池化层： 作用（1）一是对卷积层所提取的信息做进一步降维，减少计算量；
作用（2）二是加强图像特征的不变性，使之增加图像的偏移，旋转等方面的鲁棒性

本文提出了一种快速、高效的池化方法SoftPool，Softpool可以以指数加权方式累加激活。与一系列其他池化方法相比，SoftPool在下采样激活映射中保留了更多的信息，可以获得更好的分类精度。在ImageNet上，对于一系列流行的CNN架构，用SoftPool替换原来的池化操作可以带来1-2%的一致性精度提升。

Refining activation downsampling with SoftPool

Alexandros Stergiou[1] Ronald Poppe[1] Grigorios Kalliatakis[2]

[1]Utrecht University
Utrecht
The Netherlands
{a.g.stergiou,r.w.poppe}@uu.nl

[2]University of Warwick
Coventry
United Kingdom
grigorios.kalliatakis@warwick.ac.uk

Abstract

Convolutional Neural Networks (CNNs) use pooling to decrease the size of activation maps. This process is crucial to locally achieve spatial invariance and to increase the receptive field of subsequent convolutions. Pooling operations should minimize the loss of information in the activation maps. At the same time, the computation and memory overhead should be limited. To meet these requirements, we propose SoftPool: a fast and efficient method that sums exponentially weighted activations. Compared to a range of other pooling methods, SoftPool retains more information in the downsampled activation maps. More refined downsampling leads to better classification accuracy. On ImageNet1K, for a range of popular CNN architectures, replacing the original pooling operations with SoftPool leads to consistent accuracy improvements in the order of 1-2%. We also test SoftPool on video datasets for action recognition. Again, replacing only the pooling layers consistently increases accuracy while computational load and memory remain limited. These favorable properties make SoftPool an excellent replacement for current pooling operations, including max-pool and average-pool[1].

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, increasing the receptive field of subsequent convolutions [5, 25, 42].

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.

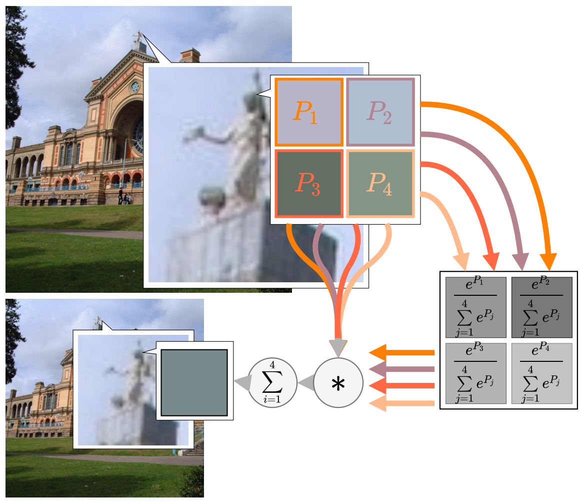
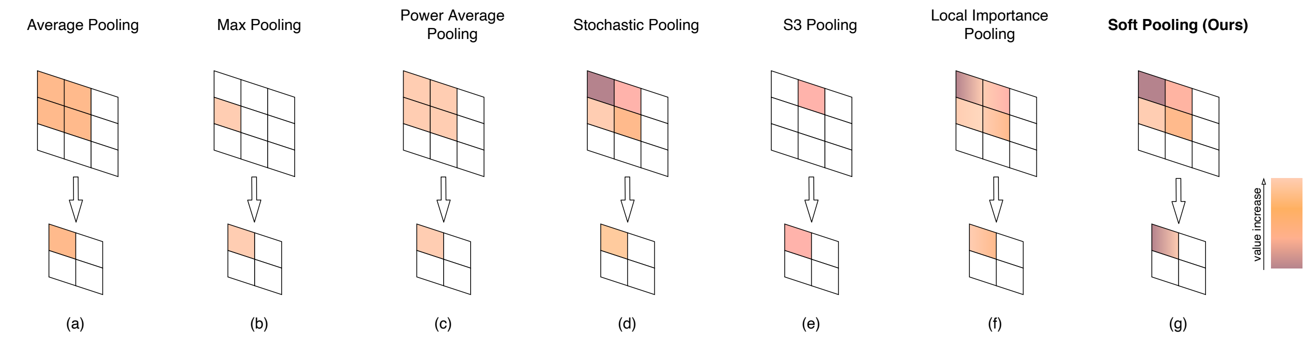

P_1
P_2
P_3
P_4
$\frac{e^{P_1}}{\sum_{j=1}^{4} e^{P_j}}$
$\frac{e^{P_2}}{\sum_{j=1}^{4} e^{P_j}}$
$\frac{e^{P_3}}{\sum_{j=1}^{4} e^{P_j}}$
$\frac{e^{P_4}}{\sum_{j=1}^{4} e^{P_j}}$
$*$
$\sum_{i=1}^{4}$

Figure 1. **SoftPool illustration**. The original image is subsampled with a 2×2 ($k = 2$) kernel. The output is the sum of the exponentially weighted original pixel values in the kernel region.

We introduce *SoftPool*, a kernel-based pooling method that uses the softmax weighted sum of activations. Through extensive experiments, we demonstrate that SoftPool largely preserves descriptive activation features, while remaining computation- and memory-efficient. SoftPool equiped CNNs consistently outperform their vanilla counterparts. 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 by preserving the original features, measured using image similarity.
- Experimental results on image and video classification tasks show consistent improvement when replacing the

[1]Our an code is available at: https://git.io/JL5zL



$$\tilde{\mathbf{a}} = \sum_{i \in \mathbf{R}} \frac{\mathbf{a}_i}{|R|}$$

$$\tilde{\mathbf{a}} = \max_{i \in \mathbf{R}} \mathbf{a}_i$$

$$\tilde{\mathbf{a}} = \sqrt[p]{\sum_{i \in \mathbf{R}} \mathbf{a}_i^p}$$

$$\tilde{\mathbf{a}} = \mathbf{a}_l,\ l \sim \mathbf{a}_{(p_1,\ldots,p_{|\mathbf{R}|})}, \quad p_i = \frac{\mathbf{a}_i}{\sum_{j \in \mathbf{R}} \mathbf{a}_j}$$

$$\tilde{\mathbf{a}} = \mathbf{a}_i, \quad i \sim \{1\ldots,|\mathbf{R}|\}$$

$$\tilde{\mathbf{a}} = \frac{\sum_{i \in \mathbf{R}} \mathbf{a}_i * e^{G(\mathbf{a}_i)}}{\sum_{j \in \mathbf{R}} e^{G(\mathbf{a}_j)}}$$

$$\tilde{\mathbf{a}} = \sum_{i \in \mathbf{R}} \frac{e^{\mathbf{a}_i} * \mathbf{a}_i}{\sum_{j \in \mathbf{R}} e^{\mathbf{a}_j}}$$

Figure 2. **Pooling variants**. We simplify the equations by using R for representing the width and height dimensions. (a,b) **average and maximum pooling** are based on averaging and maximum activation selection in a kernel. (c) **Power Average pooling** (L_p) [8, 13] is proportional to average pooling raised to a power (p), with the output being equal to max-pool ($p \to \infty$) or sum pooling ($p = 1$). (d) **Stochastic pooling** [46] outputs a randomly selected activation from the kernel region. (e) **Stochastic Spatial Sampling** (S3Pool) [47] samples random horizontal and vertical regions given a specified stride. (f) **Local Importance Pooling** (LIP) [10] uses a trainable function G to enhance specific features. (g) **Softpool** (ours) exponentially weighs the effect of activations using a softmax kernel.

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 (Sec. 3) and evaluate it in terms of feature loss and image and video classification performance (Sec. 4).

2. Related Work

Pooling for hand-crafted features. Downsampling has been a widely used technique in hand-coded feature extraction methods such as Bag-of-Words and Bag-of-Features [7] where images are viewed as collections of local patches, pooled and encoded as vectors [39]. This approach has also been studied in combination with Spatial Pyramid Matching [20] to preserve spatial information. Later works considered the selection of the maximum SIFT features in a spatial region [43]. Pooling has primarily been linked to the use of max-pooling, because of the feature robustness of biological max-like cortex signals [30]. Boureau *et al.* [1] studied maximum and average pooling in terms of their robustness and usability. Max-pooling has been effectively used 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 [21]. The main benefit of pooling has traditionally been the creation of condensed feature representations that are significantly smaller than the originals, thus reducing computational requirements and empowering the creation of larger and deeper architectures [31, 35].

Recent efforts have focused on preserving relevant features in the downsampling process. An overview of a number of popular pooling methods appears in Figure 2. Initial approaches include stochastic pooling [46], 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 [44] or through a combination of portions from each method [22]. Based on the combination of averaging and maximization, Power average (L_p) pooling [8, 13] 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, as in S3Pool [47], in which the downsampled outputs stem from randomly sampling the rows and columns from the original feature map grid. Detail Preserving Pooling (DPP, [29]) uses average pooling while enhancing activations with above-average values. Local Importance Pooling (LIP, [10]) has further evaluated the use of learned weights as a sub-network attention-based mechanism that can be utilized for pooling informative features while discarding uninformative ones.

In contrast to the previously described operations, some pooling methods have strict architectural or task-based criteria that limit their application to specific CNN architectures and tasks. For example, spatial pyramid pooling (SPS, [15]) is based on object windows and requires multi-layer bins and global average pooling [23]. ROI-pool [12] uses bounding boxes to distinguish foreground from background. ROI-align [14] further includes quantization.

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 soft-

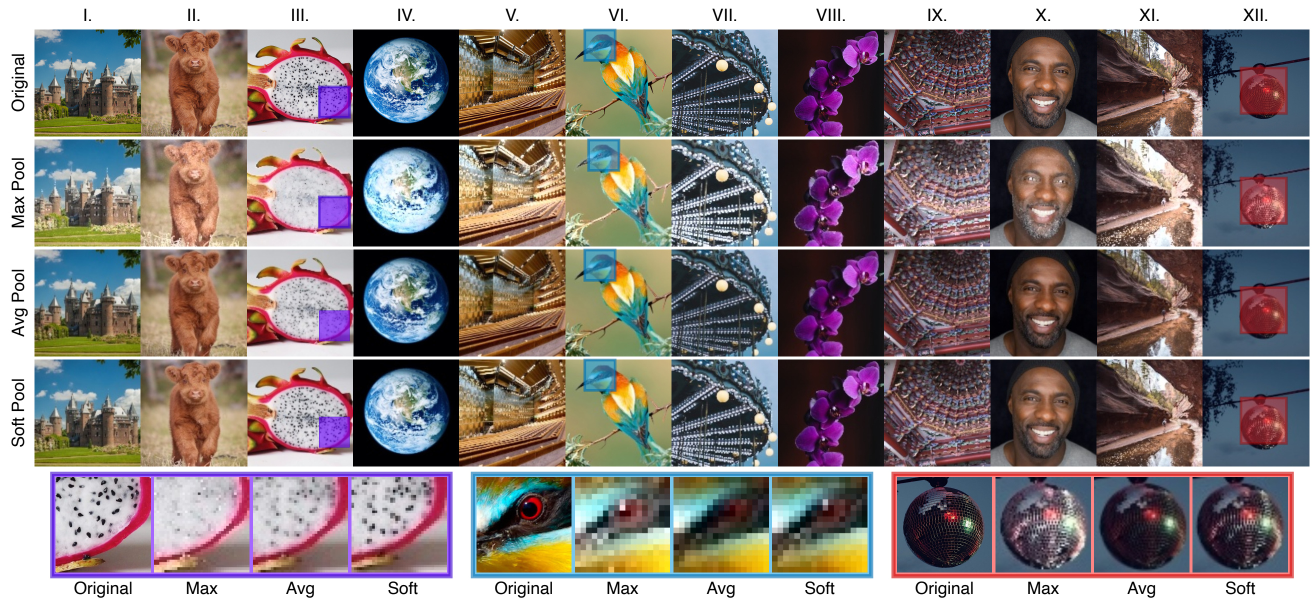


Figure 3. **Examples of maximum, average and soft pooling**. Original images are of size 1200 × 1200 pixels with the pooled ones downsampled to 12.5% of the original size with 3× pooling operations. Images have been selected based on their overall contrast (I, III, V, VIII, XII), color (II, IV, VI, IX), detail (III, V, X, XI, XII) and texture (II, VII). Images on the bottom row include zoomed-in regions. In most cases, max-pooling significantly distorts image features by increasing high-valued pixels (seen at the dragon fruit and disco ball examples II & XII). Average pooling can decrease the neighborhood of pixels (as with the bird example VI). SoftPool preserves both the global structure, as well as informative details.

max weighting approach to preserve the basic properties of the input while amplifying feature activations of greater intensity. Different from max-pooling, softmax pooling is differentiable. Consequently, we obtain a gradient for each input during backpropagation, which is likely to increase the training efficacy. We demonstrate the effects of SoftPool in Figure 3, where the zoomed-in regions show that features are not completely lost as with hard-max selection, or suppressed by others within the region, through averaging.

Other previous methods use trainable parameters resulting in increased computational costs, directly impacting usability in larger networks. In contrast, SoftPool can replace any pooling operation, including maximum and average pooling, and is both computation- and memory-efficient.

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 [26] as well as the early pooling experiments with hand-coded features of Boureau *et al.* [1]. 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 at least a minimum gradient value during backpropagation. This is in contrast to pooling methods that employ hard-max 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 all natural exponents of the activations within neighborhood 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

[illegible]

activations with greater effect is a more balanced approach than simply selecting the 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 significantly a reduction in the overall regional feature intensity.

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

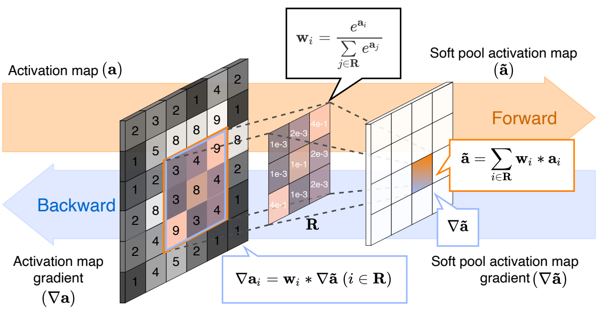

$\mathbf{w}_i = \frac{e^{\mathbf{a}_i}}{\sum_{j \in \mathbf{R}} e^{\mathbf{a}_j}}$
Activation map $(\mathbf{a})$
Soft pool activation map $(\tilde{\mathbf{a}})$
Forward
Backward
$\tilde{\mathbf{a}} = \sum_{i \in \mathbf{R}} \mathbf{w}_i * \mathbf{a}_i$
$\mathbf{R}$
$\nabla\tilde{\mathbf{a}}$
Activation map gradient $(\nabla\mathbf{a})$
$\nabla\mathbf{a}_i = \mathbf{w}_i * \nabla\tilde{\mathbf{a}}\ (i \in \mathbf{R})$
Soft pool activation map gradient $(\nabla\tilde{\mathbf{a}})$

Figure 4. **SoftPool operation overview**. In forward operation, in orange, the kernel uses the exponential softmax value of each activation as weight and then calculates the weighted sum for the region $\mathbf{R}$. The same weights are also used for the gradients $(\nabla\tilde{\mathbf{a}}_i)$, denoted in blue. The 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. This corresponds to updates in the gradients of smaller activations being less than those of substantial activations.

Softmax is differentiable unlike max or stochastic pooling. As a result, during backpropagation, a minimum nonzero weight will be assigned to every activation within a kernel region. This enables the calculation of a gradient for every single activation in that region, as 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). We retain the differentiable nature of softmax by assigning a lower arithmetic limit given the number of bits used by each type. This prevents arithmetic underflow. We implement this mechanism for both the kernel values, and the produced activation value used in the final summation.

3.3. Feature preservation

An integral goal of sub-sampling is the preservation of the 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 Figure 3. Average pooling decreases the effect of all activations in the region equally, while max pooling selects only the single highest activation in the region. The effect of SoftPool falls between these two pooling methods, as all activations in the region contribute to the final output while higher activations are more dominant compared to lower ones. This can balance out the effects of both average and max pooling, while leveraging the beneficial properties of both. An example of the level of detail that can be preserved with SoftPool is shown in Figure 5. The colors (channel values) remain true to those of the original images even in cases where the image has been sub-sampled substantially (e.g., 25% of the original).

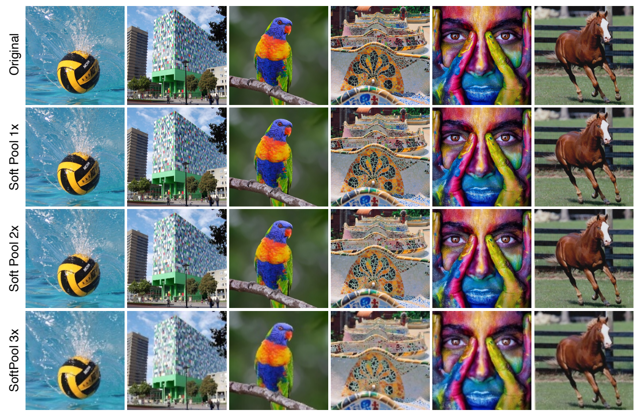


Figure 5. **Results after repeated SoftPool downsampling**. Top row shows the original 1000 $\times$ 1000 images. Rows 2–4 show the downsampled images with SoftPool applied 1$\times$, 2$\times$ and 3$\times$, respectively. The corresponding image sizes are 500 $\times$ 500, 250 $\times$ 250 and 125 $\times$ 125. Best viewed zoomed-in.

3.4. Spatio-temporal kernels

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

The produced output now holds condensed spatio-temporal information. 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. In the next section, we demonstrate that SoftPool performs favorably with 3D inputs.

4. Experimental Results

We first evaluate the information loss for various pooling operators. We compare the downsampled outputs to the original inputs using three 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 (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 4.6). 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 the ImageNet1K dataset [28]. It contains approximately 1.2M images from 1,000 classes. For action recognition, we use the HACS [48] and Kinetics-700 [2] benchmark datasets which include 500K and 600K videos from 200 and 700 classes, respectively. For our transfer learning experiment, we use the UCF-101 [32] dataset. It contains 101 classes and 9K videos.

Implementation details. Training for the image classification task is performed with a random region selection of 294 $\times$ 294 height and width, which was then resized to 224 $\times$ 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 128 to ensure all models are trained with the same batch size. Possible deviations (in the range of 1-2%) from reported models in the literature are due to our training environment settings.

For our experiments on videos, we use a multigrid training scheme [41], 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 five different kernel sizes (i.e., $k \times k$, with $k = \{2...6\})$ and report the following three different similarity measures.

Structural Similarity Index (SSI) [40] is a widely-used similarity measure between two images, in our case the original and downsampled image. The SSI is based on the computation of a luminance term, a contrast term and a structural term. Larger index values correspond to larger structural similarities between the images compared.

Pixel-wise similarity (Pix Sim) measures the normalized inverse of the pixel-wise L_1 distance between the original and the downsampled image up-scaled through spline interpolation to match the image sizes. The results are summed and then averaged over the image dimensions.

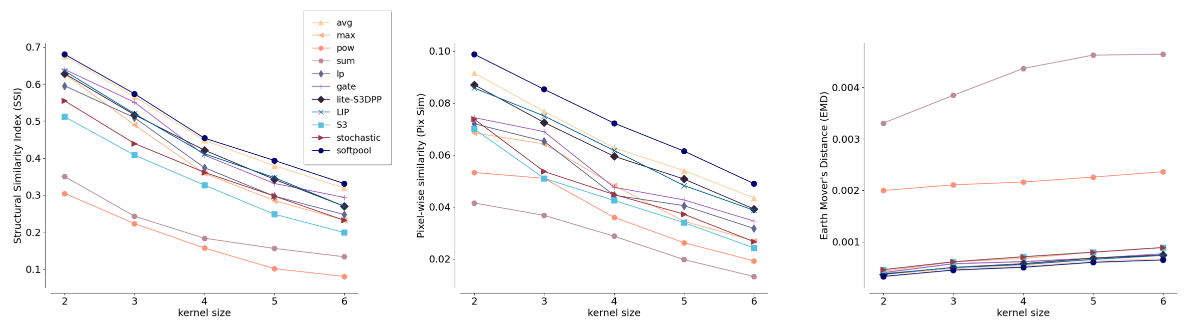
Earth mover's Distance (EMD) uses the Wasserstein metric [27] to compare the color-based histogram distributions of the original and downsampled images. Both histograms use a single channel version of their pixel values produced by the ITU-R 601-2 luma transform [6].

We randomly select 10 images for each class in ImageNet1K and evaluate performance of a number of pooling methods in terms of these three similarity measures. We apply pooling directly on the images, instead of on activation maps. Results appear in Table 1 and Figure 6.

Overall, SoftPool has a SSI above all other pooling methods for all tested kernel sizes. While differences with average pooling are small for SSI, we see larger differences in pixel-wise similarity. Finally, the Earth Mover's Distance is lower for the SoftPool downsampled image in comparison to those of other pooling methods, albeit these differences are again small. In sum, SoftPool consistently achieves the best similarity scores, independent of the kernel size. In Section 4.4, we investigate whether this favorable property also leads to better generalization and, consequently, higher classification rates.

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, a slow or memory-intensive pooling operations can have a



(a) SSI (b) Pix Sim (c) EMD

Figure 6. **Comparisons with different kernel sizes, based on Structural Similarity Index (SSI), Pixel Similarity (Pix Sim) and Earth Mover's Distance (EMD)** for different pooling methods. Results are averaged over 10 images for each class in ImageNet1K.

	Pooling method	CPU (ms) (↓ F / ↑ B)	CUDA (ms) (↓ F / ↑ B)	Memory (MB)	$k=2$			$k=3$			$k=4$			$k=5$			$k=6$		
					SSI	PS	EMD	SSI	PS	EMD	SSI	PS	EMD	SSI	PS	EMD	SSI	PS	EMD
Simple	Average	9.5 / 49.1	14.5 / 76.3	225.8	0.675	0.051	3.6e-4	0.567	0.077	4.7e-4	0.448	0.063	5.1e-4	0.379	0.054	6.2e-4	0.319	0.043	6.7e-4
	Maximum	91.5 / 152.3	195.3 / 267.8	370.8	0.627	0.044	4.2e-4	0.489	0.064	6.1e-4	0.359	0.048	6.7e-4	0.285	0.035	8.0e-4	0.235	0.027	8.8e-4
	Pow-average	74.9 / 329.6	120.3 / 433.7	790.6	0.305	0.028	1.9e-3	0.223	0.036	2.1e-3	0.157	0.028	2.2e-3	0.101	0.019	2.2e-3	0.080	0.013	2.3e-3
	Sum	26.7 / 163.5	79.3 / 323.7	810.8	0.350	0.011	3.3e-3	0.243	0.037	3.8e-3	0.183	0.029	4.3e-3	0.156	0.020	4.6e-3	0.135	0.013	4.6e-3
Trainable	L_p [13]	116.8 / 338.0	214.3 / 422.9	825.9	0.595	0.044	4.0e-4	0.509	0.069	5.7e-4	0.374	0.047	6.1e-4	0.297	0.042	6.8e-4	0.247	0.038	7.6e-4
	Gate [22]	245.1 / 339.7	327.5 / 540.1	1180.7	0.639	0.047	3.9e-4	0.551	0.068	5.7e-4	0.408	0.047	6.1e-4	0.331	0.043	6.7e-4	0.293	0.034	7.6e-4
	Lite-S3DPP [29]	427.1 / 860.3	634.0 / 1228.4	1441.4	0.628	0.049	3.6e-4	0.517	0.072	5.0e-4	0.421	0.059	5.7e-4	0.343	0.051	6.8e-4	0.270	0.039	7.4e-4
	LIP [10]	134.0 / 257.5	258.6 / 362.1	699.04	0.635	0.048	3.7e-4	0.520	0.075	4.9e-4	0.412	0.061	5.5e-4	0.347	0.048	6.6e-4	0.270	0.038	7.3e-4
Stoch.	Stochastic	162.3 / 341.8	219.1 / 485.7	820.4	0.555	0.039	4.5e-4	0.439	0.054	6.1e-4	0.361	0.046	7.1e-4	0.297	0.038	7.9e-4	0.239	0.027	8.6e-4
	S3 [47]	233.1 / 410.5	345.4 / 486.2	945.6	0.511	0.035	4.5e-4	0.408	0.051	6.2e-4	0.327	0.042	7.1e-4	0.248	0.034	7.9e-4	0.199	0.024	8.8e-4
	SoftPool (Ours)	31.3 / 156.9	56.7 / 234.6	263.6	**0.680**	**0.060**	**3.2e-4**	**0.573**	**0.085**	**4.5e-4**	**0.454**	**0.072**	**5.0e-4**	**0.393**	**0.061**	**6.0e-4**	**0.331**	**0.049**	**6.4e-4**

Table 1. **Similarity measure results on five different kernel sizes (k) and computational latency (in msec.) for forward and backward passes in both CPU and CUDA-enabled operations**. All similarity experiments for SSI, Pix Sim and EMD are averaged over 10 images per class of ImageNet1K, with the target output size being half of that of the original. Latency rates are summed over 100 consecutive pooling operations with randomly generated inputs.

detrminetal effect on the performance.

To test the computation and memory overhead, we report the memory use and inference times for runs on both CPU and GPU (CUDA) in Table 1. Our testing environment consists of an AMD Threadripper 2950X with 2400MHz RAM frequency and four Nvidia 2080 Tis. Results are based on 100 consecutive pooling operations. We use PyTorch, and test the publicly available implementations. *Simple* pooling methods (see Table 1) are integrated parts of the PyTorch library and thus have been better optimized.

It is evident from Table 1 that our implementation of SoftPool can achieve low inference times for both CPU- and CUDA-based operations, while remaining memory-efficient as the proposed method is simple and very efficient to compute within a kernel region. SoftPool is second only to average pooling in terms of latency and memory use.

4.4. Classification performance on ImageNet

We investigate whether SoftPool's favorable performance in terms of retaining information also leads to better classification accuracy. We replace the original pooling layers (denoted as vanilla) in ResNet50, ResNet101, DenseNet121 and ResNeXt50 32x4d with several pooling methods reported in the literature. These models are chosen based on their widespread usability. Directly replacing the original pooling layers allows us to compare the classification accuracy on ImageNet1K.

Network	Pooling substitution methods						
	Vanilla	L_p	Gate	Lite-S3DPP	LIP	S3	SoftPool
ResNet50 [16]	72.964	73.026	73.724	74.135	74.242	72.821	**74.799**
ResNet101 [16]	74.662	74.819	75.120	75.376	75.104	74.681	**75.825**
DenseNet121 [17]	71.916	72.342	72.674	73.264	73.245	71.896	**73.828**
ResNeXt50 [42]	74.685	74.834	75.503	75.898	75.637	74.694	**76.489**

Table 2. **Pooling layer substitution** top-1 accuracy for a variety of pooling methods. Experiments were performed on Imagenet1K.

As shown in Table 2, replacements with SoftPool yield consistent accuracy improvements in the order of 1.16-1.98% over the vanilla networks. In addition, results with SoftPool outperform all other tested pooling operations.

In Table 3, we report on a larger variety of CNN architectures with pairwise comparisons between the performance on ImageNet1K using the original pooling layers, and with all pooling operations replaced by SoftPool. For (Wide-) ResNets and ResNeXt models, we replace the max-pool operation after the first convolution layer. For DenseNet, we additionally replace four average pooling layers.

Replacing pooling layers with SoftPool improves the overall performance of the network, for all tested networks.

Model	Params	GFLOP	Original		SoftPool	
			top-1	top-5	top-1	top-5
ResNet18	11.69M	1.83	67.246	87.302	**68.972**	**88.762**
ResNet34	21.80M	3.68	70.689	89.764	**72.717**	**90.937**
ResNet50	25.56M	4.14	72.964	90.989	**74.799**	**92.236**
ResNet101	44.55M	7.87	74.662	91.348	**75.825**	**92.835**
ResNet152	60.20M	11.61	75.494	92.536	**76.587**	**93.226**
DenseNet121	7.98M	2.90	71.916	90.239	**73.828**	**91.742**
DenseNet161	28.68M	7.85	75.386	92.165	**76.953**	**93.524**
DenseNet169	14.15M	3.44	74.197	91.254	**75.252**	**92.402**
DenseNet201	20.01M	4.39	75.763	92.546	**77.271**	**93.869**
ResNeXt50 32x4d	25.03M	4.29	74.685	92.074	**76.489**	**93.136**
ResNeXt101 32x4d	44.18M	8.06	75.864	92.982	**77.716**	**93.750**
ResNeXt101 32x8d	88.79M	16.55	77.286	93.758	**78.673**	**94.534**
ResNeXt101 64x4d	83.46M	15.59	76.923	93.261	**78.492**	**94.245**
Wide-ResNet50	68.88M	11.46	74.964	91.863	**76.664**	**93.312**
Wide-ResNet101	126.89M	22.84	75.173	92.385	**76.996**	**93.514**

Table 3. **Pairwise comparisons of top-1 and top-5 accuracies** on ImageNet1K between the vanilla networks and the same networks with the original pooling operations replaced by SoftPool.

The classification accuracy gain is consistent, between 1.06 and 2.03% (average 1.60%).

Table 3 also summarizes the number of parameters and GLOPs. SoftPool does not increase the number of parameters because it does not use trainable variables, in contrast to recent pooling methods [10, 13, 19, 29]. Also, the number of operations (GFLOPs) is the same as the maximum and average pooling operators that we replaced.

For ResNet [16] architectures, when switching to SoftPool, an average top-1 accuracy improvement of 1.57% is observed with a maximum improvement of 2.03% on ResNet50. DenseNet [17] top-1 accuracy gains are in the 1.06–1.91% range. The top-5 accuracy follows similar incremental trends with a 1.33% average increase. For both top-1 and top-5, the largest increase is for DenseNet121.

For ResNeXt [42] models, an average of 1.65% top-1 accuracy improvement is reported. The best model, ResNeXt 32x8d, achieves 78.492% top-1 accuracy (+1.39%) and 94.534% top-5 accuracy (+0.78%) with SoftPool.

Finally, Wide-Resnets [45] have accuracy increases of 1.70% and 1.45% for top-1 and top-5, respectively, on Wide-ResNet50. On Wide-ResNet101, these gains are 1.82% and 1.13% for top-1 and top-5 respectively.

Together, these results demonstrate that simply replacing a single pooling layer (ResNet, ResNeXt and Wide-ResNet) or only five pooling layers (DenseNet) with SoftPool operations, leads to a modest but important increase in accuracy without affecting the number of parameters and GFLOPs. The use of SoftPool typically performs as well as using a deeper network with maximum or average pooling.

4.5. Multi-layer ablation study

In order to better understand how SoftPool effects the network performance at different depths, we use an InceptionV3 model [36] which integrates pooling within its layer structure. We systematically replace the max-pool operations within Inception blocks at different network layers.

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 (%)	74.549	74.861	75.024	75.461	75.706	75.983	76.307	**76.562**
Top-5 (%)	91.844	92.130	92.289	92.524	92.697	92.823	92.996	**93.148**

Table 4. **InceptionV3 with substituted layers at various depths**. Experiments are performed on ImageNet1K and are numbered from N-VII based on the number of replaced pooling layers. Replaced layers are marked with ✓.

Judging from the top-1 and top-5 results summarized in Table 4, the accuracy increases with the number of pooling layers that are replaced with SoftPool. An average increase of 0.29% in top-1 and 0.20% in top-5 accuracies are observed over single layer replacements. The final accuracy surge between the original network with max-pool (N) and the SoftPool model (VII) is +2.01% for top-1 and +1.30% for top-5. This demonstrates that the proposed SoftPool operation can be used as a direct replacement regardless of the network depth.

4.6. 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 frames in a video. Representing time-based features stands as a major challenge in action recognition research [33]. This challenge is also present when downsampling space-time data because the downsampled volume needs to include key temporal information without decreasing the quality of the spatial structure of the input activations.

In this experiment, we use pre-existing 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 [48] using the implementations provided by the authors. 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 for Kinetics-700 that were provided by the authors. ir-CSN-101 [37] is pre-trained on IG65M [11], a large dataset consisting of 65M video that is not publicly available. SlowFast [9] is pre-trained on the full ImageNet dataset.

Results appear in Table 4.5. For three architectures,

Model	GFLOPs	HACS		Kinetics-700		UCF-101	
		top-1(%)	top-5(%)	top-1(%)	top-5(%)	top-1(%)	top-5(%)
r3d-50 [18]**	53.16	78.361	93.763	49.083	72.541	93.126	96.293
r3d-101 [18]**	78.52	80.492	95.179	52.583	74.631	95.756	98.423
r(2+1)d-50 [38]**	50.04	81.340	94.514	49.927	73.396	93.923	97.843
I3D [3]‡*	55.27	79.948	94.482	53.015	69.193	92.453	97.619
ir-CSN-101 [37]‡†	17.26	N/A	N/A	54.665	73.784	95.126	97.851
MF-Net [4]†*	22.50	78.312	94.625	54.249	73.378	93.863	98.372
SlowFast r3d-50 [9]‡†	36.71	N/A	N/A	56.167	75.569	94.619	98.756
SRTG r3d-50 [34]††	53.22	80.362	95.548	53.522	74.171	96.853	98.263
SRTG r(2+1)d-50 [34]††	50.10	83.774	96.560	54.174	74.620	95.993	98.197
SRTG r3d-101 [34]††	78.66	81.659	96.326	56.462	76.819	97.325	99.557
r3d-50 with SoftPool (Ours)	53.16	79.821	94.638	50.362	73.716	93.896	97.021
SRTG r(2+1)d-50 with SoftPool (Ours)	50.10	**84.780**	**97.724**	55.273	75.441	96.465	98.731
SRTG r3d-101 with SoftPool (Ours)	78.66	83.284	97.042	**57.756**	**77.839**	**98.064**	**99.823**

** 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 re-trained models.

Table 5. **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.

we report the performance on both the vanilla models and their counterparts with all pooling operations replaced by SoftPool. For r3d-50, a ResNet with 3D convolutions, using SoftPool increases the top-1 classification accuracy by 1.46%. The accuracy performance also increases with 1.00% and 1.63% for the two SRTG models [34]. For the SRTG model with ResNet-3D backbone with (2+1), we achieve state-of-the-art performance on HACS. Also when using spatio-temporal data, SoftPool does not add computational complexity.

When looking at the performance for 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 57.756% and a top-5 accuracy score 77.839%. These models also outperform state-of-the-art models such as SlowFast r3d-50 and ir-CSN-101.

4.7. Feature transferability evaluation

As fine-tuning is an important aspect when training CNNs, we test the transfer-learning capabilities of the video-based 3D CNN models on the significantly smaller UCF-101 dataset [32]. As shown in Table 4.5, SoftPool networks outperform their max-pooling counterparts consistently. The accuracy gain is on average 0.66% despite an almost saturated performance. SRTG r3d-101 is the best performing model when utilizing SoftPool with a top-1 accuracy of 98.064% and top-5 of 99.823%. Currently, only LGD-3D two-stream [24] has a higher reported accuracy of 98.2%, but with additional optical flow input.

5. Conclusions

We have introduced SoftPool, a novel pooling method that better preserves informative features and, consequently, improves classification performance in CNNs. SoftPool uses the softmax of inputs 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. Our operation is differentiable, which benefits efficient training. SoftPool does not require additional parameters nor increases the number of performed operations.

Experiment results with Structural Similarity Index (SSI), Pix Sim and Earth Mover's Distance show that, in comparison to other methods, SoftPool better preserves features when downsampling, independent of the kernel size. We compare the inference times for both CPU and GPU settings, showing that SoftPool is fast and memory-efficient.

We demonstrate that the better preservation of information is beneficial for classification accuracy. In several experiments, the systematic substitution of pooling operations by SoftPool consistently leads to improvements. Experiments on ImageNet1K show performance gains in the range 1.16-1.98% for a wide range of popular CNN architectures. Similarly, we demonstrate the use of SoftPool on space-time data for action recognition from video. Again, systematic increases in classification accuracy of 1.06-2.03% are obtained on HACS, Kinetics-700 and UCF101.

The increase in classification performance combined with the low computation and memory requirements make SoftPool an excellent replacement for current pooling operations, including max-pool and average pooling.

References

[1] 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

[2] 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. 5

[3] 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. IEEE, 2017. 8

[4] 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

[5] 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

[6] Alex Clark. Pillow (pil fork) documentation, 2015. 5

[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] 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

[9] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast networks for video recognition. In *International Conference on Computer Vision (ICCV)*, pages 6202–6211. IEEE, 2019. 7, 8

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

[11] 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)*. IEEE, 2019. 7

[12] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 2

[13] 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, 6, 7

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *International Conference on Computer Vision (ICCV)*, pages 2961–2969. IEEE, 2017. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision (ECCV)*, pages 346–361. Springer, 2014. 2

[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. IEEE, 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. IEEE, 2017. 6, 7

[18] 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

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

[20] 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. IEEE, 2006. 2

[21] 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

[22] 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, 6

[23] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *International Conference on Learning Representations (ICLR)*, 2014. 2

[24] Zhaofan Qiu, Ting Yao, Chong-Wah Ngo, Xinmei Tian, and Tao Mei. Learning spatio-temporal representation with local and global diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12056–12065. IEEE, 2019. 8

[25] 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. AAAI, 2019. 1

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

[27] Ludger Rüschendorf. Wasserstein metric. *Hazewinkel, Michiel, Encyclopaedia of Mathematics, Springer*, 2001. 5

[28] 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. 5

[29] 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. IEEE, 2018. 2, 6, 7

[30] 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

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

[32] 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. 5, 8

[33] Alexandros Stergiou and Ronald Poppe. Analyzing human-human interactions: A survey. *Computer Vision and Image Understanding*, 188:102799, 2019. 7

[34] Alexandros Stergiou and Ronald Poppe. Learn to cycle: Time-consistent feature discovery for action recognition. *Pattern Recognition Letters*, 141:1–7, 2020. 8

[35] 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

[36] 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. IEEE, 2016. 7

[37] 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. IEEE, 2019. 7, 8

[38] 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

[39] 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. IEEE, 2010. 2

[40] 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

[41] 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

[42] 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. IEEE, 2017. 1, 6, 7

[43] 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. IEEE, 2009. 2

[44] 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. Springer, 2014. 2

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

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

[47] 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. IEEE, 2017. 2, 6

[48] 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. 5, 7

Refining activation downsampling with SoftPool – Supplementary Material

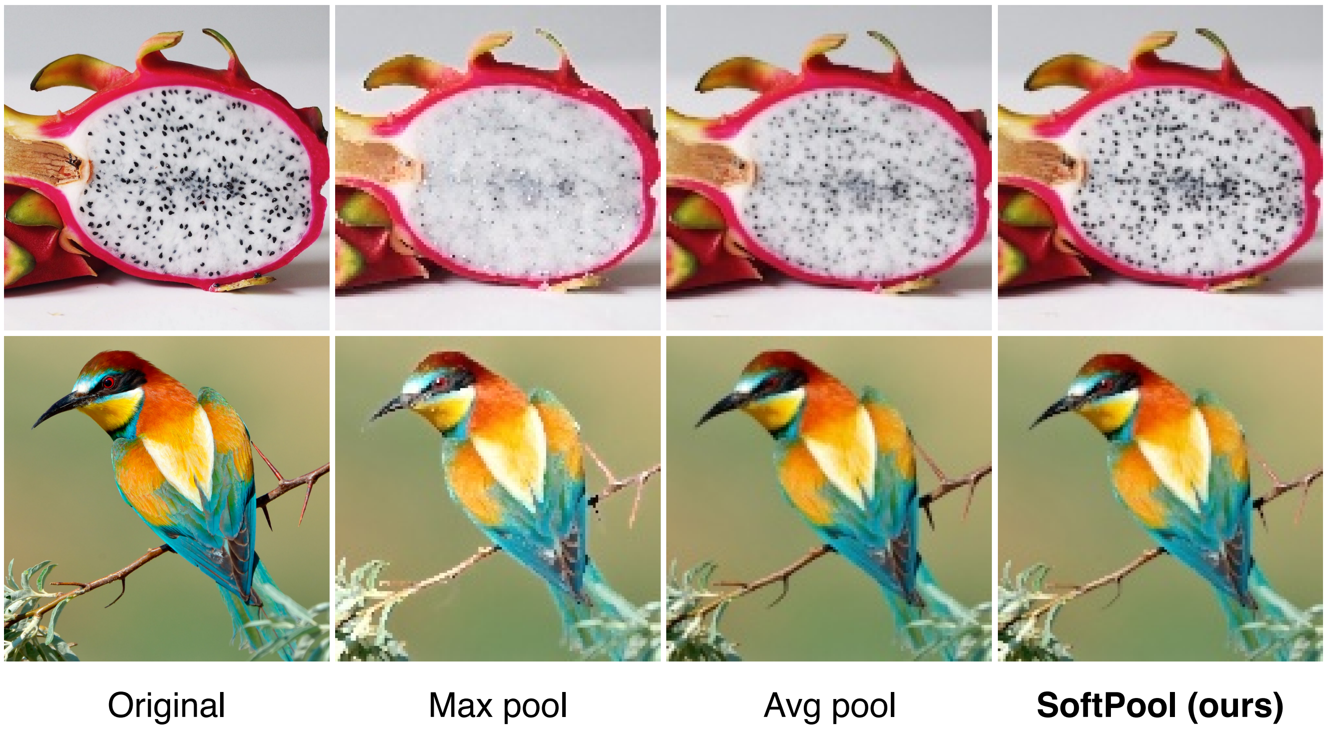


Figure 7. **High resolution pooled images**. Original images used are of size 1200 × 1200. The downsampled images were created with ×3 pooling operations. To match the sizes of the original images and make the downsampling result more visible, we used inter area interpolation to resize the pooled images. This does not create a smoothing effect between neighboring pixels that are interpolated but rather populates new pixels based on the area relation.

6. Detail preservation

We discussed and demonstrated the feature preservation capabilities of SoftPool in Section 3.3. Here, we provide high-resolution images from examples in Figure 3 of the paper, in order to demonstrate clearer the effects of each pooling method. As it can be seen in Figure 7, SoftPool can better capture detail in high-contrasting areas. Evidence of this can be seen in the top image where the dragon fruit seeds are not always preserved in the down-sampled images. For max pooling, most of the seeds are lost. This also applies to a certain extent to average pooling. By selecting the average within a region, features with high contrast are smoothed over, which reduces their effect significantly. In contrast, SoftPool preserves such regions in the sub-sampled outputs. By including part of the low-intensity regions in the output while weighting the high-intensity regions more, it can preserve the little-contrasting pixels. A similar pattern can also be seen with low-contrasting regions such as the bird's eye in the second row where, again, max pooling will highlight the high intensity features in the output while the down-sampled output becomes less similar to that of the original image. Average pooling would instead make low-contrast features much more difficult to be recognized. SoftPool provides a balance between the two methods by weighting each part of the region respectfully to it's intensity value.

7. Model feature visualization

Learned feature interpretability aims at understanding the features that networks associate with each class. One technique is *activation maximization* [S2] which creates a synthetic image by maximizing the activations relating to a specific neuron. Neurons could correspond to a specific class [S6] or features within the feature extractor [S1].

To test the representation capabilities of networks with SoftPool, we use InceptionV3 [36] as a backbone model and visualize the top-10 most informative features. To train, we initialize a random noise image which we use as input for each of the two tested networks. During the training process, the image is optimized with the maximization of the activations of the top-10 kernels in the final InceptionV3 block (*Mixed7c*) as objective function. The top-10 kernels are selected based on the highest average activations across

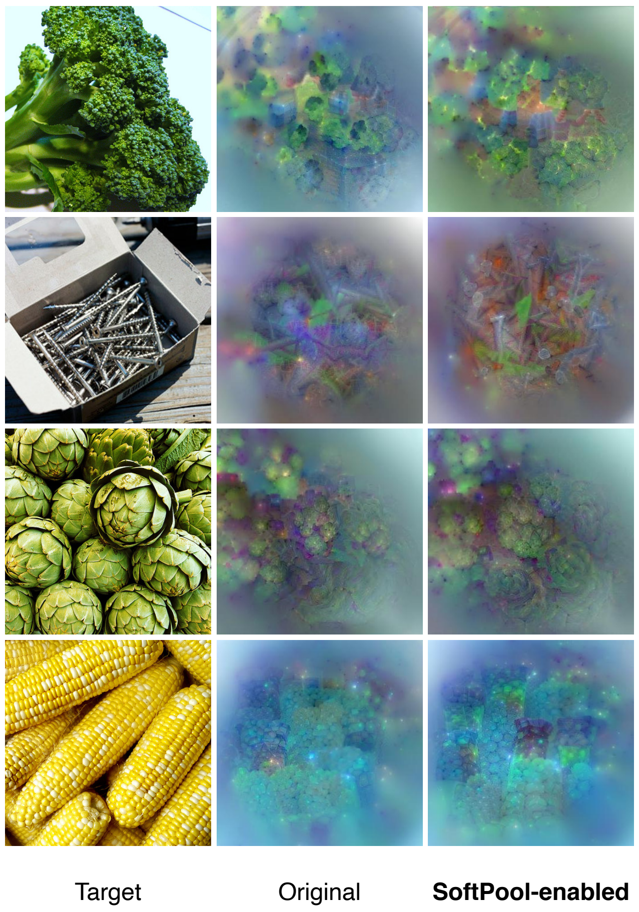


Figure 8. **Neuron activation maximization for InceptionV3 with original and SoftPool pooling layers.** We maximized the top-10 neurons [S2, S9] of the final block *(Mixed7c)* in InceptionV3 [36]. Target top-10 neuron combinations for each row were selected for ImageNet1K classes “broccoli”, “nails”, “artichoke” and “corn”.

all class examples. To eliminate additional noise, we use a mask-based approach similar to Wei *et al.* [S9]. However, in our setting, the mask is responsible for reducing the size of the gradient vectors for regions that are further away from the image center.

We used input images of size 512×512 in order to get higher-definition features. We use an SGD optimizer with an initial learning rate of 0.1 with a linear decrease to 0.01 over 2,000 total iterations. We use a weight decay of 1e-6. The weights for both models were initialized from those shown in Table 3 of the paper.

We show the outputs in Figure 8 for four ImageNet1K classes: “broccoli”, “nails”, “artichoke” and “corn”. We include in the left column the ImageNet1K images with the highest activations to provide representative examples. The majority of the features are fairly similar between the two models. Since SoftPool does not change the overall architecture nor the parameters, a high degree of similarity is expected. However, in cases such as the heads of the nails, we do notice a better definition of the objects. We also notice for the artichoke class that the structure of the petals and the thorns are more easy to distinguish from the network with SoftPool layers compared to the network with the original pooling layers. Although the differences remain small, by only changing the downsampling method used by the network, it can effect the robustness and improve the feature interpretability to a degree.

8. Spatio-temporal volume pooling

Pooling operations in time-inclusive volumes present the additional challenge of encoding time in the output. One large problem is between-frame motion as it can significantly impact the representation of spatial features within frames in the sub-sampled volume.

We show in Figure 3 with four different examples the effects of spatio-temporal pooling with max-pool, average pooling and SoftPool operations. As none of the methods is tailored towards completely alleviating the effects of encoded motion in the pooled output, they are visible in edges and regions where cross-frame motion exists. However, differences between the three methods do exist.

We demonstrate part of these differences in Figure 3(e). Where we include two cases of zoomed-in frame regions that show some variance based on the pooling method used. In the top one, gaps in the wooden planks of the floor are significantly less distinguishable within the max-pooled frame. Consequently, in the average pooled frame region, the nails are not visible at all anymore. This effect is in line with our observations for image-based downsampling of high-contrast and low-contrast regions. In contrast, SoftPool preserves features in both cases, which allows for the extraction of representative features after pooling.

9. Time-inclusive salient regions

To better understand the use of SoftPool in 3D-CNNs, we study the spatio-temporal regions that the network finds more informative. Similar to the activation maximization visualizations for images, we use a fixed network structure as a backbone and only study the variations produced by replacing the original pooling operations with SoftPool. For the visualizations in Figure 4, we use the r3d-50 network from Table 5 in the paper. The examples are sampled from the Kinetics-700 dataset from the classes “building lego”, “abseiling”, “baseball hit” and “archery”.

Based on the examples presented in Figure 4, there are no significant differences in the salient regions. However, in some cases such as in the abseiling example, the network with SoftPool does show that additional regions that are not part of a motion but are contextually associated with a target

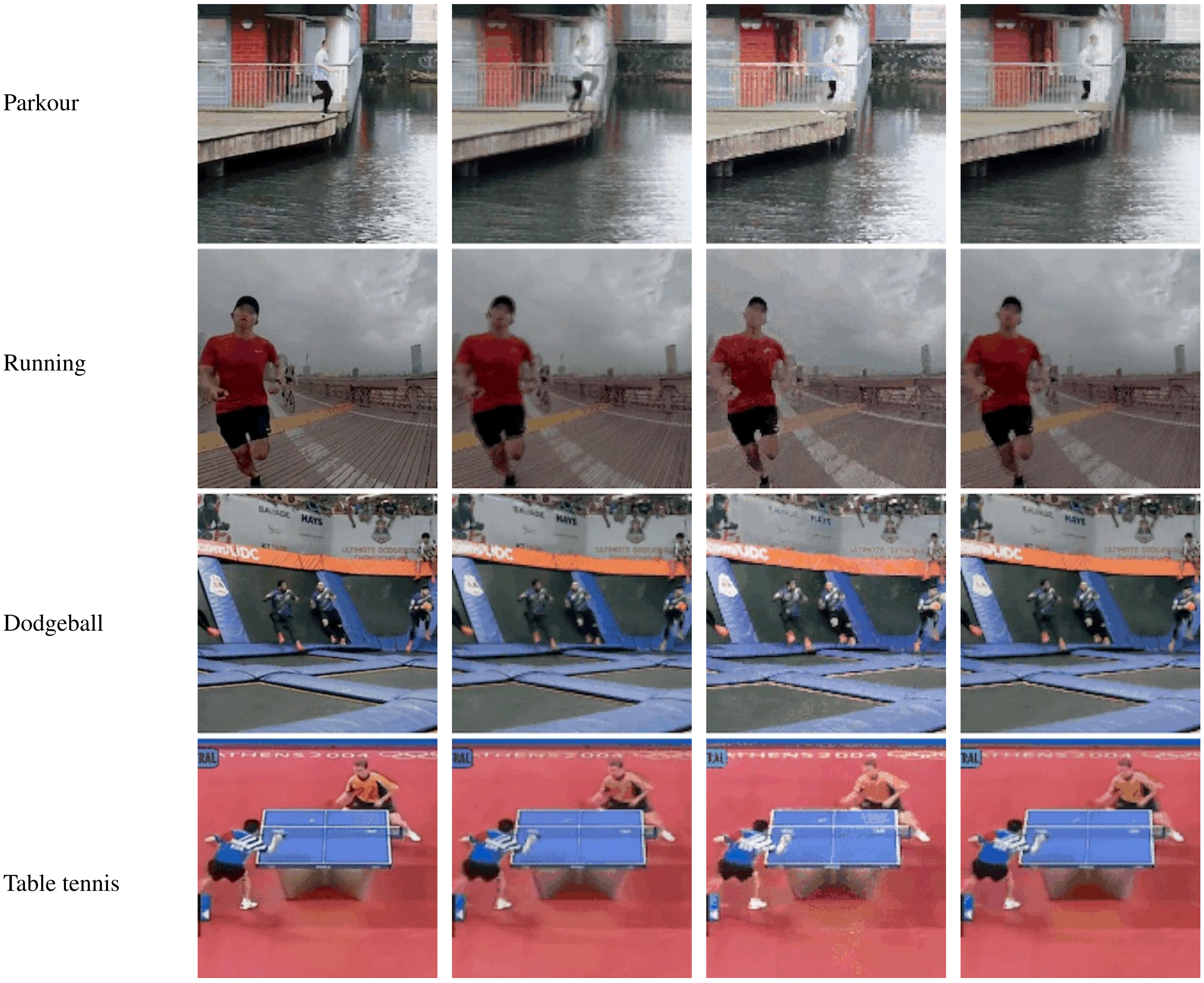


Figure 3. a. Original Figure 3. b. Avg. pool (3D) Figure 3. c. Max-pool (3D) d. SoftPool (3D)

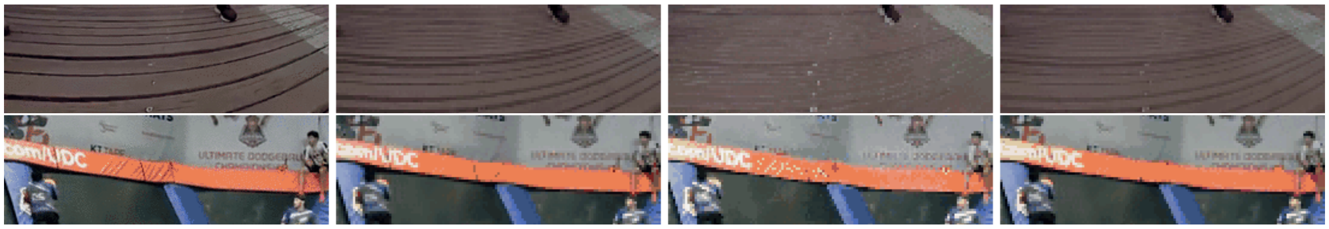


Figure 3. e. Zoomed-in frame regions

Figure 3. **Spatio-temporally downsampled videos.** Sub-sampling over spatio-temporal volumes compared to original videos (a), and downsampled with average pooling (b), maximum pooling (c) and our proposed SoftPool (d). Two zoomed-in frame regions appear in (e).

action are also considered. In multi-object scenes such as for the "building lego" class, both networks focus on the region where there is a clear definition of the action performed (i.e. the hand with the lego brick). The last two cases of "baseball hit" and "archery" exhibit very small amounts of variations, with both either focusing on the action ("baseball hit") or the main actor within the video ("archery").

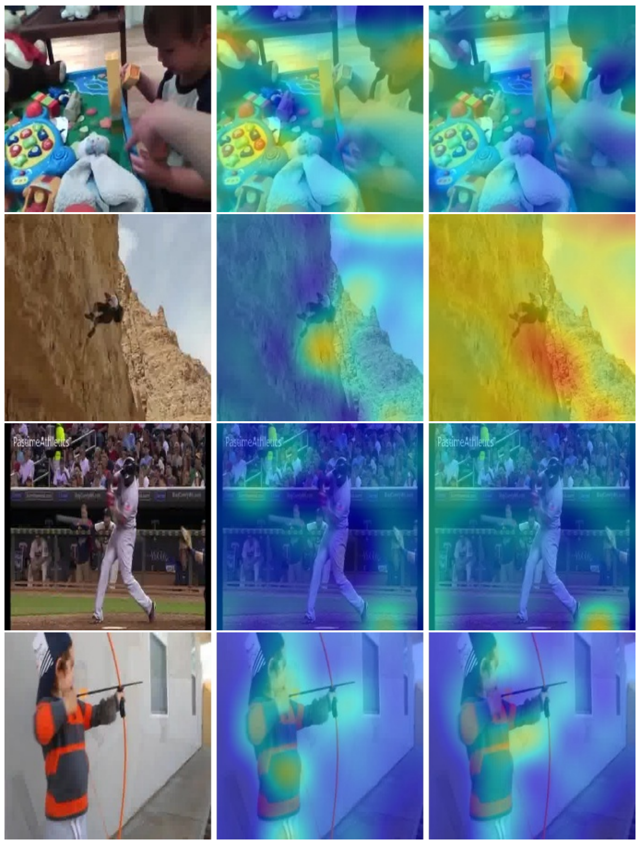


Figure 4. **Spatio-temporal saliency region visualizations for r3d-50 with and without SoftPool.** Class Feature Pyramids [S7] were used to generate the regional activations in the final *conv* layer of r3d-50.

10. Embedding spaces visualizations

We provide t-SNE [S5] visualizations of feature embeddings from an InceptionV3 model with original pooling operations and their counterparts with all pooling operations replaced by SoftPool. We use the averaged feature vectors of the final block in InceptionV3 (*Mixed7c*) with a reduced dimensionality of 50 channels produced by PCA [S3] and then perform t-SNE. We further perform k-means clustering [S4] to better represent the different sub-clusters within the embedding space. In well-defined feature spaces, images in clusters that are closer together should be more similar.

The visualizations in Figures 4-6 show distinct embeddings for the two networks. While structurally, both model yield similar embeddings, for some classes the differences are more apparent. For example, class "jack-o-lantern" (Figure 5) and "sax" (Figure 6) show more compact representations when SoftPool is used.

Supplementary References

[S1] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4829–4837. IEEE, 2016.

[S2] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[S3] IT Jolliffe. Principal component analysis. *Technometrics*, 45(3):276, 2003.

[S4] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[S5] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[S6] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[S7] Alexandros Stergiou, Georgios Kapidis, Grigorios Kalliatakis, Christos Chrysoulas, Ronald Poppe, and Remco Veltkamp. Class feature pyramids for video explanation. In *International Conference on Computer Vision Workshop (ICCVW)*, pages 4255–4264. IEEE, 2019.

[S8] 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. IEEE, 2016. 7

[S9] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside CNN. *arXiv preprint arXiv:1507.02379*, 2015.

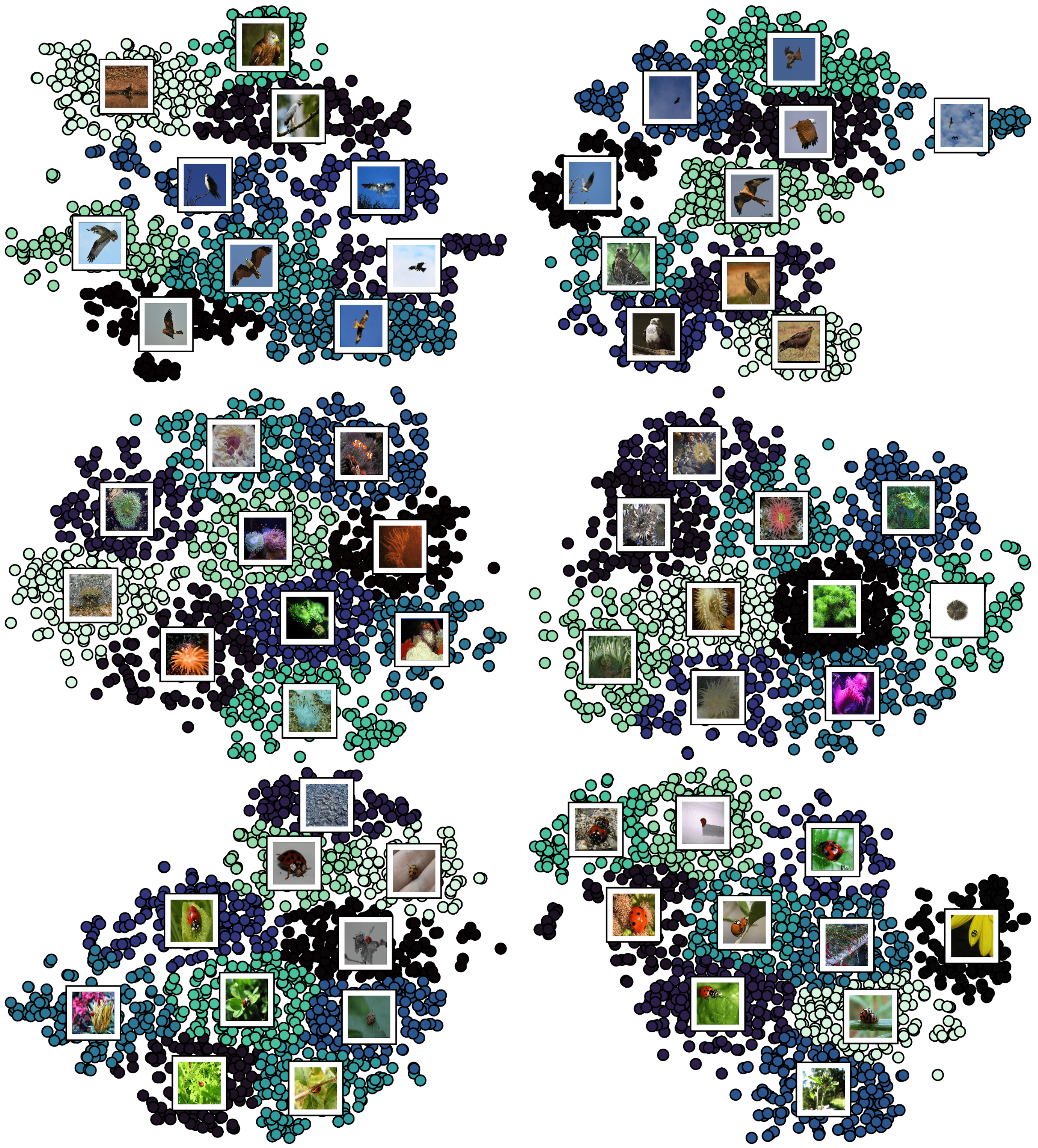


Figure 4. a. Original

Figure 4. b. SoftPool

Figure 4. **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes “bald eagle”, “sea anemone” and “ladybug”. Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.

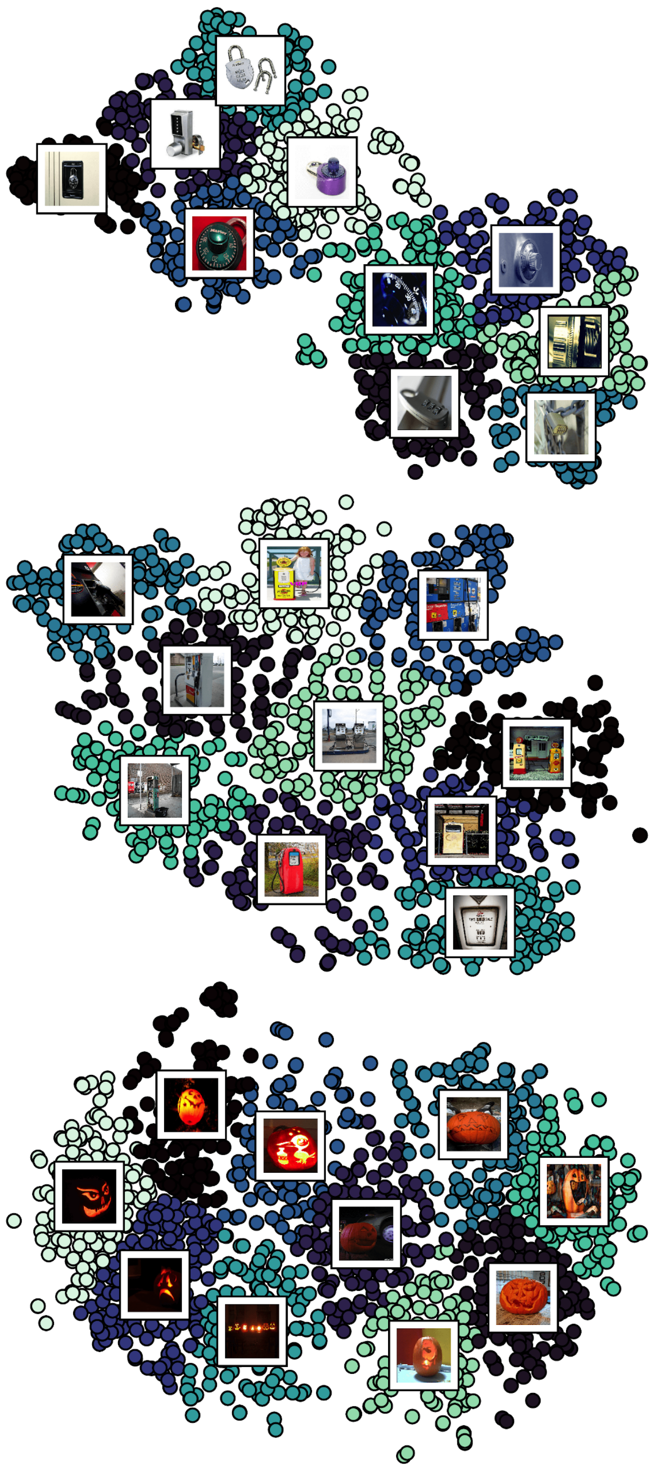


Figure 5. a. Original

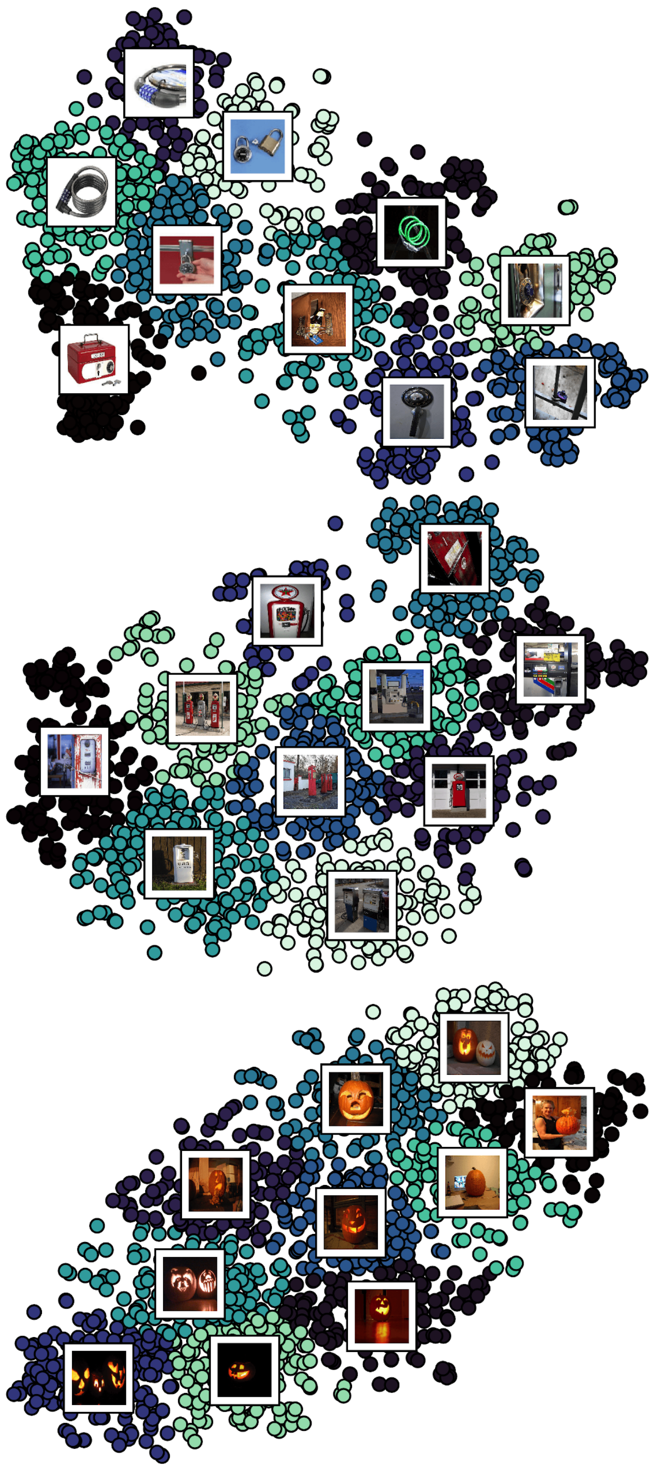


Figure 5. b. SoftPool

Figure 5. **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes “combination lock”, “gas pump” and “jack-o-lantern”. Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.

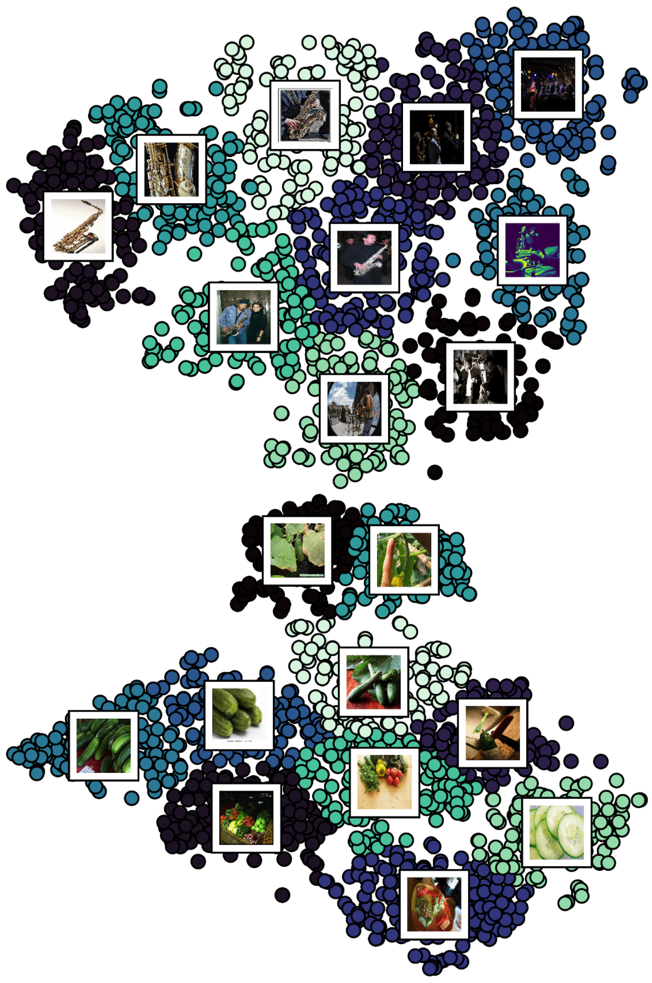


Figure 6. a. Original

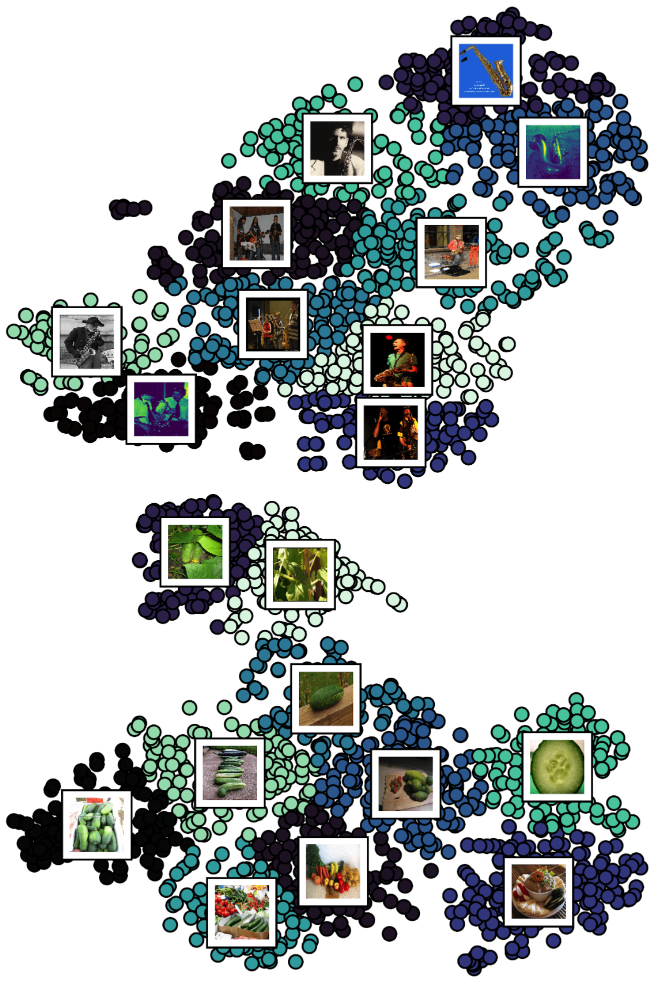


Figure 6. b. SoftPool

Figure 6. **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes "sax" and "zucchini". Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.