Refining activation downsampling with SoftPool

Alexandros Stergiou
Utrecht University
a.g.stergiou@uu.nl

Ronald Poppe
Utrecht University
r.w.poppe@uu.nl

Grigorios Kalliatakis
University of Warwick
grigorios.kalliatakis@warwick.ac.uk

Abstract

Convolutional Neural Networks (CNNs) use pooling to decrease the size of activation maps. This process is crucial to increase the receptive fields and to reduce computational requirements of subsequent convolutions. An important feature of the pooling operation is the minimization of information loss, with respect to the initial activation maps, without a significant impact on the computation and memory overhead. To meet these requirements, we propose SoftPool: a fast and efficient method for exponentially weighted activation downsampling. Through experiments across a range of architectures and pooling methods, we demonstrate that SoftPool can retain more information in the reduced activation maps. This refined downsampling leads to improvements in a CNN's classification accuracy. Experiments with pooling layer substitutions on ImageNet1K show an increase in accuracy over both original architectures and other pooling methods. We also test SoftPool on video datasets for action recognition. Again, through the direct replacement of pooling layers, we observe consistent performance improvements while computational loads and memory requirements remain limited[1].

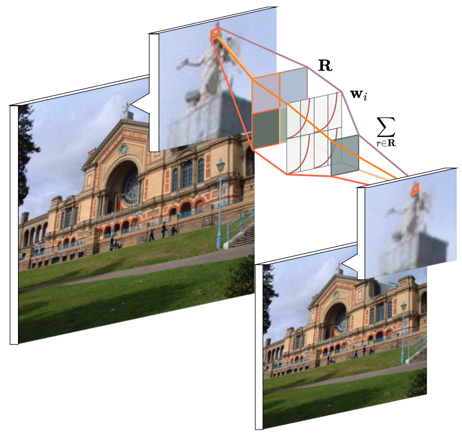

Figure 1. **SoftPool illustration**. The original image is sub-sampled with a 2×2 ($k=2$) kernel. The output is based on the exponentially weighted sum of the original pixels within the kernel region. This can improve the representation of high-contrast regions, present around object edges or specific feature activations.

1. Introduction

Pooling layers are essential in convolutional neural networks (CNNs) to decrease the size of activation maps. They reduce the computational requirements of the network while also achieving spatial invariance, and increase the receptive field of subsequent convolutions [6, 27, 43].

A range of pooling methods has been proposed, each with different properties (see Section 2). Most architectures use maximum or average pooling, both of which are fast and memory-efficient but leave room for improvement in terms of retaining important information in the activation map.

We introduce *SoftPool*, a kernel-based pooling method that uses the softmax weighted sum of activations. We demonstrate that SoftPool largely preserves descriptive activation features, while remaining computationally and memory-efficient. Owing to better feature preservation, models that include SoftPool consistently show improved classification performance compared to their original implementations. We make the following contributions:

- We introduce SoftPool: a novel pooling method based on softmax normalization that can be used to downsample 2D (image) and 3D (video) activation maps.
- We demonstrate how SoftPool outperforms other pooling methods in preserving the original features, measured using image similarity.
- Experimental results on image and video classification tasks show consistent improvement when replacing the original pooling layers by SoftPool.

The remainder of the paper is structured as follows. We first discuss related work on feature pooling. We then detail SoftPool (Section 3) and evaluate it in terms of feature loss and image and video classification performance over multiple pooling methods and architectures (Section 4).

[1]Code is available at: http://www.tinyurl.com/softpool

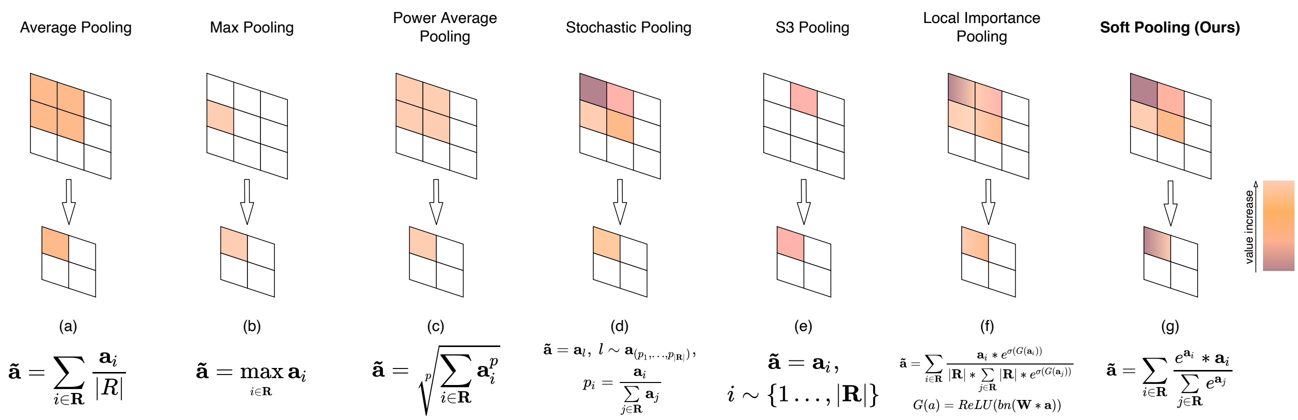

Figure 2. **Pooling variants**. $\mathbf{R}$ is the set of pixel values in the kernel neighborhood. (a,b) **Average and maximum pooling** are based on averaging and maximum activation selection in a kernel. (c) **Power Average pooling** (L_p) [10, 14] is proportional to average pooling raised to a power (p). The output is equal to max-pool for $p \to \infty$, and sum pooling when $p = 1$. (d) **Stochastic pooling** [47] outputs a randomly selected activation from the kernel region. (e) **Stochastic Spatial Sampling** (S3Pool) [48] samples random horizontal and vertical regions given a specified stride. (f) **Local Importance Pooling** (LIP) [12] uses a trainable sub-net G to enhance specific features. (g) **SoftPool** (ours) exponentially weighs the effect of activations using a softmax kernel.

2. Related Work

Pooling for hand-crafted features. Downsampling is a widely used technique in hand-coded feature extraction methods. In Bag-of-Words (BoW) [7], images were viewed as collections of local patches, pooled and encoded as vectors [40]. Combinations with Spatial Pyramid Matching [21] aimed to preserve spatial information. Later works considered the selection of the maximum SIFT features in a spatial region [44]. Pooling has primarily been linked to the use of max-pooling, because of the feature robustness of biological max-like cortex signals [31]. Boureau *et al.* [2] studied maximum and average pooling in terms of their robustness and usability, and found max-pooling to produce representative results in low feature activation settings.

Pooling in CNNs. Pooling has also been adapted to learned feature approaches, as seen in early works in CNNs [22]. The main benefit of pooling has traditionally been the creation of condensed feature representations which reduce the computational requirements and enable the creation of larger and deeper architectures [32, 36].

Recent efforts have focused on preserving relevant features during downsampling. An overview of a number of popular pooling methods appears in Figure 2. Initial approaches include stochastic pooling [47], which uses a probabilistic weighted sampling of activations within a kernel region. Mixed pooling based on maximum and average pooling has been used either probabilistically [45] or through a combination of portions from each method [23]. Based on the combination of averaging and maximization, Power Average (L_p) pooling [10, 14] utilizes a learned parameter p to determine the relative importance of both methods. When $p = 1$, the local sum is used, while $p \to \infty$ corresponds to max-pooling. More recent approaches have considered grid-sampling methods. In S3Pool [48], the downsampled outputs stem from randomly sampling the rows and columns of the original feature map grid. Methods that depend on learned weights include Detail Preserving Pooling (DPP, [30]) that uses average pooling while enhancing activations with above-average values. Local Importance Pooling (LIP, [12]) utilizes learned weights as a subnetwork attention-based mechanism. Other learned pooling approaches such as Ordinal Pooling [8], which order kernel pixels in discerningly and assigning them trainable weights. More recently, Zhao and Snoek [50] proposed a pooling technique named LiftPool based on the use of four different learnable sub-bands of the input. The produced output is composed by a mixture of the discovered sub-bands.

Most of the aforementioned methods rely on different combinations of maximum and average pooling. Instead of combining existing methods, our work is based on a softmax weighting approach to preserve the basic properties of the input while amplifying feature activations of greater intensity. SoftPool does not require trainable parameters, thus is independent to the training data used. Moreover, it is significantly more computational and memory efficient compared to learned approaches. In contrast to max-pooling, our approach is differentiable. Gradients are obtained for each input during backpropagation, which improves neural connectivity during training. Through the weighted softmax, pooled regions are also less susceptible to vanishing local kernel activations, a common issue with average pooling. We demonstrate the effects of SoftPool in Figure 3, where the zoomed-in regions show that features are not

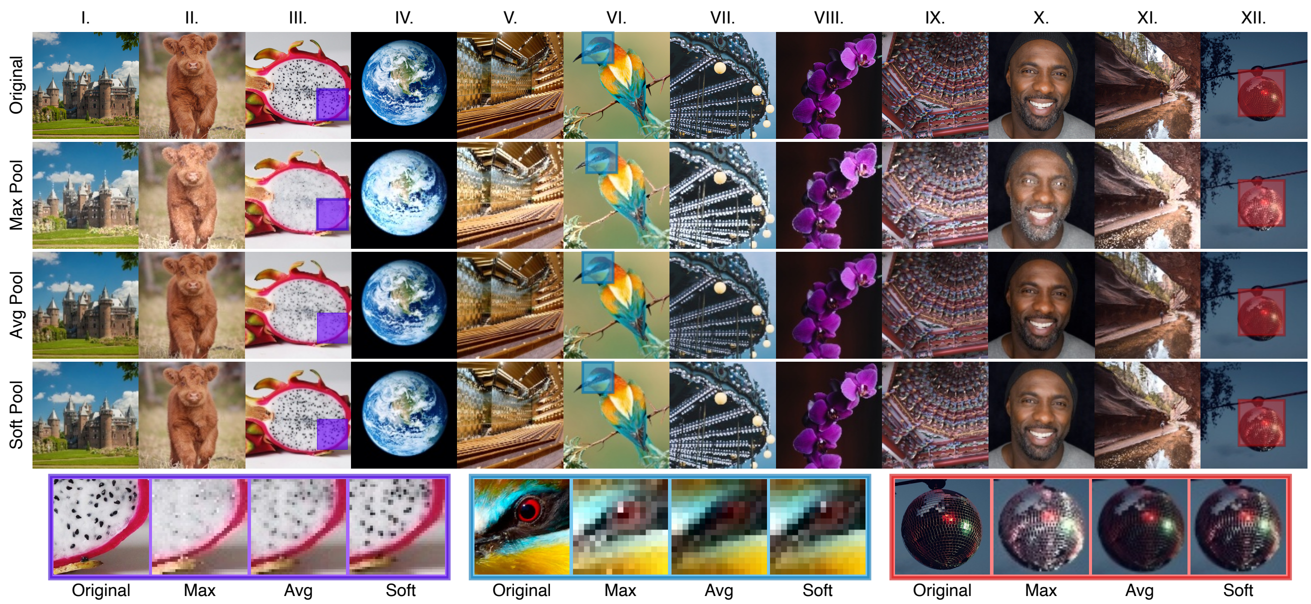

Figure 3. **Examples of maximum, average and soft pooling**. Images are 1200×1200 pixels with the $3\times$ pooled equivalents reduced to 12.5% of the original size. Image selection was based on overall contrast (I, III, V, VIII, XII), color (II, IV, VI, IX), detail (III, V, X, XI, XII) and texture (II, VII). Images on the bottom row are zoomed-in regions. A high resolution version of the regions appears alongside a detailed discussion in §1 of the Supplementary Material.

completely lost as with hard-max selection, or suppressed by the overall region through averaging.

3. SoftPool Downsampling

We start by formally introducing the forward flow of information in SoftPool and the gradient calculation during backpropagation. We consider a local region ($\mathbf{R}$) in an activation map ($\mathbf{a}$) with dimension $C \times H \times W$ with C the number of channels, H the height and W the width of the activation map. For simplicity of notation, we omit the channel dimension and assume that $\mathbf{R}$ is the set of indices corresponding to the activations in the 2D spatial region under consideration. For a pooling filter of size $k \times k$, we consider $|\mathbf{R}| = k^2$ activations. The output of the pooling operation is $\tilde{\mathbf{a}}_R$ and the corresponding gradients are denoted with $\nabla \tilde{\mathbf{a}}_i$.

3.1. Exponential maximum kernels

SoftPool is influenced by the cortex neural simulations of Riesenhuber and Poggio [28] as well as the early pooling experiments with hand-coded features of Boureau *et al.* [2]. The proposed method is based on the natural exponent (e) which ensures that large activation values will have greater effect on the output. The operation is differentiable, which implies that all activations within the local kernel neighborhood will be assigned a proportional gradient, of at least a minimum value, during backpropagation. This is in contrast to pooling methods that employ hard-max or average pooling. SoftPool utilizes the smooth maximum approximation of the activations within kernel region $\mathbf{R}$. Each activation $\mathbf{a}_i$ with index i is applied a weight $\mathbf{w}_i$ that is calculated as the ratio of the natural exponent of that activation with respect to the sum of the natural exponents of all activations within neighborhood $\mathbf{R}$:

$$\mathbf{w}_i = \frac{e^{\mathbf{a}_i}}{\sum\limits_{j \in \mathbf{R}} e^{\mathbf{a}_j}} \tag{1}$$

The weights are used as non-linear transforms in conjunction with the value of the corresponding activation. Higher activations become more dominant than lower-valued ones. Because most pooling operations are performed in high-dimensional feature spaces, highlighting the activations with greater effect is a more balanced approach than simply selecting the average or maximum. In the latter case, discarding the majority of the activations presents the risk of losing important information. Conversely, an equal contribution of activations in average pooling can correspond to local intensity reductions by considering the overall regional feature intensity equally.

The output value of the SoftPool operation is produced through a standard summation of all weighted activations within the kernel neighborhood $\mathbf{R}$:

$$\tilde{\mathbf{a}} = \sum_{i \in \mathbf{R}} \mathbf{w}_i * \mathbf{a}_i \tag{2}$$

In comparison to other max- and average-based pooling approaches, using the softmax of regions produces normalized results with a probability distribution proportional to the values of each activation with respect to the neighboring activations for the kernel region. This is in direct contrast to popular maximum activation value selection or averaging all activations over the kernel region, where the output activations are not regularized. A full forward and backward information flow is shown in Figure 4.

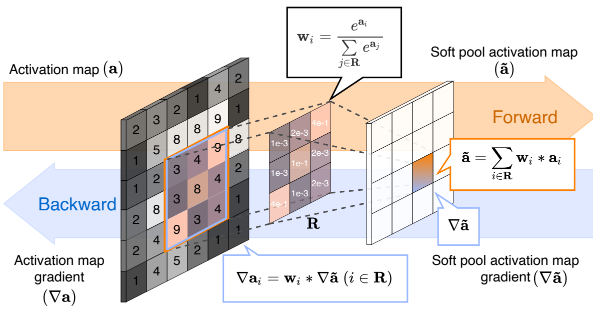

Figure 4. **SoftPool calculation**. In forward operation, in orange, the kernel uses the exponential softmax value of each activation as weight and calculates the weighted sum for region $\mathbf{R}$. These weights are also used for the gradients ($\nabla\tilde{\mathbf{a}}_i$), in blue. Activation gradients are proportional to the calculated softmax weights.

3.2. Gradient calculation

During the update phase in training, gradients of all network parameters are updated based on the error derivatives calculated at the proceeding layer. This creates a chain of updates, when backpropagating throughout the entire network architecture. In SoftPool, gradient updates are proportional to the weights calculated during the forward pass.

As softmax is differentiable, unlike maximum or stochastic pooling methods, during backpropagation, a minimum non-zero weight will be assigned to every positive activation within a kernel region. This enables the calculation of a gradient for every non-zero activation in that region, as shown in Figure 4.

In our implementation of SoftPool, we use finite ranges of possible values given a precision level (i.e., half, single or double) as detailed in §7 of the Supplementary Material. We retain the differentiable nature of softmax by assigning a lower arithmetic limit given the number of bits used by each type preventing arithmetic underflow.

3.3. Feature preservation

An integral goal of sub-sampling is the preservation of representative features in the input, while simultaneously minimizing the overall resolution. Creating unrepresentative downsampled versions of the original inputs can be harmful to the overall model's performance as the representation of the input is detrimental for the task.

Currently widely used pooling techniques can be ineffective in certain cases, as shown in Figures 3 & 5. Average pooling decreases the effect of all activations in the region equally, while max pooling selects only the single highest activation in the region. SoftPool falls between the two, as all activations in the region contribute to the final output, with higher activations are more dominant than lower ones. This balances the effects of both average and max pooling, while leveraging the beneficial properties of both.

3.4. Spatio-temporal kernels

CNNs have also been extended to 3D inputs to include additional dimensions such as depth and time. To accommodate these inputs, we extend SoftPool to include an additional dimension. For an input activation map $\mathbf{a}$ of $C \times H \times W \times T$, with T the temporal extent, we transform the 2D spatial kernel region $\mathbf{R}$ to a 3D spatio-temporal region with an additional third temporal dimension.

The produced output holds condensed spatio-temporal information. Issues that arise with the introduction of the temporal dimension are discussed and illustrated in §3 of the Supplementary Material. With the added dimension, desired pooling properties such as limited loss of information, a differentiable function, and low computational and memory overhead are even more important.

4. Experimental Results

We first evaluate the information loss for various pooling operators. We compare the downsampled outputs to the original inputs using standard similarity measures (Section 4.2). We also investigate each pooling operator's computation and memory overhead (Section 4.3).

We then focus on the classification performance gain when using SoftPool in a range of popular CNN architectures and in comparison to other pooling methods (Section 4.4). We also perform an ablation study where we replace max-pooling operations by SoftPool operations in an InceptionV3 architecture (Section 4.5).

Finally, we demonstrate the merits of SoftPool for spatio-temporal data by focusing on action recognition in video (Section 4.7). Additionally, we investigate how transfer learning performance is affected when SoftPool is used.

4.1. Experimental settings

Datasets. For our experiments with images, we employ five different datasets for the tasks of image quality quantitative evaluation and classification. For image quality and similarity assessment we use high-resolution DIV2K [1], Urban100 [18], Manga109 [25], and Flicker2K [1]. ImageNet1K [29] is used for the classification task. For video-based action recognition, we use the large-scale HACS [49] and Kinetics-700 [3] datasets. Our transfer learning experiments are performed on the UCF-101 [33] dataset.

Training implementation details. For the image classification task, we perform random region selection of 294 $\times$

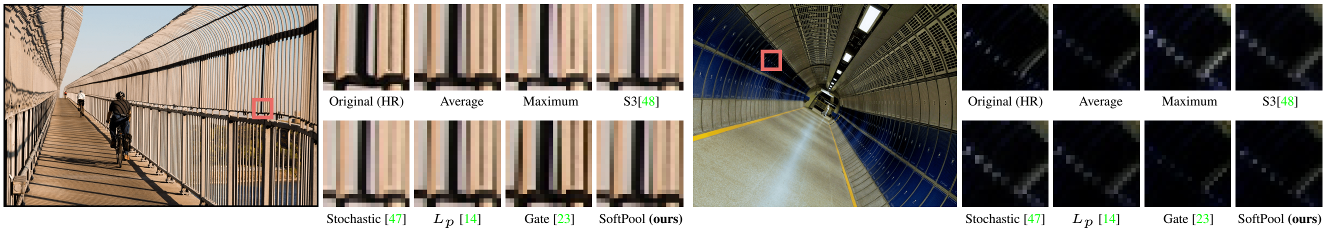

Figure 5. **Visual comparisons over pooling methods**. Both images are from Urban100 [18]. Left image (*img024*) shows high contrasting borders/edges. Right image (*img078*) presents the inverse of high local values within an overall low-valued region.

	Pooling method	DIV2K [1]						Urban100 [18]						Manga109 [25]					
		$k=2$		$k=3$		$k=5$		$k=2$		$k=3$		$k=5$		$k=2$		$k=3$		$k=5$	
		SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
	Average	0.714	51.247	0.578	44.704	0.417	29.223	0.691	50.380	0.563	41.745	0.372	28.270	0.695	54.326	0.582	43.657	0.396	29.862
	Maximum	0.685	49.826	0.370	41.944	0.358	22.041	0.662	48.266	0.528	40.709	0.330	20.654	0.671	50.085	0.544	41.128	0.324	22.307
	Pow-average	0.419	35.587	0.286	26.329	0.178	16.567	0.312	31.911	0.219	24.698	0.124	15.659	0.381	29.248	0.276	18.874	0.160	9.266
	Sum	0.408	35.153	0.268	26.172	0.193	17.315	0.301	31.657	0.208	24.735	0.123	15.243	0.374	30.169	0.271	20.150	0.168	13.081
Trainable	L_p [14]	0.686	49.912	0.542	43.083	0.347	25.139	0.676	48.508	0.534	39.986	0.326	26.365	0.675	51.721	0.561	41.824	0.367	27.469
	Gate [23]	0.689	50.104	0.560	43.437	0.353	25.672	0.675	49.769	0.537	40.422	0.328	26.731	0.679	51.980	0.569	42.127	0.374	27.754
	Lite-S3DPP [30]	0.702	50.598	0.562	44.076	0.396	27.421	0.684	49.947	0.551	40.813	0.365	27.136	0.691	52.646	0.573	42.794	0.386	28.598
	LIP [12]	0.711	50.831	0.559	44.432	0.401	28.285	0.689	50.266	0.558	41.159	0.370	27.849	0.689	53.537	0.579	43.018	0.391	29.331
Stoch.	Stochastic [47]	0.631	45.362	0.479	39.895	0.295	21.314	0.616	44.342	0.463	37.223	0.286	19.358	0.583	46.274	0.427	39.259	0.255	22.953
	S3 [48]	0.609	44.760	0.454	39.326	0.280	20.773	0.608	44.239	0.459	36.965	0.272	19.645	0.576	46.613	0.426	39.866	0.232	23.242
	SoftPool (Ours)	**0.729**	**51.436**	**0.594**	**44.747**	**0.421**	**29.583**	**0.694**	**50.687**	**0.578**	**41.851**	**0.394**	**28.326**	**0.704**	**54.563**	**0.586**	**43.782**	**0.403**	**30.114**

Table 1. **Quantitative results on benchmark high-res datasets**. Best results from each similarity measure are denoted in **bold**.

294 height and width, which was then resized to 224×224. We use an initial learning rate of 0.1 with an SGD optimizer and a step-wise learning rate reduction every 40 epochs for a total of 100 epochs. The epochs number was chosen as no further improvements were observed for any of the models. We also set the batch size to 256 across all models.

For our experiments on videos, we use a multigrid training scheme [42], with frame sizes between 4–16 and frame crops of 90–256 depending on the cycle. On average, the video inputs are of size $8 \times 160 \times 160$. With the multigrid scheme, the batch sizes are between 64 and 2048 with each size counter-equal to the input size in every step to match the memory use. We use the same learning rate, optimizer, learning rate schedule and maximum number of epochs as in the image-based experiments.

4.2. Downsampling similarity

We first assess the information loss of various pooling operations. We compare the original inputs with the downsampled outputs in terms of similarity. We use three of the most widely used kernel sizes (i.e., $k \times k$, with $k = \{2, 3, 5\}$). Our experiments are based on two standardized image similarity evaluation metrics [41]:

Structural Similarity Index Measure (SSIM) is used between the original and downsampled images. SSIM is based on the computation of a luminance, contrast and structural term. Larger index values correspond to larger structural similarities between the images compared.

Peak Signal-to-Noise Ratio (PSNR) measures the compression quality of the resulting image based on the Mean Squared Error (MSE) inverse between the weighted averages of their channels. PSNR depends on the MSE with higher values relate to lower errors between the two images.

Visual examples of different compression methods are shown in Figure 5. The proposed SoftPool method can represent regions with borders between low and high frequencies better than other methods, shown by the border of the black bar in the left image. The inverse also hold true for a high-frequency location within an overall low-frequency region as shown in the left image. In such cases, max and stochastic-based methods [14, 47, 48] over-amplify pixel locations while these pixels are completely lost with average and gate [23] approaches. In contrast, SoftPool shows the ability to preserve such patterns as also shown in Figure 6.

In Tables 1 and 3, we show the average SSIM and PSNR values obtained on DIV2K [1], Urban100 [18], Manga109 [25], and Flicker2K [1] high-resolution datasets over different kernel sizes. For both measures, SoftPool outperforms all other methods by a reasonable margin. Notably, it significantly outperforms non-trainable and stochastic methods. The randomized strategy of stochastic methods does not effectively allow their use as a standalone method as they lack non-linear operations. Trainable approaches are bounded by both the image types they have been trained on as well as on the discovered channel correlations during pooling.

4.3. Latency and memory use

Memory and latency costs of pooling operations are largely overlooked as a single operation has negligible latency times and memory consumption. However, because of the parallelization of deep learning models, operations may be performed thousands of times per step. Eventually,

Model	Params (M)	GFLOP	Original top-1	Original top-5	SoftPool (pre-train) top-1	SoftPool (pre-train) top-5	SoftPool (from scratch) top-1	SoftPool (from scratch) top-5
ResNet18	11.7	1.83	69.76	89.08	70.56 (+0.80)	89.89 (+0.81)	**71.27** (+1.51)	**90.16** (+1.08)
ResNet34	21.8	3.68	73.30	91.42	74.03 (+0.73)	91.85 (+0.43)	**74.67** (+1.37)	**92.30** (+0.88)
ResNet50	25.6	4.14	76.15	92.87	76.60 (+0.45)	93.15 (+0.28)	**77.35** (+1.17)	**93.63** (+0.76)
ResNet101	44.5	7.87	77.37	93.56	77.74 (+0.37)	93.99 (+0.43)	**78.32** (+0.95)	**94.21** (+0.65)
ResNet152	60.2	11.61	78.31	94.06	78.73 (+0.42)	94.47 (+0.41)	**79.24** (+0.92)	**94.72** (+0.66)
DenseNet121	8.0	2.90	74.65	92.17	75.27 (+0.57)	92.60 (+0.43)	**75.88** (+1.23)	**92.92** (+0.75)
DenseNet161	28.7	7.85	77.65	93.80	78.12 (+0.47)	94.15 (+0.35)	**78.72** (+0.93)	**94.41** (+0.61)
DenseNet169	14.1	3.44	76.00	93.00	76.49 (+0.49)	93.38 (+0.38)	**76.95** (+0.95)	**93.76** (+0.76)
ResNeXt50 32x4d	25.0	4.29	77.62	93.70	78.23 (+0.61)	93.97 (+0.27)	**78.48** (+0.86)	**93.37** (+0.67)
ResNeXt101 32x8d	88.8	7.89	79.31	94.28	78.89 (+0.58)	94.73 (+0.45)	**80.12** (+0.81)	**94.88** (+0.60)
Wide-ResNet50	68.9	11.46	78.51	94.09	79.14 (+0.63)	94.51 (+0.42)	**79.52** (+1.01)	**94.85** (+0.76)

Table 2. **Trained from scratch and pre-trained pairwise comparisons of top-1 and top-5 accuracies** on ImageNet1K between the original networks and the same networks with pooling replaced by SoftPool. Extensive runs are reporeted in the Supplementary Material.

Pooling	CPU (ms)	CUDA (ms)	Flicker2K[1]					
			$k=2$		$k=3$		$k=5$	
	(↓ F / ↑ B)	(↓ F / ↑ B)	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR
Avg	9 / 49	14 / 76	0.709	51.786	0.572	44.246	0.408	28.957
Max	91 / 152	195 / 267	0.674	47.613	0.385	40.735	0.329	21.368
Pow-avg	74 / 329	120 / 433	0.392	34.319	0.271	26.820	0.163	15.453
Sum	26 / 163	79 / 323	0.386	34.173	0.265	26.259	0.161	15.218
L_p [14]	116 / 338	214 / 422	0.683	48.617	0.437	42.079	0.341	24.432
Gate [23]	245 / 339	327 / 540	0.687	49.314	0.449	42.722	0.358	25.687
DPP [30]	427 / 860	634 / 1228	0.691	50.586	0.534	43.608	0.385	27.430
LIP [12]	134 / 257	258 / 362	0.696	50.947	0.548	43.882	0.390	28.134
Stoch. [47]	162 / 341	219 / 485	0.625	46.714	0.474	38.365	0.264	21.428
S3 [48]	233 / 410	345 / 486	0.611	46.547	0.476	37.706	0.252	21.363
SoftPool	31 / 156	56 / 234	**0.721**	**52.356**	**0.587**	**44.893**	**0.416**	**29.341**

Table 3. **Latency and pixel similarity**. Latency times for forward (↓ F) and backward (↑ B) are averaged over ImageNet1K [29].

Figure 6. **Results after SoftPool pooling with different kernel and stride sizes**. Image (`img003`) is from Urban100 [18].

a slow or memory-intensive pooling operation can have a detrimental effect on the performance.

To test the computation and memory overhead, we report running-time memory use and inference on both CPU and GPU (CUDA) in Table 3. We detail our testing environment and implementation in §7 of the Supplementary Material.

From Table 3 we observe that our implementation of SoftPool achieves low inference times on both CPU and CUDA operations, while remains memory-efficient. This is because of it's simplicity and ease of parallelization. SoftPool is second to only average pooling in terms of latency and memory use as operations can be performed in-place.

4.4. Classification performance on ImageNet1K

We investigate whether classification accuracy improves as a result of SoftPool's superior ability to retain information. We replace the pooling layers from ResNet [16], DenseNet [17], ResNeXt [43] and wide-ResNet [46] networks with SoftPool. We consider two distinct settings. In the *from scratch* setting, we replace the pooling operators of the original models by SoftPool and train with weights randomly initialized. In the *pre-trained* setting, we replace pooling layers of the original trained networks and evaluate the effects of the pooling strategy change without further training. Results of both settings appear in Table 2.

Networks trained from scratch with pooling layers replaced by SoftPool yield consistent accuracy improvements over the original networks. The same trend is also visible for the pre-trained networks for which the models have been trained with their original pooling methods. We now discuss the results per CNN architecture family.

ResNet [16]. By training from scratch with SoftPool, an average top-1 accuracy improvement of 1.17% is observed with a maximum of 1.51% on ResNet18. When replacing pooling layers on pre-trained networks, we obtain an average of +0.59% accuracy. All ResNet-based models only include a single pooling operation after the first convolution. Thus, results are based on a single layer replacement which emphasizes the merits of using SoftPool.

DenseNet [17]. Based on the DenseNet overall architecture that incorporates five pooling operations, we replace the max pooling operation that follows after the first layer and average pooling layers between Dense blocks with our proposed method. Top-1 accuracy gains are in the 0.93–1.23% range when training from scratch and 0.47–0.57% with substitution on pre-trained networks. Top-5 accuracy follows similar incremental trends.

Model	pooling replacement	statistic (χ^2)	p-value (ρ)
ResNet18	Max → SoftPool	90.08	$2.28e^{-21}$
ResNet34		15.04	$1.05e^{-4}$
ResNet50		61.80	$3.80e^{-15}$
ResNet101		23.13	$1.51e^{-6}$
ResNet152		50.29	$1.32e^{-12}$
DenseNet121	Avg+Max → SoftPool	411.02	2.19^{-91}
DenseNet161		46.52	$9.06e^{-12}$
DenseNet169		27.32	$1.72e^{-7}$
ResNeXt50 32x4d	Max → SoftPool	45.13	$1.85e^{-11}$
ResNeXt101 32x8d		695.87	$2.36e^{-53}$
wide-ResNet50	Max → SoftPool	39.41	$3.43e^{-10}$
InceptionV1	Max → SoftPool	106.41	$5.98e^{-25}$
InceptionV3		80.19	$3.39e^{-19}$

Table 4. **McNemar's [9, 26] statistical significance for pooling substitution results correlation**. The top-1 accuracies on ImageNet1K val set from Tables 2, 5 and 6 are used. A detailed view of the statistics is presented in §6 in the Supplementary Material.

Pooling	Networks					
	ResNet18 [16]	ResNet34 [16]	ResNet50 [16]	ResNeXt50 [43]	DenseNet121 [17]	InceptionV1 [17]
Original	(Max) 69.76	(Max) 73.30	(Max) 76.15	(Max) 77.62	(Avg+Max) 74.65	(Max) 69.78
Stochastic [47]	70.13	73.34	76.11	77.71	74.84	70.14
S3 [48]	70.15	73.56	76.24	77.82	74.85	70.17
L_p [23]	70.45	73.74	76.56	77.86	74.93	70.32
Gate [14]	70.74	73.68	76.75	77.98	74.88	70.52
DPP [30]	70.86	74.25	77.09	78.20	75.37	70.95
LIP [12]	70.83	73.95	77.13	78.14	75.31	70.77
SoftPool **(ours)**	**71.27**	**74.67**	**77.35**	**78.48**	**75.88**	**71.43**

Table 5. **Pooling layer substitution** top-1 accuracy for a variety of pooling methods. Experiments were performed on Imagenet1K.

ResNeXt [43]. Average of 0.83% top-1 accuracy improvement when trained from scratch. The best model, ResNeXt101 32x8d, achieves 80.12% top-1 accuracy (+0.81%) and 94.88% top-5 accuracy (+0.60%) with SoftPool. On the pre-trained settings, we note average improvements of +0.59% on top-1 and +0.36% top-5 accuracies. We again note these accuracies come by a replacement of their only pooling operation after the first convolution layer.

Wide-Resnet-50 [46]. We observe top-1 and top-5 accuracy increases of 1.01% and 0.76% when trained from scratch with SoftPool. The initialized network also achieves improvements with +0.63% top-1 and +0.42% top-5.

These combined experiments demonstrate that by replacing a single (ResNet, ResNeXt and Wide-ResNet) or only five pooling layers (DenseNet) with SoftPool lead to a modest but important increase in accuracy. To understand whether these improvements are statistically significant, we performed a McNemar's test [9, 26] to calculate the probabilities (ρ) of marginal homogeneity between the original and SoftPool-replaced networks. The results are summarized in Table 4 for the models obtained in the *from scratch* setting. As shown for all networks, $\rho \ll 0.01$ corresponding to a $\gg$ |99% confidence that the improved results are indeed due to the change in the pooling operations.

Layer	Pooling layer substitution with SoftPool							
	N	I	II	III	IV	V	VI	VII
$pool_1$		✓	✓	✓	✓	✓	✓	✓
$pool_2$			✓	✓	✓	✓	✓	✓
$mixed\ 5_{b-d}$				✓	✓	✓	✓	✓
$mixed\ 6_a$					✓	✓	✓	✓
$mixed\ 6_{b-e}$						✓	✓	✓
$mixed\ 7_a$							✓	✓
$mixed\ 7_{b-d}$								✓
Top-1 (%)	77.45	77.93	78.14	78.37	78.42	78.65	78.83	**79.04**
Top-5 (%)	93.56	93.61	93.68	93.74	93.78	93.84	93.90	**93.98**

Table 6. **Progressive layer substitution for InceptionV3**. Experiments are performed on ImageNet1K. Column numbers refer to the number of replaced pooling layers, marked with ✓.

Memory and computation requirements. We also summarize the number of parameters and GLOPs in Table 2. SoftPool does not include trainable variables and thus does not affect the number of parameters, in contrast to recent pooling methods [12, 14, 20, 30]. We also depart from these methods as the number of GFLOPs remains the same as the maximum and average pooling that we replace.

Pooling method comparisons. In Table 5, we compare multiple pooling methods across six networks trained from scratch. SoftPool performs similarly to learnable approaches without requiring additional convolutions, while outperforming stochastic methods. These classification accuracies correlate with image similarities in Table 1. This enforces the notion that pooling methods that retain information will also improve classification accuracy.

4.5. Multi-layer ablation study

In order to better understand how SoftPool affects the network performance at different depths, we use an InceptionV3 model [37] which integrates pooling in its layer structure. We systematically replace the max-pool operations within Inception blocks at different network layers.

From the top-1 and top-5 results summarized in Table 6, we observe that the accuracy increases with the number of pooling layers that are replaced with SoftPool. An average increase of 0.23% in top-1 accuracy is obtained with single layer replacements. The final top-1 accuracy surge between the original network with max-pool (N) and the SoftPool model (VII) is +1.59%. This shows that SoftPool can be used as direct replacement regardless of the network depth.

4.6. Object detection performance

We additionally present results for object detection in order to evaluate the capabilities of SoftPool to preserver local features. In Table 8, we include comparisons for object detection on MS COCO with RetinaNet [24] and Mask R-CNN [15] for several backbones. The results show an average +1.0% AP improvement, +1.1% on AP_{50} and +1.3% on AP_{75}. This further demonstrates the merits of SoftPool in preserving important local information.

Model	GFLOPs	HACS		Kinetics-700		UCF-101	
		top-1(%)	top-5(%)	top-1(%)	top-5(%)	top-1(%)	top-5(%)
r3d-50 [19]**	53.16	78.36	93.76	49.08	72.54	93.13	96.29
r3d-101 [19]**	78.52	80.49	95.18	52.58	74.63	95.76	98.42
r(2+1)d-50 [39]**	50.04	81.34	94.51	49.93	73.40	93.92	97.84
I3D [4]‡*	55.27	79.95	94.48	53.01	69.19	92.45	97.62
ir-CSN-101 [38]‡†	17.26	N/A	N/A	54.66	73.78	95.13	97.85
MF-Net [5]†*	22.50	78.31	94.62	54.25	73.38	93.86	98.37
SlowFast r3d-50 [11]‡†	36.71	N/A	N/A	56.17	75.57	94.62	98.75
SRTG r3d-50 [35]††	53.22	80.36	95.55	53.52	74.17	96.85	98.26
SRTG r(2+1)d-50 [35]††	50.10	83.77	96.56	54.17	74.62	95.99	98.20
SRTG r3d-101 [35]††	78.66	81.66	96.37	56.46	76.82	97.32	99.56
r3d-50 with SoftPool (Ours)	53.16	79.82	94.64	50.36	73.72	93.90	97.02
SRTG r(2+1)d-50 with SoftPool (Ours)	50.10	**84.78**	**97.72**	55.27	75.44	96.46	98.73
SRTG r3d-101 with SoftPool (Ours)	78.66	83.28	97.04	**57.76**	**77.84**	**98.06**	**99.82**

** re-implemented models trained from scratch. †† models and weights from official repositories. ‡∗ unofficial models trained from scratch.
‡† models from unofficial repositories with official weights. †∗ official models trained from scratch.

Table 7. **Action recognition top-1 and top-5 accuracy for HACS, Kinetics-700 and UCF-101**. Models are trained on HACS and fine-tuned for Kinetics-700 and UCF-101, except for ir-CSN-101 and SlowFast r3d-50 (see text). N/A means no trained model was provided.

Model	Backbone	Original						SoftPool					
		AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
RetinaNet	ResNet18	28.3	48.7	31.6	12.6	33.6	40.9	**29.7**	**50.2**	**33.3**	**14.1**	**35.2**	**42.6**
	ResNet34	31.6	50.8	33.9	15.1	36.0	43.6	**32.8**	**52.1**	**35.5**	**16.2**	**37.3**	**45.0**
	ResNet50	34.0	52.5	36.5	17.0	37.4	45.1	**34.9**	**53.4**	**37.6**	**18.0**	**38.5**	**46.4**
	ResNet101	39.1	59.1	42.3	21.8	42.7	50.2	**39.8**	**59.9**	**43.3**	**22.4**	**43.5**	**51.1**
Mask R-CNN	ResNet34	32.9	53.6	32.7	14.5	35.1	43.2	**34.0**	**54.8**	**34.1**	**15.7**	**36.6**	**44.6**
	ResNet50	33.6	55.2	35.3	15.4	36.8	45.3	**34.5**	**56.2**	**36.4**	**16.2**	**37.7**	**46.3**
	ResNet101	38.2	60.3	41.7	20.1	41.1	50.2	**39.0**	**61.1**	**42.6**	**20.9**	**42.0**	**51.3**

Table 8. **Object detection** *single-mode* results (bounding box AP) on COCO `test-dev` for models with original backbone networks and the same backbone networks with pooling layers replaced by SoftPool. All models are pre-trained on ImageNet1K.

4.7. Classification performance on video data

Finally, we demonstrate the merits of SoftPool in handling spatio-temporal data. Specifically, we address action recognition in videos where the input to the network is a stack of subsequent video frames. Representing time-based features stands as a major challenge in action recognition research [34]. The main challenge in space-time data downsampling is the inclusion of key temporal information without impacting the spatial quality of the input.

In this experiment, we use popular time-inclusive networks and replace the original pooling methods with SoftPool. Most space-time networks extend 2D convolutions to 3D to account for the temporal dimension. They use stacks of frames as inputs. For the tested networks with SoftPool, the only modification is that we use SoftPool to deal with the additional input dimension (see Section 3.4).

We trained most architectures from scratch on HACS [49] using author provided implementations. Results for Kinetics-700 and UCF-101 are fine-tuned from the HACS-trained models. We make exceptions for ir-CSN-101 and SlowFast, for which we used the networks trained by the authors. ir-CSN-101 [38] is pre-trained on IG65M [13] while SlowFast [11] is pre-trained on ImageNet.

Results appear in Table 7. We report the performance of three models with the original and SoftPool-replaced implementations. For 3D conv ResNet-50 (r3d-50), using SoftPool increases the top-1 classification accuracy by 1.46%. The accuracy performance also increases by 1.00% and 1.63% for the two SRTG models [35]. For the SRTG ResNet-(2+1) network, we observe state-of-the-art performance on HACS. Also when using space-time data, SoftPool does not add computational complexity (GFLOPS).

For the performance on Kinetics-700, we observe important performance gains. An average of 1.22% increase in top-1 accuracy is shown for the three models that have their pooling operations substituted by SoftPool. The best performing model is SRTG r3d-101 with SoftPool, which achieves a top-1 accuracy of 57.76% and a top-5 accuracy score 77.84%. These models also outperform state-of-the-art models such as SlowFast r3d-50 and ir-CSN-101.

When fine-tuning on UCF-101, the average accuracy gain is 0.66% despite an almost saturated performance. SRTG r3d-101 with SoftPool is the best performing model with a top-1 accuracy of 98.06% and top-5 of 99.82%.

5. Conclusions

We have introduced SoftPool, a novel pooling method to better preserve informative features and, consequently, improve CNN classification and detection performance. SoftPool uses the softmax within a kernel region where each of the activations has a proportional effect on the output. Activation gradients are relative to the weights assigned to them. SoftPool is differentiable, which benefits efficient training. Moreover, it does not require additional parameters nor increases the number of performed operations. We have shown the merits of SoftPool through experimentation on image similarity, image classification, object detection and action recognition in videos. The increase in classification performance combined with the low computation requirements make SoftPool an excellent replacement for current pooling operations, including max and average pooling.

References

[1] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 126–135, 2017. 4, 5, 6

[2] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *International Conference on Machine Learning (ICML)*, pages 111–118, 2010. 2, 3

[3] Joao Carreira, Eric Noland, Chloe Hillier, and Andrew Zisserman. A short note on the Kinetics-700 human action dataset. *arXiv preprint arXiv:1907.06987*, 2019. 4

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017. 8

[5] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Multi-fiber networks for video recognition. In *European Conference on Computer Vision (ECCV)*, pages 352–367, 2018. 8

[6] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Advances in neural information processing systems (NeurIPS)*, pages 4467–4475, 2017. 1

[7] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *European Conference on Computer Vision Workshops (ECCVW)*, pages 1–22, 2004. 2

[8] Adrien Deliège, Maxime Istasse, Ashwani Kumar, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. Ordinal pooling. In *British Machine Vision Conference*, 2019. 2

[9] Allen L Edwards. Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, 13(3):185–187, 1948. 7

[10] Joan Bruna Estrach, Arthur Szlam, and Yann LeCun. Signal recovery from pooling representations. In *International Conference on Machine Learning (ICML)*, pages 307–315. PMLR, 2014. 2

[11] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *International Conference on Computer Vision (ICCV)*, pages 6202–6211, 2019. 8

[12] Ziteng Gao, Limin Wang, and Gangshan Wu. LIP: Local importance-based pooling. In *International Conference on Computer Vision (ICCV)*, October 2019. 2, 5, 6, 7

[13] Deepti Ghadiyaram, Du Tran, and Dhruv Mahajan. Large-scale weakly-supervised pre-training for video action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 8

[14] Caglar Gulcehre, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 530–546. Springer, 2014. 2, 5, 6, 7

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision (ICCV)*, pages 2961–2969. IEEE, 2017. 7

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 6, 7

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 6, 7

[18] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5197–5206, 2015. 4, 5, 6

[19] Hirokatsu Kataoka, Tenga Wakamiya, Kensho Hara, and Yutaka Satoh. Would mega-scale datasets further enhance spatiotemporal 3D CNNs? *arXiv preprint arXiv:2004.04968*, 2020. 8

[20] Takumi Kobayashi. Global feature guided local pooling. In *International Conference on Computer Vision (ICCV)*, pages 3365–3374, 2019. 7

[21] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2169–2178, 2006. 2

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[23] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472, 2016. 2, 5, 6, 7

[24] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017. 7

[25] Yusuke Matsui, Kota Ito, Yuji Aramaki, Azuma Fujimoto, Toru Ogawa, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch-based manga retrieval using manga109 dataset. *Multimedia Tools and Applications*, 76(20):21811–21838, 2017. 4, 5

[26] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. 7

[27] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Conference on Artificial Intelligence (AAAI)*, pages 4780–4789, 2019. 1

[28] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999. 3

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and

Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4, 6

[30] Faraz Saeedan, Nicolas Weber, Michael Goesele, and Stefan Roth. Detail-preserving pooling in deep networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9108–9116, 2018. 2, 5, 6, 7

[31] Thomas Serre, Lior Wolf, and Tomaso Poggio. Object recognition with features inspired by visual cortex. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 994–1000. IEEE, 2005. 2

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[33] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 4

[34] Alexandros Stergiou and Ronald Poppe. Analyzing human-human interactions: A survey. *Computer Vision and Image Understanding*, 188:102799, 2019. 8

[35] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *Pattern Recognition Letters*, 141:1–7, 2021. 8

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition, (CVPR)*. IEEE, 2015. 2

[37] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. 7

[38] Du Tran, Heng Wang, Lorenzo Torresani, and Matt Feiszli. Video classification with channel-separated convolutional networks. In *International Conference on Computer Vision (ICCV)*, pages 5552–5561, 2019. 8

[39] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6450–6459. IEEE, 2018. 8

[40] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3360–3367, 2010. 2

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. 5

[42] Chao-Yuan Wu, Ross Girshick, Kaiming He, Christoph Feichtenhofer, and Philipp Krähenbühl. A multigrid method for efficiently training video models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 153–162. IEEE, 2020. 5

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. 1, 6, 7

[44] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1794–1801, 2009. 2

[45] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *International Conference on Rough Sets and Knowledge Technology*, pages 364–375, 2014. 2

[46] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference (BMVC)*, pages 87.1–87.12, 2016. 6, 7

[47] Matthew D Zeiler and Robert Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representations (ICLR)*, 2013. 2, 5, 6, 7

[48] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4970–4978, 2017. 2, 5, 6, 7

[49] Hang Zhao, Antonio Torralba, Lorenzo Torresani, and Zhicheng Yan. HACS: Human action clips and segments dataset for recognition and temporal localization. In *International Conference on Computer Vision (ICCV)*, pages 8668–8678. IEEE, 2019. 4, 8

[50] Jiaojiao Zhao and Cees G. Snoek. LiftPool: Bidirectional convnet pooling. In *International Conference on Learning Representations (ICLR)*, 2021. 2