly sampled rows and columns. In the extreme case, where the grid size equals to the image size, S3Pool selects $\frac{h}{s}$ rows and $\frac{w}{s}$ columns from the entire input feature map in a purely random fashion, which yields the maximum amount of randomness in sampling.

The behavior of $\tilde{\mathcal{D}}^s_g(\cdot)$ is intuitively visualized using an image as input (Figure 1), which is downsampled by applying uniform sampling, $\mathcal{D}^2(\cdot)$, and stochastic downsampling with different grid sizes, $\tilde{\mathcal{D}}^2_{\frac{w}{4}}(\cdot)$, $\tilde{\mathcal{D}}^2_{\frac{w}{2}}(\cdot)$, $\tilde{\mathcal{D}}^2_{w}(\cdot)$. It can be seen that all the stochastic spatial sampling variants produce images that are recognizable to human eyes, with certain degrees of distortion, even in the extreme case where the grid size equals to the image size. The benefit of S3Pool is thus obvious in that, each draw from the pooling step will produce different yet plausible downsampled feature maps, which is equivalent to performing data augmentation [20] at the pooling layer level. However, compared with traditional data augmentation, such as image cropping [13], the distortion introduced by S3Pool is more aggressive. As a matter of fact, cropping (which corresponds to horizontal and vertical translation) can be considered as a special case of S3Pool in the input layer, with $s = 1$ and $g = w$, with the additional constraint that the sampled rows and columns are spatially contingent.

To further illustrate the idea of S3Pool and its difference from the standard max pooling, and another non-deterministic variant of max pooling [26], we demonstrate the different pooling processes in Figure 2 using a toy feature map of size $1 \times 4 \times 4$. From the two-step view of max pooling, stochastic pooling [26] modifies the first step: instead of outputing a deterministic maximum in each pooling window of $k \times k$, it randomly draws a response according to the magnitude of the activation; the second downsampling step, however, remains the same as in max pooling. Different from stochastic pooling [26] and deterministic max pooling, S3Pool offers the flexibility to control the amount of distortion introduced in each sampling step by varying the grid size g in each layer. This is useful especially for building deep CNNs with multiple pooling layers, which makes it possible to control the trade-off between the regularization strength and the converging speed.

In terms of implementation concerns, S3Pool does not introduce any additional parameters. It is easy to implement, and fast to compute during training time (in our experiments, we show that S3Pool introduces very little computational overhead compared to max pooling).

Inference Stage. During testing time, a straightforward but inefficient approach is to take the average classification outputs from many instances of CNN with S3Pool, which can otherwise act as a finite sample estimate of the expectation of S3Pool downsampling. A more efficient approach is to use the expectation of the downsampling procedure during testing. The expected value at a location (i, j) in the feature map (with $\widetilde{si} := ((i-1) \mod g/s) + 1, \breve{si} := \lfloor s(i-1)/g \rfloor$, similarly for sj, and $i \in [h/s]$, $j \in [w/s]$) is given as

$$E[\mathbf{z}_{n,i,j}] = \sum_{a=\breve{si}}^{g-g/s+\breve{si}} \sum_{b=\breve{si}}^{g-g/s+\breve{si}} w_{a,b} \mathbf{o}_{n,g\breve{si}+a,g\breve{sj}+b},$$

where $w_{ab} = h_a h_b$ with $h_a = \binom{a-1}{\breve{si}-1}\binom{g-a}{g/s-\breve{si}} / \binom{g}{g/s}$ with

[2] Although we work with max pooling as the underlying pooling mechanism since it is widely used, the proposed S3Pool is oblivious to the nature of the first stage pooling and is applicable just as well to other types of pooling schemes, e.g., average pooling, stochastic pooling [26], as well as strided convolution.

the convention $\binom{0}{0} = 1$ (similar for h_b with $\widetilde{si}$ replaced with $\widetilde{sj}$). For $g = s$, this expectation reduces to average pooling over the $s \times s$ windows in the second downsampling step. For $g > s$, computing this expectation is expensive and cannot be easily parallelized in a GPU implementation, we thus still use average pooling with window and stride s in our experiments during testing as an approximation of this expectation. We also experimented with standard uniformly spaced downsampling at testing time (i.e., picking the top-left corner pixel), however this was consistently outperformed by average pooling, with negligible computational overhead. Hence, all the testing results of S3Pool in this paper are computed with average pooling over $s \times s$ windows.

4. Experiments

We evaluate S3Pool with three popular image classification benchmarks: CIFAR-10, CIFAR-100 and STL-10. Both CIFAR-10 and CIFAR-100 consist of 32×32 color images, each with 50,000 images for training and 10,000 images for testing. STL-10 consists of 96×96 colored images evenly distributed in 10 classes, with 5,000 images for training and 8,000 images for testing. All the three datasets have relatively few examples, which makes proper regularization extremely important. We note that it is not our goal to obtain state-of-the-art results on these datasets, but rather to provide a fair analysis of the effectiveness of S3Pool compared to other pooling and regularization methods.

Table 1: The configurations of NIN and ResNet used on CIFAR-10 and CIFAR-100. Conv-c-d stands for a convolutional layer with c filters of size $d \times d$. Pool-k-s stands for a pooling layer with pooling window $k \times k$ and stride $s \times s$.

NIN	ResNet
Conv-192-5 Conv-160-1 Conv-96-1	Conv-32-3
Pool-2-2	$3 \times$ { Conv-32-3 Conv-32-3 }
Conv-192-5 Conv-192-1 Conv-192-1	Pool-2-2
Pool-2-2	$3 \times$ { Conv-64-3 Conv-64-3 }
Conv-192-3 Conv-192-1 Conv-10-1	Pool-2-2
Global Average Pooling	$3 \times$ { Conv-128-3 Conv-128-3 }
Softmax	Conv-10-1
	Global Average Pooling
	Softmax

4.1. CIFAR-10 and CIFAR-100

For CIFAR-10 and CIFAR-100, we experiment with two state-of-the-art architectures, network in network (NIN) [17] and residual networks (ResNet) [9], both of which are well established architectures, but with different designs. We apply identical architectures on CIFAR-10 and CIFAR-100, except for the top convolultional layer for softmax (10 versus 100). The architectures we use in this paper differ slightly from those in [17, 9], which we summarize in Table 1. Here Conv-c-d denotes a convolutional layer with c filters of size $d \times d$; Pool-k-s denotes a pooling layer implementation with pooling window $k \times k$ and stride $s \times s$ [3]. Batch normalization [12] is applied to each convolutional layer for each of the two models, with ReLU as the nonlinearity.

For each of the two models, we experiment with three variants of the pooling layers:

Standard pooling: for NIN, both of the two Pool-2-2 layers are max pooling with pooling window of size 2×2 and stride 2×2; a dropout layer with rate 0.5 is also inserted after each pooling layer. For ResNet, we follow the original design in [9] by replacing the Pool-2-2 layer with stride 2 convolution, without dropout.

Stochastic pooling: proposed by Zeiler et al. [26] with pooling window of size 2×2 and stride 2×2.

S3Pool: the proposed pooling method with pooling window of size 2×2 and stride 2×2. Grid size g is set as 16 and 8 for the first and second S3Pool layer, respectively (that is, each feature map is divided into 2 vertical and horizontal strips). We denote this implementation of S3Pool as S3Pool-16-8.

In addition to experimenting with different network structures and pooling methods, we also employ different data augmentation strategies: with or without horizontal flipping and without or without cropping [4]. We train all the models with ADADELTA [25] with an initial learning rate of 1 and a batch size of 128. For all the NIN variants, training takes 200 epochs with the learning rate reduced to 0.1 at the 150-th epoch. All the ResNet variants are trained for a total of 120 epochs with the learning rate reduced to 0.1 at the 80-th epoch.

The experimental results are summarized in Table 2 and Table 3 for NIN and Resnet respectively. For each set of the experiments, we show the training and testing error of the final epoch (for S3Pool, an average pooling layer of pooling window and stride 2×2 is added following each S3Pool layer). We also show the average training time of each pooling option when used with different networks, measured by the number of seconds per epoch (that is, the time taken for a full pass of the training data for weight updates, and a full

[3] except for the baseline ResNet, which refers to a simple downsampling of $s \times s$.

[4] 4 pixels are padded at each border of the 32×32 images, and random 32×32 crops are selected at each forward pass.

Table 2: Control experiments with NIN [17] on CIFAR-10 and CIFAR-100 (best seen in color).

Model	**flip**	**crop**	CIFAR-10		CIFAR-100		sec/epoch
			train err	test err	train err	test err	
NIN + dropout	N	N	0.63	10.68	6.15	35.24	131
NIN + dropout	N	Y	1.62	10.11	11.64	34.08	
NIN + dropout	Y	N	1.28	9.75	8.57	33.48	
NIN + dropout	Y	Y	2.67	9.34	14.15	32.36	
Zeiler et al.[26]	N	N	0.01	12.86	0.1	39.64	218
Zeiler et al.[26]	N	Y	0.06	10.97	0.78	35.44	
Zeiler et al.[26]	Y	N	0.02	10.47	0.20	36.82	
Zeiler et al.[26]	Y	Y	0.22	9.14	1.54	33.47	
S3Pool-16-8	N	N	1.85	9.30	9.25	33.85	142
S3Pool-16-8	N	Y	2.86	8.77	11.44	33.24	
S3Pool-16-8	Y	N	3.26	8.04	13.19	31.04	
S3Pool-16-8	Y	Y	4.39	**7.71**	16.66	**30.90**	

Table 3: Control experiments with ResNet [17] on CIFAR-10 and CIFAR-100 (best seen in color).

Model	**flip**	**crop**	CIFAR-10		CIFAR-100		sec/epoch
			train err	test err	train err	test err	
ResNet	N	N	0.00	14.07	0.02	42.32	120
ResNet	N	Y	0.01	9.21	0.06	33.88	
ResNet	Y	N	0.00	11.14	0.02	36.05	
ResNet	Y	Y	0.06	7.72	0.48	30.88	
Zeiler et al.[26]	N	N	0.01	9.94	0.04	34.42	152
Zeiler et al.[26]	N	Y	0.04	8.60	0.27	33.16	
Zeiler et al.[26]	Y	N	0.05	8.06	0.15	31.76	
Zeiler et al.[26]	Y	Y	0.23	8.58	1.24	30.09	
S3Pool-16-8	N	N	0.82	8.86	3.97	32.78	125
S3Pool-16-8	N	Y	1.47	8.48	7.24	32.21	
S3Pool-16-8	Y	N	1.90	7.31	8.28	30.65	
S3Pool-16-8	Y	Y	3.23	**7.09**	12.47	**29.36**	

pass of the testing data).

We observe that for every combination of dataset type, network architecture and data augmentation technique (denoted by rows with the same color in Table 2 and Table 3), S3Pool achieves the lowest testing error, while yielding higher training errors than NIN with dropout, ResNet and their counterparts with stochastic pooling [26]. More remarkably, S3Pool without any data augmentation can outperform other methods with data augmentation in most of cases. In particular, S3Pool without data augmentation is able to outperform the baselines with cropping on all of the four dataset and architecture combinations. On CIFAR-10, S3Pool is even able to outperform image flipping and cropping augmented dropout version of NIN (9.30 versus 9.34). The high performance of S3Pool even without data augmentation is consistent with our understanding of the stochastic spatial sampling step as an implicit data augmentation strategy. Interestingly, while both flipping and cropping are beneficial to S3Pool, flipping seems to produce more performance gain than cropping. This is reasonable since the stochastic downsampling step in S3Pool does not change the horizontal spatial order of sampled columns.

As for the computational cost, S3Pool increases the training time by 8% and 4% on NIN and ResNet, respectively. Stochastic pooling, on the other hand, yields a much higher computational overhead of 66% and 27%, respectively [5]. This demonstrates that S3Pool is indeed a practical as well as effective implementation choice when used in deep CNNs.

Effect of grid size To investigate the effect of the grid size of S3Pool, we take the same ResNet architecture used in Section 4.1, replace the S3Pool-16-8 layers with differ-

[5] All models are implemented with Theano, and ran on a single NVIDIA K40 GPU.

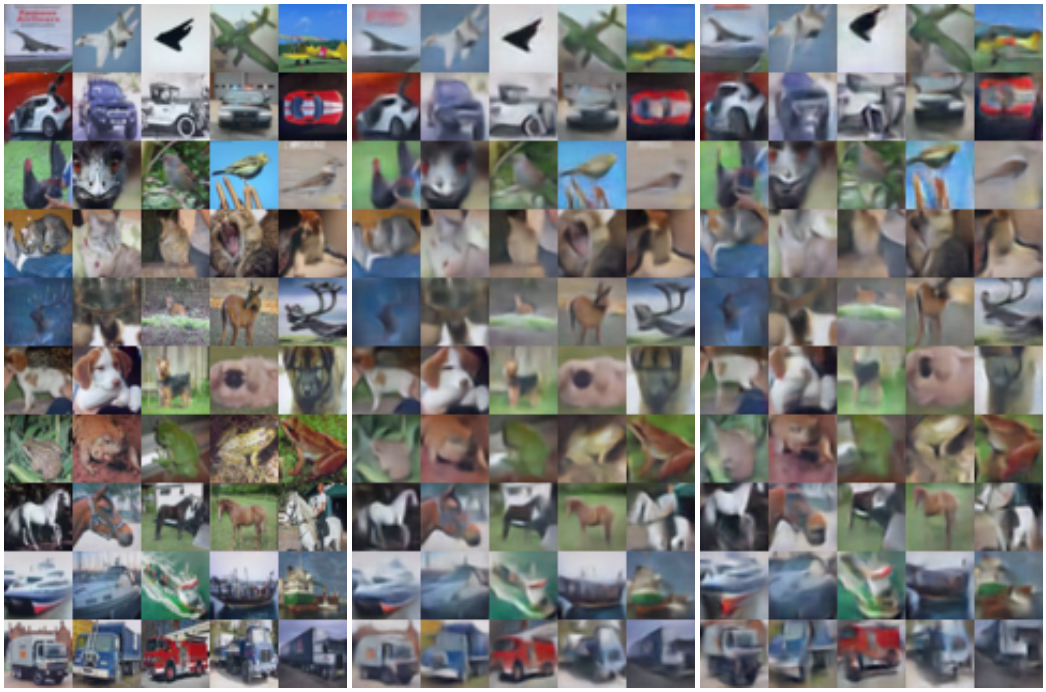


Figure 4: Illustration of the behavior of S3Pool with deconvolutional neural networks on CIFAR-10 (best seen in color). From left to right: 50 images sampled from the test set, reconstructions obtained after the second pooling layer when using deterministic max pooling (center) and S3Pool (right). Note that even after two layers of stochastic spatial sampling, one is able to reconstruct recognizable images with various spatial distortions.

Table 4: Performance of different configurations of S3Pool by varying the grid sizes. All results are obtained with ResNet on CIFAR-10, without any data augmentation.

Configuration	train err	test err
S3Pool-32-16	2.58	9.32
S3Pool-16-8	0.82	**8.86**
S3Pool-8-8	1.29	10.14
S3Pool-8-4	0.92	11.04
S3Pool-4-4	0.72	11.02
S3Pool-2-2	0.26	13.01

ent grid size settings, and report the results on CIFAR-10 in Table 4. We can observe that, in general, increasing the grid size of S3Pool yields larger training errors, as a result of more stochasticity; the testing error on the other hand, first decreases thanks to stronger regularization, then increases when the training error is too high. This observation suggests a trade-off between the optimization feasibility and the generalization ability, which can be adjusted in different applications by setting the grid sizes of each S3Pool layer.

Learning with limited training data We further take the same ResNet architecture, and perform experiments with fewer training examples in CIFAR-10, which is shown in Figure 5. The results indicate that, by varying the number of training examples from as low as 1000 to 10000, S3Pool achieves consistently lower testing errors compared with the baseline ResNet as well as stochastic pooling [26].

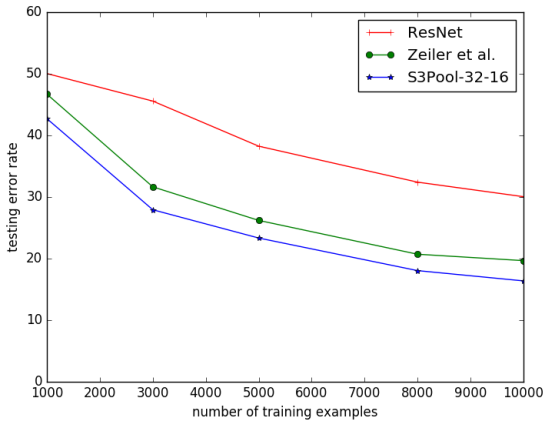


Figure 5: Testing error rate on CIFAR-10 with different training data sizes (best seen in color).

Table 5: Results on STL-10. S3Pool-g_1-g_2-g_3-g_4 denotes the configuration of the grid size at each of the four S3Pool layer.

model	train err	test err	sec/epoch
ResNet	0.00	39.84	30
Zeiler et al. [26]	0.00	25.93	70
S3Pool-96-48-24-12	2.12	**24.06**	35
S3Pool-48-24-12-6	1.04	25.36	
S3Pool-24-12-6-4	0.12	29.21	
S3Pool-12-6-4-4	0.12	30.01	
S3Pool-4-4-4-4	0.06	29.60	
S3Pool-2-2-2-2	0.02	35.14	
Zhao et al. [28]	-	25.47	-
Dosovitskiy et al. [3]	-	27.2	
Yang et al. [24]	-	26.85	

4.2. STL-10

STL-10 has much fewer training examples and larger image sizes compared with CIFAR-10/CIFAR-100. We adopt the 18-layer ResNet based architecture on this dataset, and test different pooling methods by replacing the stride 2 convolutions by stochastic pooling [26] and S3Pool with different grid size settings. We follow similar training protocols as in Section 4.1, except that all the models are trained for 200 epochs with the learning rate decreased by a factor of 10 at the 150-th epoch, with no data augmentation applied.

The results are summarized in Table 5. All variations of S3Pool significantly improve the performance of the baseline ResNet. In particular, S3Pool with the strongest regularization (S3Pool-96-48-24-12) achieves the state-of-the-art testing error on STL-10, outperforming supervised learning [24] as well as semi-supervised learning [28, 3] approaches. In terms of computational cost, S3Pool only increases the training time by 16% compared with the basic ResNet, even with four S3Pool layers.

4.3. Visualization

Despite the convenient visualization of stochastic spatial sampling in the pixel space as shown in Figure 1, it is still unclear whether the same intuition holds when S3Pool is used in higher layers, and/or several S3Pool layers are stacked in a deep CNN. To this end, we obtain a trained NIN with two S3Pool layers as specified in Section 4.1, fix all the weights below the second S3Pool layer, turn off the stochasticity (i.e., using the test model of S3Pool) and stack a deconvolutional network [27] on top. The output of the deconvolutional network is then trained to reconstruct the inputs from the training set of CIFAR-10 in a deterministic way. After training, we can sample reconstructions from the deconvolutional network with stochasticity. The results are shown in Figure 4, where in the left column we show 50 images from the testing set, and each row shows the first 5 images from each of the 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. The second column shows the reconstructions produced by the deconvolutional network with the test mode of S3Pool (no sampling). The third column shows the a single draw of the reconstructions from the network with S3Pool layers. Note that the third column gives different reconstructions at each run of the deconvolutional network, due to its stochastic nature.

It is noticed that by turning off the stochastic spatial sampling (second column), the deconvolutional network is able to faithfully reconstruct the shape and the location of the objects, subject to reduced image details. The reconstructions from the network with S3Pool are also visually meaningful, even with strong stochasticity (in this case, the grid sizes are set to 16 and 8 for the two S3Pool layers). In particular, most reconstructions correspond to recognizable objects with various spatial distortions: local rescaling, translation, and etc.. Also note that these distortions do not follow a fixed pattern, thus can not be easily obtained by applying a basic geometric transform to the images directly. Therefore, the benefit of S3Pool can be understood as, during training, instead of using samples from the training set directly (first column in Figure 4), the S3Pool layers sample locally distorted features (third column in Figure 4) which are used implicitly for training. This corresponds to an aggressive data augmentation, which can significantly improve the generalization ability. The observation agrees with the results in Table 2 and Table 3, where S3Pool outperforms all image cropping augmented baselines, as image cropping can be considered as a much milder data augmentation than S3Pool.

5. Conclusions

We proposed S3Pool, a novel pooling method for CNNs. S3Pool extends the standard max pooling by decomposing pooling into two steps: max pooling with stride 1 and a non-deterministic spatial downsampling step by randomly sampling rows and columns from a feature map. In effect, S3Pool implicitly augments the training data at each pooling stage which enables superior generalization ability of the learned model. Extensive experiments on CIFAR-10 and CIFAR-100 have demonstrated that, S3Pool, either used in conjunction with data augmentation or not, significantly outperforms standard max pooling, dropout, and an existing stochastic pooling approach. In particular, by adjusting the level of stochasticity introduced by S3Pool using a simple mechanism, we obtained state-of-art result on STL-10. Additionally, S3Pool is simple to implement and introduces little computational overhead compared to general max pooling, which makes it a desirable design choice for learning deep CNNs.

References

[1] A. Coates and A. Y. Ng. Selecting receptive fields in deep networks. In *NIPS*, 2011. 2

[2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Workshop*, 2004. 2

[3] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014. 8

[4] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *ICML*, 2013. 2

[5] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014. 2

[6] K. Grauman and T. Darrell. Pyramid match kernels: Discriminative classification with sets of image features. In *ICCV*, 2005. 2

[7] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *MLKDD*, 2014. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016. 2, 5

[10] J. R. Higgins. *Sampling theory in Fourier and signal analysis: foundations*. Oxford University Press on Demand, 1996. 1

[11] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cats visual cortex. *The Journal of Physiology*, 160:106–154, 1962. 2

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 5

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 4

[14] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 2

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[16] C. Lee, P. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016. 2

[17] M. Lin, Q. Chen, and S. Yan. Network in network. *ICLR*, 2014. 2, 5, 6

[18] X. Lu, Z. Lin, X. Shen, R. Mech, and J. Wang. Deep multi-patch aggregation network for image style, aesthetics, and quality estimation. In *ICCV*, 2015. 2

[19] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 1949. 1

[20] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR*, 2003. 1, 4

[21] J. T. Springenberg and M. Riedmiller. Improving deep neural networks with probabilistic maxout units. *ICLR Workshop*, 2014. 2

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 2

[23] G. Xie, X. Zhang, X. Shu, S. Yan, and C. Liu. Task-driven feature pooling for image classification. In *ICCV*, 2015. 2

[24] S. Yang, P. Luo, C. C. Loy, K. W. Shum, and X. Tang. Deep representation learning with target coding. In *AAAI*, 2015. 8

[25] M. D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 5

[26] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *ICLR*, 2013. 2, 3, 4, 5, 6, 7, 8

[27] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*, 2010. 8

[28] J. Zhao, M. Mathieu, R. Goroshin, and Y. Lecun. Stacked what-where auto-encoders. *ICLR Workshop*, 2016. 8