Generalized Stochastic Pooling

Davide Bacciu and Francesco Landolfi

Università di Pisa - Department of Computer Science
Largo Bruno Pontecorvo, 3, 56127, Pisa - Italy

Abstract. Pooling layers play a critical role in Convolutional Neural Networks by reducing spatial dimensions and enhancing translation invariance. While conventional methods like max pooling and average pooling are effective, they can respectively amplify noise or dilute important features. Stochastic pooling introduces probabilistic sampling to improve generalization but is susceptible to biases from outliers, often mimicking max pooling in such cases. To address these limitations, we propose a generalization of stochastic pooling that introduces a tunable parameter to control the balance between uniform sampling, stochastic pooling, and max pooling. Experiments on multiple datasets demonstrate that uniform sampling outperforms the biased one, achieving a favorable trade-off between regularization and performance.

1 Introduction

Convolutional Neural Networks (CNNs) [1, 6] have revolutionized the field of deep learning by providing state-of-the-art performance in various domains such as computer vision, natural language processing, and medical imaging. A hallmark of CNN architectures is their ability to effectively process and extract hierarchical features from high-dimensional data through a combination of convolutional, pooling, and fully connected layers. Among these components, pooling layers are integral to achieving translation invariance, reducing spatial dimensions, and enhancing computational efficiency.

Pooling operations, such as max pooling [9] and average pooling [1, 6], have been instrumental in CNNs. Max pooling selects the most prominent feature within a region, while average pooling computes the mean value, thus offering distinct benefits and drawbacks. Max pooling emphasizes strong activations but risks losing subtle details and propagating noise, whereas average pooling smooths feature maps at the cost of diluting salient features. To address these limitations, alternative pooling strategies have been proposed, including stochastic variants, which introduce randomness in feature selection to improve generalization and mitigate overfitting [2, 10, 11].

Stochastic pooling [10] samples features based on their relative magnitudes, striking a balance between max and average pooling by allowing all features within a region to influence the output probabilistically. However, the effectiveness of stochastic pooling is constrained by its inherent sensitivity to extreme values, which can cause it to mimic max pooling behavior under certain conditions.

This paper introduces a generalization of stochastic pooling that incorporates a tunable parameter, α, to control the degree of randomness and mitigate overreliance on dominant features. By varying α, the method transitions smoothly between uniform sampling, standard stochastic pooling, and max pooling, enabling greater flexibility and adaptability to diverse learning tasks. We systematically evaluate this approach on benchmark datasets and demonstrate its capacity to enhance model performance and robustness compared to existing pooling strategies.

2 Generalized Stochastic Pooling

The architecture of a typical CNN consists of three primary components: convolutional layers, performing convolution operations on the input data to extract spatial and temporal features, pooling layers, which are used to downsample feature maps generated by convolutional layers, and fully connected layers, that make the final predictions or classifications.

Pooling functions as a sliding window over the input data, aggregating local regions into single representative values. One of the most common aggregation is *max pooling* [9], which selects the maximum value from each region of the feature map. Formally, for an input feature map $X \in \mathbb{R}^{C\times H\times W}$, where C is the number of channels and H and W are the height and width, the max pooling operation is defined as $Y_{cij} = \max_{(m,n)\in\mathcal{R}_{ij}} X_{cmn}$, where $\mathcal{R}_{ij}$ represents the region of size $k \times k$ with top-left corner centered at (ik, jk) in the input feature map, where k is the kernel size, and $Y \in \mathbb{R}^{C\times H'\times W'}$ is the downsampled output feature map, of height $H' = \lfloor H/k \rfloor$ and width $W' = \lfloor W/k \rfloor$. Another common aggregation is *average pooling* [1, 6], which computes instead the mean value of each region in the feature map. This aggregation is defined as $Y_{cij} = |\mathcal{R}_{ij}|^{-1} \sum_{(m,n)\in\mathcal{R}_{ij}} X_{cmn}$, where $|\mathcal{R}_{ij}|$ is the total number of elements in the pooling region $\mathcal{R}_{ij}$. Max pooling and average pooling have inherent limitations that can hinder model performance. While max pooling may amplify noise and overlook subtler features, average pooling tends to dilute significant features, leading to the loss of critical details in tasks requiring high sensitivity or fine-grained distinctions.

Stochastic pooling. To overcome these issues, Zeiler and Fergus [10] proposed *stochastic pooling*, a probabilistic pooling method inspired by max pooling which introduces randomness during the downsampling process, which can help improve generalization and reduce overfitting. In stochastic pooling, instead of taking the maximum or mean value, a probability is assigned to each feature in the pooling window, and a single value is sampled according to these probabilities. Formally, given a pooling region $\mathcal{R}_{ij}$, the probability p_{cmn} of selecting a value X_{cmn} for a given channel c is proportional to its magnitude, that is

$$p_{cmn} = \frac{X_{cmn}}{\sum_{(u,v)\in\mathcal{R}_{ij}} X_{cuv}}. \tag{1}$$

Notice that this normalization is well-defined since the authors assume that all the entries of the feature map X are positive. This is true for example whenever the feature map is passed through a sigmoid or a rectified linear unit activation function. Notice also that the normalization in Eq. (1) is performed channel-wise.

The output of the stochastic pooling operation is then sampled as

$$Y_{cij} = X_{cmn} \qquad \text{where} \qquad (m,n) \sim \text{Categorical}(\{p_{cuv}\}_{(u,v)\in\mathcal{R}_{ij}}).$$

That is, every given pair of indices (u, v) is sampled with probability p_{cuv} from the categorical distribution. Notice again that the samples are drawn independently for each channel.

At inference time, stochastic pooling needs to provide consistent and deterministic results since random sampling during prediction can lead to non-reproducible outcomes. Similarly to dropout [8], instead of sampling a value randomly, the output of each pooling region is set to the expected value of the distribution used for stochastic pooling. This ensures inference to be deterministic and aligned with the learned probability distribution from training. Hence, a weighted average is performed as follows:

$$\bar{Y}_{cij} = \mathrm{E}[Y_{cij}] = \sum_{(m,n)\in\mathcal{R}_{ij}} p_{cmn} \cdot X_{cmn}, \tag{2}$$

where $\bar{Y} \in \mathbb{R}^{C\times H'\times W'}$ is the downsampled output feature map at inference time. This probabilistic weighting can be seen as a form of model averaging: during training, stochastic pooling introduces randomness by sampling values from the feature distributions of pooling regions. This randomness effectively trains the model on slightly different representations of the same input, akin to training an ensemble of models with different subsets of the data. At inference time, instead, replacing the random sampling with the expected value aggregates the contributions of all possible sampled outputs in a weighted manner, where the weights correspond to their probabilities. This mirrors how model averaging combines predictions from multiple models to produce a single, stable output.

Generalized stochastic pooling. When the feature maps contain an outlier with a high magnitude, stochastic pooling behaves similarly to max pooling because the probability of selecting the outlier becomes disproportionately large. This phenomenon arises from the probability distribution used in stochastic pooling, where each value's likelihood is proportional to its magnitude: if one a feature map $M = \max_{(m,n)\in\mathcal{R}_{ij}} X_{cmn}$ in the pooling region is much larger than the others, the probability of selecting it will approach 1, as the relative contributions of smaller values become negligible. In this case, stochastic pooling is almost certain to select $Y_{cij} = M$, mimicking max pooling behavior.

We can study this phenomenon more in depth by introducing in Eq. (1) a *randomness temperature* α, which regulates how much a stochastic pooling layer behaves as a max pooling one. This can be done as follows:

$$p^{(\alpha)}_{cmn} = \frac{|X_{cmn}|^{\alpha}}{\sum_{(u,v)\in\mathcal{R}_{ij}} |X_{cuv}|^{\alpha}}. \tag{3}$$

Here we also applied the absolute value to the feature maps to drop the positiveness assumption and, to avoid division by zero, we also clamp every entry of X with a lowerbound of a small $\epsilon = 10^{-12}$. Furthermore, to avoid numerical errors, in practice Eq. (3) is equivalently computed as

$$p^{(\alpha)}_{cmn} = \frac{(|X_{cmn}|/M)^{\alpha}}{\sum_{(u,v)\in\mathcal{R}_{ij}} (|X_{cuv}|/M)^{\alpha}} \qquad \text{with} \qquad M = \max_{(u,v)\in\mathcal{R}_{ij}} |X_{cuv}|.$$

One can easily see that, for $\alpha = 0$, we have a uniform sampling distribution, with $p^{(0)}_{cmn} = |\mathcal{R}_{ij}|^{-1}$ and that, at inference time, Eq. (2) reduces to classical average pooling; for $\alpha = 1$, we obtain a standard stochastic pooling; and for $\alpha \to \infty$ we have

$$\lim_{\alpha \to \infty} p^{(\alpha)}_{cmn} = \begin{cases} \dfrac{1}{|S_M|} & \text{if } |X_{cmn}| = M, \\ 0 & \text{otherwise,} \end{cases}$$

where $S_M = \{(u, v) \in \mathcal{R}_{ij} \mid |X_{cuv}| = M\}$. Thus, during training, only the entries having the maximum absolute value will be sampled. During inference, instead, Eq. (2) will reduce to $\bar{Y}_{cij} = M$. Hence, for $\alpha \to \infty$, stochastic pooling will behave exactly as max pooling whenever the input feature maps are positive.

In the following, we will experimentally demonstrate that any choice of $\alpha > 0$ (including $\alpha = 1$, which corresponds to the original stochastic pooling setting) adversely affects the model's performance. This choice causes the model to behave more like max pooling, thereby inheriting its limitations and becoming more prone to overfitting.

3 Experiments

We tested stochastic pooling with varying values of $\alpha \in \{0, 1, 2, 4, 8\}$ on the CIFAR-10, CIFAR-100 [5], and SVHN [7] datasets. We also tested several baselines, namely strided convolution (i.e., downsampling with no pooling), average pooling, max pooling, L2-norm pooling [4], fractional max pooling [2], and S3 pooling [11]. In the experiments, we used a ResNet-like [3] architecture as a single backbone model with different pooling layers depending on the selected baseline. The architecture consists of 4 convolutional blocks of hidden dimensions $(32, 64, 128, 256)$ interleaved by 3 pooling layers of kernel size 2 each, followed by a global pooling layer of the same type of the local ones and, finally, by a linear layer. The convolutional blocks consist of 2 residual blocks of 2 convolutional layers each. For fractional max pooling, we used instead a reduction ratio of $1/\sqrt{2}$ to introduce randomness (with a ratio of $1/2$ it would behave as max pooling). We trained all models for 100 epochs using batch size 128 with learning rate 0.001 and cross-entropy loss. We chose this setting during a preliminary evaluation on a validation set (20% of the training set).

As a preliminary experiment, we repeatedly trained for 10 runs stochastic pooling with different values of α on CIFAR-10 and plotted the average training and validation curves on Fig. 1 ($\alpha = \infty$ means max pooling). We can clearly see from the picture that the performance slowly degrades as α increases, with $\alpha = 8$ obtaining almost the same performance of max pooling. It is also clear from the picture how lower values of α mitigate overfitting, as the difference between the validation and the training errors reduces with α, suggesting that the randomness introduced by stochastic pooling acts as a form of regularization of the model.

In Table 1, we present the average classification error (calculated over 10 train/test runs) and the average forward pass time (measured over 100 runs) for stochastic pooling and all baseline models. The results align with the findings from

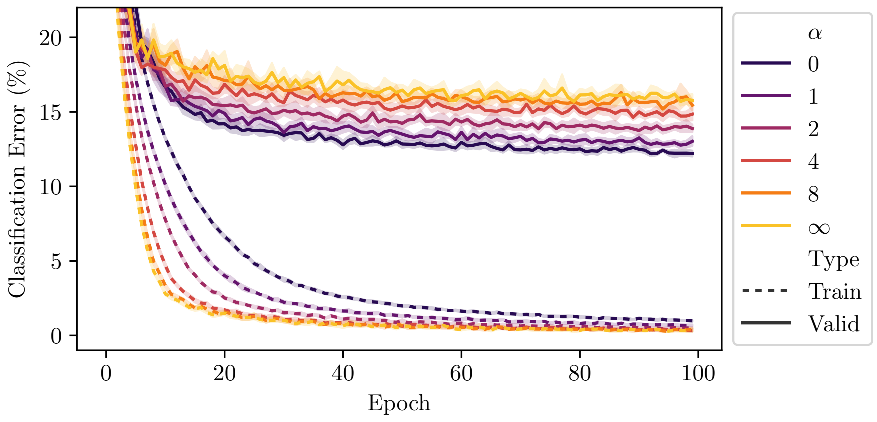

Fig. 1: Train and validation error on CIFAR-10 using different values of α.

	CIFAR10	CIFAR100	SVHN	time/batch
Strided	17.94 ± 0.37	50.76 ± 1.53	5.49 ± 0.25	3.59 ± 0.62
AvgPool	13.42 ± 0.38	43.17 ± 0.54	4.56 ± 0.20	3.63 ± 0.25
L2Pool	14.42 ± 0.33	46.46 ± 0.84	5.13 ± 0.26	4.07 ± 0.30
FractionalMaxPool	11.73 ± 0.28	39.82 ± 0.68	4.06 ± 0.13	3.93 ± 0.37
S3Pool	11.18 ± 0.24	39.04 ± 0.49	3.53 ± 0.14	4.52 ± 0.40
MaxPool+S3Pool	14.05 ± 5.86	43.78 ± 0.45	4.02 ± 0.15	4.81 ± 0.41
StochasticPool ($\alpha = 0$)	11.39 ± 0.25	39.85 ± 0.76	3.83 ± 0.17	6.13 ± 0.11
StochasticPool ($\alpha = 1$)	12.21 ± 0.50	40.39 ± 0.50	4.26 ± 0.18	7.21 ± 0.28
StochasticPool ($\alpha = 2$)	12.83 ± 0.42	41.80 ± 0.62	4.57 ± 0.17	7.26 ± 0.49
StochasticPool ($\alpha = 4$)	13.56 ± 0.36	43.78 ± 0.72	4.75 ± 0.20	7.29 ± 0.54
StochasticPool ($\alpha = 8$)	14.68 ± 0.56	45.27 ± 0.73	5.15 ± 0.19	7.24 ± 0.24
MaxPool ($\alpha = \infty$)	14.88 ± 0.31	45.63 ± 0.86	5.10 ± 0.16	3.68 ± 0.24

Table 1: Classification error (%, mean ± std) and time required for a forward pass (ms, mean ± std) using different pooling strategies.

the preliminary experiments, showing that the classification error increases as α grows. Stochastic pooling achieves its best performance at $\alpha = 0$, outperforming the original setting at $\alpha = 1$. Conversely, the worst performance is observed at $\alpha = \infty$, corresponding to max pooling. Among all models, the best performance is achieved by S3 pooling, which slightly outperforms stochastic pooling. We attribute this improvement primarily to the fact that S3 pooling, by selecting random rows and columns, effectively performs a form of uniform pixel sampling. However, unlike stochastic pooling, the selected coordinates are consistent across all channels, thereby avoiding a "shuffling" effect of feature maps across different pixels. It is worth noting, however, that the authors of S3 pooling included a max pooling layer before their proposed downsampling method, which, as shown in Table 1, resulted in worse performance. In terms of forward pass time, we observed that stochastic pooling requires more time compared to the baseline models, with the $\alpha = 0$ setting being slightly faster due to its more efficient implementation. However, it is important to note that, apart from S3 pooling (which we reimplemented), all other baseline methods benefit from native C++/CUDA implementations, whereas our implementation of stochastic pooling is currently in Python. Furthermore, stochastic pooling with $\alpha = 0$ at inference time is as fast as average pooling, which is the second fastest-performing model.

4 Conclusions

In this work, we introduced a generalized stochastic pooling that incorporates a tunable parameter α to control the balance between uniform sampling, stochastic pooling, and max pooling. Through extensive experiments on CIFAR-10, CIFAR-100, and SVHN datasets, we demonstrated that lower values of α mitigate overfitting, improve generalization, and outperform the original stochastic pooling ($\alpha = 1$), as well as many established pooling methods. Our results highlight the versatility of stochastic pooling, which adapts to varying task requirements by modulating randomness in the pooling process. The performance gains and increased adaptability make this approach a valuable addition to the toolbox for CNN design, paving the way for further advancements in pooling strategies.

References

[1] K. Fukushima. “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position”. In: *Biological Cybernetics* 36.4 (1980).

[2] B. Graham. *Fractional Max-Pooling.* 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition.* version: 1. 2015.

[4] O. J. Hénaff and E. P. Simoncelli. *Geodesics of learned representations.* 2016.

[5] A. Krizhevsky, G. Hinton, and others. “Learning multiple layers of features from tiny images”. In: (2009). Publisher: Toronto, ON, Canada.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998). Conference Name: Proceedings of the IEEE.

[7] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, and others. “Reading digits in natural images with unsupervised feature learning”. In: *NIPS workshop on deep learning and unsupervised feature learning.* Vol. 2011. Issue: 2. Granada, 2011.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014).

[9] J. Weng, N. Ahuja, and T. Huang. “Learning recognition and segmentation of 3-D objects from 2-D images”. In: *1993 (4th) International Conference on Computer Vision.* 1993 (4th) International Conference on Computer Vision. 1993.

[10] M. Zeiler and R. Fergus. “Stochastic Pooling for Regularization of Deep Convolutional Neural Networks”. In: (2013).

[11] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris. “S3Pool: Pooling With Stochastic Spatial Sampling”. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.